MVA '96 IAPR Workshop on Machine Vision Applications, November. 12-14, 1996, Tokyo, Japan

LAPCAM, Linear Array of Processors using Content-Addressable Memories: A New Design of Machine Vision for Parallel Image Computations

Eril Mozef, Serge Weber, Jamal Jaber, and Claude Bataille

Laboratoire d'Instrumentation Electronique de Nancy (L.I.E.N.)
University of Nancy I, BP 239
54506, Vandoeuvre Cedex, FRANCE
Tel: (33) 83 91 20 71, Fax: (33) 83 91 23 91.

Abstract

This work presents a CAM-based design for a linear array of processors. Called LAPCAM, the significance of this architecture is, in particular, that it provides the Multi-Mode Access (MMA) memories such as FIFO, RAM, normal CAM, and interactive CAM modes. In this paper, we present the organization of the LAPCAM and we demonstrate the ability to solve global image problems in processor-time optimal performance. For this purpose, connected component labeling, area and perimeter determination take O(n) time, for an n x n image, with a very small multiplicative constant factor. In this paper we also compare the performance of various architectures and discuss other intermediate-level vision algorithms.

1 Introduction

The efficient number of processors and suitability to current VLSI technology, compared to two- or three-dimensional array of processors, make linear array of processors rather attractive in parallel architecture. In this category, several architectures [1] as well as CAM (Content Addressable Memory)-based architectures [2], [4], [5], [6], have been proposed. A well-known architecture, called STARAN, using a more complicated memory system and interconnection network, has also been proposed [3]. Similar to this, we have proposed a preliminary version of LAPCAM [4]. The significance of this architecture is to provide Multi-Mode Access (MMA) memories which differ from Multi-Dimensional Access (MDA) memories of STARAN. Moreover, it presents a special configuration of interconnection network combining a linear structure and a tree structure of switches that ensure global communication in O(log n) units of propagation time.

In order to evaluate the performance of a parallel architecture, connected component labeling is often used. This is because its local and global features render labeling a difficult task and hence a good parameter for performance evaluation. The importance and relevance of labeling has caught the attention of a large number of researchers such that the focus is currently on more rapid processing using hardware solutions [7]. Although mesh reconfigurable, pyramid, and hypercube architectures have achieved an optimal performance in labeling, unfortunately, they are very cumbersome in terms of the number of processors. To overcome such shortcoming, a *processor-time optimal* (PTO) performance must be considered. A parallel algorithm for a given problem is said to be PTO if the product of the number of processors and the parallel execution time is equal to the sequential complexity of the problem [8].

In response to this PTO definition, several solutions have been proposed [7]. Unfortunately, these remain theoretical as, due to the large *multiplicative constant factor* (MCF) in time complexity, implementation remains impossible. However, an excellent solution, based on a sequential as well as parallel approach, which leads to PTO performances to a wide class of image and vision problems, has been proposed by Alnuweiri et al., [7], [8], [9]. Using this solution, the labeling complexity takes O(n) time, for an n x n image, in linear array. Unfortunately, the MCF obtained is not completely constant because it still depends on the value of n ($\alpha(n)$ function) which is expected to grow very slowly.

Besides presenting the organization of the LAPCAM, we also show that, by using a CAM-based solution, a *purely* PTO performance (in the sense that the MCF in time complexity does not depend on the n value) can be performed for global image problems. In this case, the complexity of connected component labeling, area and perimeter determination take O(n) time with a very small MCF. Finally, a comparison of performance on various architectures and other intermediate-level vision algorithms are discussed.

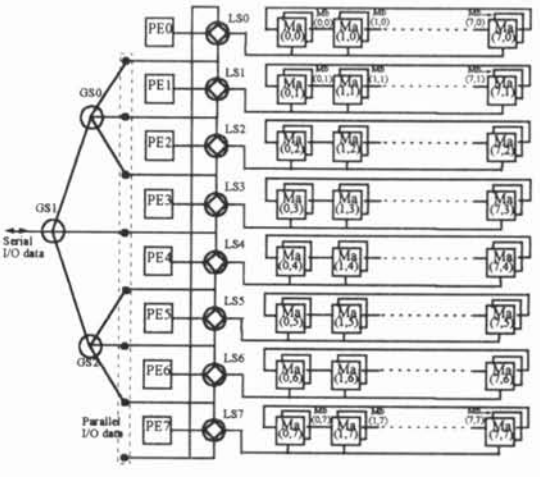

Fig. 1. The LAPCAM architecture

2 The LAPCAM Architecture

The LAPCAM architecture (Fig. 1) consists of memory modules, processor elements, and interconnection networks.

2.1 The Multi-Mode Access Memory

The memory modules are organized in two identical planes denoted $M_a[i,j]$ and $M_b[i,j]$, $(0 \le i,j \le n\text{-}1)$, of log

(n^2) bits (Fig. 1). Each plane consists of n rows, each of n memory modules. Each memory module provides Multi-Mode Access (MMA) memories such as FIFO, RAM, normal CAM, and interactive CAM modes. The FIFO mode is used to perform a data movement and to simplify a data transfer to/from the I/O bus. This mode allows a circular left/right shift operation. In the RAM mode, addressing of memory is based on *location*, while in the CAM mode, addressing of memory is based on *content*. In the normal CAM mode, CAMs with the same content can be simultaneously addressed and updated. The last mode, the interactive CAM mode, holds the particularity of this architecture. This mode allows a pair of two opposite memories (CAMs), in M_a plane and M_b plane, to operate in the interactive mode (Fig. 2). This means, one of the two CAMs can be addressed in order to write or read the content of its opposite location. This mode is similar to the BAM (Bidirectional Associative Memory) presented by Kosto [10] which is very useful in the field of neural network. The details of this mode can be found in [4]. The CAM mode is very attractive and rather promising as it allows addressing by an *object* in O(1) time (see section 2.2).

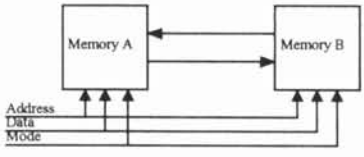

Fig. 2. A pair of memories in the multi-mode access memory.

2.2 The Processing Element (PE)

The LAPCAM consists of n PEs denoted PE_j ($0 \le j \le n-1$). The PEs operate in an SIMD mode. A number of PEs can simultaneously be activated or disactivated. A PE is able to compute a basic logical or arithmetic operation in O(1) time. It can communicate in O(1) units of propagation time either with its adjacent PEs or with its adjacent rows, row_j, row_{j+1}, or row_{j-1}. However, it communicates in O(log n) units of propagation time with either its non-adjacent PEs or its non-adjacent rows.

Through the CAM modes and the interconnection network, a PE allows addressing by an *object* (a group of pixels containing the same value) in merely O(1) time. This performs a *broadcasting operation* that achieves the same performance as that of a 2-d mesh architecture. In the normal CAM mode, a PE can address an *object* to update the contents. The interesting fact is that, in the interactive CAM mode, a PE can address an *object* to write/read the contents of its opposite object, or vice versa. These operations are undertaken in merely O(1) time. An example of this is given in Fig. 3. On the left, a PE addresses the plane A with a Target_Data=1 (corresponding to the objet "1") to update its opposite object in the plane B with a New_Data="blue". On the right, a PE addresses the plane B with a Target_Data="green" (corresponding to the color of the objet) to read the content of its opposite object in the plane A.

2.3 The Interconnection Network

The interconnection network is reconfigurable by n local switch modules denoted LS_j ($0 \le j \le n-1$), and n/2 -1 global switch modules denoted GS_g ($0 \le g \le n/2 -2$). Each switch module has an individual register to save a connection pattern. Different topologies are obtained with the use of differently stored patterns. One LS module is constructed by 4 connected switches (Fig. 1). This form allows all PEs to simultaneously write/read directly to/from their current row_j or their adjacent rows, row_{j-1} or row_{j+1} in merely O(1) time. The LS_{n-1} is reconnected to the LS_0 to perform a permutation network. One GS module is constructed by 3 connected switches. A tree structure of switches that is formed by the GSs is used to reduce the global communication in O(log n) units of propagation time. It also allows the transfer of serial/parallel input/output data to/from memories/PEs.

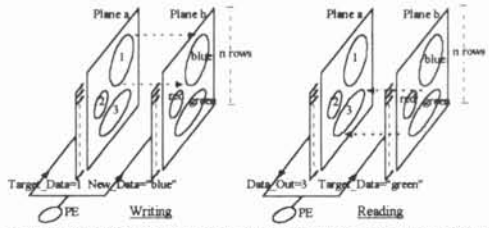

Fig. 3. The interactive CAM mode allows a PE to write/read based on an addressing by object, in O(1) time.

3 The Algorithms

The algorithms presented here use an image mapping technique to perform a PTO performance. This is similar to that proposed by Alnuweiri et al. [8]. Image mapping is a technique of image loading into the arrays following a special scheme in order to reduce communication distance and the amount of traffic on a single link of the array during subcomputation. Note that, in any case, the image loading is not included in complexity analysis [9]. In this mapping, the initial image is partitioned into n subimages each of $\sqrt{n} \times \sqrt{n}$. Each subimage is indexed using *the shuffled row-major distribution* and is then stored in one memory row according to *the block row-major distribution* (Fig. 4). The following algorithms consist of 2 phases, row-processing and merging. An initial gray-level image and 4-connectivity are assumed.

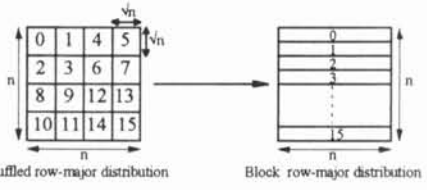

Fig. 4. Image mapping

3.1 Connected Component Labeling

In this algorithm, the mapped image is supposed to be available in the M_a plane in the form of a block row-major distribution whilst the M_b plane is used to store labels. Note that here, each gray level is simultaneously labeled.

Row processing: Each subimage is labeled using a sequential algorithm in O(n) time. Here, any sequential algorithm may be used to assign adjacent labels into a current position. However, when a pair of labels is found, instead of storing this pair into an equivalence table, 2 regions represented by the pair, are directly merged in O(1) time. In this case, the smallest label is preserved. In the following explanation, for simplicity sake, let us consider

that the coordinates of the subimages remain in the square form. The subimage stored in each row_{aj} is scanned. At each iteration, 3 adjacent pixels in the subimage, $P_{a(x,y)}$, $P_{a(x-1,y)}$ and $P_{a(x,y-1)}$ are tested. The operation is done if only $P_{a(x,y)} \neq 0$. **Initialization:** If ($P_{a(x,y)} \neq P_{a(x-1,y)}$) and ($P_{a(x,y)} \neq P_{a(x,y-1)}$), a row-major index corresponding to the position of $P_{a(x,y)}$ in the global image (not subimage) is assigned to $P_{b(x,y)}$. **Assignment of labels:** If ($P_{a(x,y)} = P_{a(x-1,y)}$) and ($P_{a(x,y)} \neq P_{a(x,y-1)}$) then $P_{b(x,y)} \leftarrow P_{b(x-1,y)}$. If ($P_{a(x,y)} \neq P_{a(x-1,y)}$) and ($P_{a(x,y)}=P_{a(x,y-1)}$) then $P_{b(x,y)} \leftarrow P_{b(x,y-1)}$. **Updating:** If ($P_{a(x,y)} = P_{a(x-1,y)}=P_{a(x,y-1)}$) and ($P_{b(x-1,y)} \neq P_{b(x,y-1)}$) then New_Data $\leftarrow$ Min[$P_{b(x-1,y)}$, $P_{b(x,y-1)}$] and Target_Data $\leftarrow$ Max[$P_{b(x-1,y)}$, $P_{b(x,y-1)}$]. Using the normal CAM mode, the New_Data and the Target_Data are used to update the row_{bj} in O(1) time. Since each iteration takes O(1) time, and there are n iterations, therefore this phase takes O(n) time.

Merging: This phase consists in recursively merging 2 subimages to obtain a larger subimage. Merging is carried out by scanning 2 adjacent boundaries belonging to the 2 subimages. This is initiated by vertical merging of the 2 smallest subimages and is continued with the horizontal merging of 2 larger subimages and so forth. This procedure is similar to the divide-and-conquer technique presented in [11] but the merging procedure remains similar to that presented in [4]. Each iteration of horizontal or vertical merging consists of a *minmax* and a *broadcasting operation*. Note that, in this phase, only the M_b plane is processed. In the **minmax operation**, 2 rows (corresponding to 2 adjacent boundaries of 2 subimages) are scanned. At each 2 different adjacent values found of the 2 adjacent boundaries, these values are read and compared. The largest of them is considered as the Target_Data whilst the smallest becomes the New_Data. In the **broadcasting operation**, the Target_Data and the New_Data are used to update the rows corresponding to the 2 subimages in O(1) time using the normal CAM mode. Note that, in each merging, these rows are physically connected by non-overlapping switches ensured by the image mapping. Hence, this allows a PE to perform the broadcasting operation in the connected rows in O(1) time. Since each iteration takes O(1) time, and there are $3(n-\sqrt{n})$ iterations, therefore this phase takes O(n) time.

Hence, the total complexity of our labeling is O(n) with a very small MCF which leads to a PTO performance.

3.2 Area Determination

Here, the previously labeled image stored in the M_b plane is used as the initial image whilst the M_a plane that will be used to store the results is initialized at 0. In this algorithm, the use of the interactive CAM mode is shown.

Row processing: This phase takes n iterations. At each iteration, incrementing and broadcasting operations are undertaken. In **incrementing operation**, each row_{bj} is scanned. At each label found, the opposite value in row_{aj} is read and is incremented in a register. This takes O(1) time. In **broadcasting operations**, the label is then used to address the row_{bj} whilst the incremented value is used to update the row_{aj} using the interactive CAM mode. This takes O(1) time. At the end of this phase, each row_{aj} represents the *subarea* of objects in row_{bj}. Since each iteration takes O(1) time, and there are n iterations, therefore this phase takes O(n) time.

Merging: Here, a similar procedure as that of labeling is undertaken, except that the minmax operation is replaced by the adding operation, and in the broadcasting operation, instead of using the normal CAM mode, the interactive CAM mode is used. In the **adding operation**, 2 rows in the M_b plane (corresponding to 2 adjacent boundaries of 2 subimages) are scanned. At the first 2 identical labels found belonging to an object across the boundaries, the opposite values, in M_a plane, are read and added in a register. This takes O(1) time. Note that the addition of the area of 2 objects must not be done more than once at the first 2 identical labels found. For that, a constant value must be added to the result of the addition to indicate that the subarea of the 2 objects has been added. Therefore, the presence of this constant must be considered before starting merging. In the **broadcasting operation**, one of the 2 labels is then used to address the rows in the M_b plane (corresponding to the 2 subimages), whilst the result of the addition is used to update the opposite rows in the M_a plane using the interactive CAM mode. This takes O(1) time. Since each iteration takes O(1) time, and there are $3(n-\sqrt{n})$ iterations, therefore this phase takes O(n) time.

Hence, our algorithm for computing area is O(n) which leads to a PTO performance.

3.3 Perimeter Determination

This algorithm uses a similar procedure to that described for area determination except that the added values belong to the perimeter of an object. Hence, the complexity of our perimeter determination is O(n), which leads to a PTO performance. Note that our algorithms for computing area and perimeter are independent of the number of connected components.

3.4 Histogramming

In this algorithm, an initial image (without image mapping) is supposed to be available in the M_a plane. The M_b plane that is initialized at 0 is used to store the result of histogramming in which its columns correspond to the gray-level values whilst its rows correspond to the number of pixels.

Row processing: Here, at each iteration i, each PE reads a gray-level value A in the i^{th} column of the M_a plane. The value B, in the A^{th} column of the M_b plane, is then incremented. Since each incrementing operation takes O(1) time and there are n iterations, therefore this phase takes O(n) time.

Sum-on-tree: Here, at each iteration i, each PE reads the value of the i^{th} column in the M_b plane. The sum-on-tree operation is employed to add all values stored in the PEs. Since each sum-on-tree operation takes O(log n) time and there are O(n) iterations, this phase takes O(n log n) time.

Hence, the complexity of our histogramming is O(n log n) which is optimal for $G \leq n$, where G is the number of the gray-level value.

4 Performance

Table 1 presents a comparison in the performance of various architectures. The reconfigurable mesh [11] pre-

sents the best performance in terms of algorithm complexities but this architecture presents an inefficient number of PEs and data propagation time, particularly with the use of the list-traversal technique [13]. The absence of interconnection network in systolic architecture [14], and of reconfigurability of mesh in fixed-mesh architecture [15], lowers the performance, particularly in labeling. The orthogonal architecture [8], [16], presents the best performance in terms of diameter and data propagation time. Using the image mapping technique, Alnuweiri et al. have shown that a PTO performance can be performed either in orthogonal or linear array architecture. However, note that, this is with the assumption that the MCF of $\alpha(n)$, in the time complexity, grows very slowly due to the use of Tarjan's algorithm [7]. However, without the image mapping technique, Balsara [16] has shown that labeling takes $O(n^2)$ time, whilst computing area and perimeter take O(Cn) time which depends on the number of connected components C. Here, these global *connected* image problems can be solved more efficiently by LAPCAM either with or without an image mapping technique. In addition, these algorithms neither depend on the form of the objects (in the case of labeling) nor on the number of connected components (in the case of computing area and perimeter). Moreover, the MCF in its time complexity is very small and is completely constant (it does not depend on the value of n). Hence, it presents a more efficient PTO performance compared to that proposed by Alnuweiri et al. Note that this complexity yields the same performance as that of a 2-d mesh architecture with n^2 processors [11]. However, a global *non-connected* image problem such as histogramming is less efficiently performed in LAPCAM, compared to the orthogonal architecture, because of an inefficiency in diameter communication.

Table 1. Comparison of performances on various architectures

Architecture / Features	2-d array of processors			1-d array of processors				
	Fixed Mesh [15]	Reconfigurable Mesh [11]		Systolic [14]	Orthogonal [8], [16]		Our Linear Array (LAPCAM)	
Processors	$O(n^2)$	$O(n^2)$		O(n)	O(n)		O(n)	
Memories	$O(n^2)$	$O(n^2)$		$O(n^2)$	$O(n^2)$		$O(n^2)$	
Diameter	O(n)	O(n)		O(n)	O(1)		O(log n)	
Technique / Algorithms	Local	Merging of regions [11]	List-traversal [13]	Local	without image mapping [16]	with image mapping [8]	without image mapping [4]	with image mapping
Propag. time	$O(n^2)$	O(n)	$O(n^2)$	O(n)	O(1)		O(log n)	
Labeling	$O(n^2)$	O(n)	O(log n)	$O(n^2)$	$O(n^2)$	$O(n *)$	O(n log n)	O(n)
Area	O(n)	O(n)		$O(n^2)$	O(Cn)	$O(n *)$	O(n log n)	O(n)
Perimeter	O(n)	O(n)		$O(n^2)$	O(Cn)	$O(n *)$	O(n log n)	O(n)
Histogram	$O(n^2)$	O(G log n) [11]		$O(n^2)$	O(n), (G≤n)		O(n log n), (G≤n) or $O(n^2)$, (G≤n^2)	

C : number of connected components. G : number of gray-level values.
* : multiplicative constant factor of $\alpha(n)$ which grows very slowly.

Helman et al. [12] have produced efficient image processing algorithms on linear array. Alnuweiri et al. [8] have shown that a number of other global image problems can be solved in O(n) time that lead to a PTO performance on linear array with n PEs. Such problems include computing the nearest neighbors of pixels and figures, the moment of each figure, etc. Our algorithms in the previous section have shown that LAPCAM is able to perform the PTO performance more efficiently. Since it has all possibilities of linear arrays, in addition, it has the possibility of the MMA memory and the efficient global communication, therefore any computation that can be performed on linear arrays can be performed more efficiently on LAPCAM.

5 Conclusion

A CAM-based design of linear array of processors using the MMA memories is presented. A number of global *connected* image problems, such as labeling, computing area and perimeter, by using LAPCAM, can reach a *purely* PTO performance. However, in global *non-connected* image problems such as histogramming, the architecture is slightly less efficient than the orthogonal architecture. Here, we can see that the use of CAM in the MMA memories shows that the LAPCAM is extremely efficient for intermediate level vision tasks. In addition, the efficient global interconnection network through the use of a tree structure of switches entails an excellent solution in the reduction of data propagation time and provides the possibility of real-time processing. Although no explicit implementation has been discussed, we are confident that the proposed architecture is well suited for current VLSI technology.

References

[1] V.K.P. Kumar, Parallel Architectures and Algorithms for Image Understanding, New York: Academic Press INC, 1991.

[2] B. Parhami, "Associative Memories and Processors: an Overview and Selected Bibliography," *Proc. IEEE*, vol. 61, no. 6, pp. 722-730, June 1973.

[3] K.E. Batcher, "STARAN Parallel Processor System Hardware," *Proc. AFIPS Nat. Comp. Conf.*, pp. 405-410, 1974.

[4] E. Mozef, S. Weber, J. Jaber, and E. Tisserand, "Design of Linear Array Processors with Content Addressable Memory for Intermediate Level Vision," *ISCA 9th Inter. Conf. on Parallel and Distributed Computing Systems*, Dijon-France, Sept. 96. (accepted)

[5] E. Mozef, S. Weber, J. Jaber, and E. Tisserand, "Parallel Architecture Dedicated to Connected Component Analysis," *IAPR 13th Inter. Conf. on Pattern Recognition*, Vienna-Austria, pp. 699-703, Aug. 96.

[6] E. Mozef, S. Weber, J. Jaber, and E. Tisserand, "Real-Time Connected Component Labeling on One-Dimensional Array Processors based on Content-Addressable Memory: Optimization and Implementation," *3rd Inter. Workshop on Image and Sig. Proces.*, Manchester-UK, Nov. 96. (accepted).

[7] H.M. Alnuweiri and V.K. Prasanna, "Parallel Architectures and Algorithms for Image Component Labeling," *IEEE Trans. on Pattern Analysis and Mach. Intell.*, vol. 14, no. 10, pp. 1014-1034, Oct.1992.

[8] H.M. Alnuweiri and V.K. Prasanna, "Optimal Image Computations on Reduced Processing Parallel Architectures," *In Parallel Architectures and Algorithms for Image Understanding* (V.K. Prasanna, Ed.). New York: Academic Press, pp. 158-183, June 1991.

[9] H.M. Alnuweiri and V.K. Prasanna, "Optimal Geometric Algorithms for Digitized Image on Fixed Size Linear Arrays and Scan Line Arrays," *IEEE Inter. Conf. on Comput. Vision Pattern Recognition*, Ann Arbor, pp. 931-936, 1988.

[10] B. Kosto, "Bidirectional Associative Memories," *IEEE Trans. on System, Man, and Cybernetics*, vol. 18, no. 1, pp. 49-60, Feb. 88.

[11] M. Maresca and H. Li, "Connected Component Labeling on Polymorphic Torus Architecture," *IEEE Inter. Conf. on Comput. Vision Pattern Recognition*, Ann Arbor, pp. 951-956, 1988.

[12] D. Helman and J. Jaja, "Efficient Image Processing Algorithms on the Scan Line Array Processor," *IEEE Trans. on. Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, pp. 47-56, Jan. 1995.

[13] S. Olariu, J.L. Schwing, and J. Zhang, "Fast Component Labelling and Convex Hull Computation Reconfigurable Meshes," *Image and Vision Computing*, vol. 11, no. 7, pp. 447-455, 1993.

[14] M. Annaratone et al., "The Warp Computer: Architecture, Implementation, and Performance," *IEEE Trans. on Computers*, vol. C-36, no. 12, pp. 1523-1538, Dec. 1987.

[15] M.J. Duff and T.J. Fountain, Cellular Logic Image Processing, New York: Academic, 1986.

[16] P.T. Balsara and M.J. Irwin, "Intermediate-Level Vision Tasks on a Memory Array Architecture," *Machine Vision and Applications*, vol. 6, pp. 50-65, 1993.