AD-A102 059 MARYLAND UNIV COLLEGE PARK COMPUTER VISION LAB F/G 9/2
CELLULAR ARCHITECTURES: FROM AUTOMATA TO HARDWARE.(U)
MAY 81 A ROSENFELD AFOSR-77-3271
UNCLASSIFIED TR-1048 AFOSR-TR-81-0557 NL

1 OF 1
AD
A102059

END
DATE
FILMED
8 81
DTIC

COMPUTER SCIENCE
TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

LEVEL II

9

TR-1048
AFOSR-77-3271

May 1981

CELLULAR ARCHITECTURES: FROM AUTOMATA TO HARDWARE

Azriel Rosenfeld
Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

Cellular automata have been studied for many years both as pattern generators and as acceptors for pattern languages. At the same time, computer architectures analogous to two-dimensional cellular automata have been proposed and used for image processing and recognition. In recent years, various extensions to the basic cellular automaton concept have been proposed. These extensions should be of interest in connection with the design of future hardware systems for image processing and analysis.

The support of the U.S. Air Force Office of Scientific Research under Grant AFOSR-77-3271 is gratefully acknowledged, as is the help of Sherry Palmer in preparing this report.

1. Introduction

Two-dimensional cellular automata - arrays of finite-state machines that operate in parallel and each of which can communicate with its neighbors - were introduced nearly 30 years ago as models for pattern generation and self-reproduction. Over 20 years ago, a two-dimensional cellular architecture (i.e., an array of processing elements) was proposed by Unger for image processing applications [1,2]. More recently, some attention was given to the use of cellular automata as acceptors for two-dimensional formal languages [3,4]. Three monographs dealing primarily with cellular automata were published in the USSR, Germany, and the USA at the end of the 1970's [5-7].

Since Unger's work, a succession of hardware systems incorporating cellular parallelism have been proposed or built. Notable among these are McCormick's ILLIAC III, involving a 36 by 36 array [8] (in contrast, the later ILLIAC IV has only 64 processors); Duff's CLIP IV, 96 by 128 [9]; ICL's DAP (128 by 128 [10]; and NASA's MPP (128 by 128) [11].

In recent years, the classical cellular automaton model has been generalized in a number of ways. The assumption that the individual cells are finite-state machines (i.e., each cell has a bounded amount of memory, no matter how many cells there are) is overly restrictive in practice; by assuming, e.g., that a cell's memory grows logarithmically with the number of cells, the power and speed of the system can be increased in a variety of ways. More interestingly, the topology of the system can be generalized, e.g., by permitting communication over power-of-2 distances, or by building a "pyramid" of arrays and allowing communication vertically as well as (or instead of) horizontally. Still more generally, one can consider cellular automata in which the "neighbor" relation defines an arbitrary graph of bounded degree (or perhaps degree logarithmic in the number of cells), rather than a regular array. Finally, one can allow the neighbor relation to change during the course of the computation.

This paper reviews basic concepts about cellular automata and their generalizations, and suggests that some of these generalizations may be of interest to the designers of future hardware systems for image processing and recognition.

2. Bounded cellular automata (BCA's)

Traditionally, a two-dimensional bounded cellular automation (BCA) is an array (say rectangular) of finite-state machines ("cells") all having the same transition function, where the new state of a cell depends on the current states of itself and its neighbors in the array. In more practical terms, we can think of a BCA as an array of processing elements each of which has a finite amount of memory, and all of which operate synchronously, in discrete time steps, in accordance with the same stored program. Each processor initially receives a piece of input data, and at subsequent time steps, each processor accepts inputs from its neighbors.

When a BCA is used for image processing or analysis, the input data given to each processor is the gray level of a pixel (or block of pixels), with neighboring processors getting data from neighboring pixels or blocks. For simplicity, we will usually assume a single pixel per processor from now on. We will not consider here how images are input to or output from the BCA; to avoid the need to consider I/O time, we may suppose that the input images are sensed directly by an array of sensing elements, one per processor, and that output images are directly displayed by an array of light-emitting elements, one per processor.

The greatest advantages of BCA's over conventional computers is that BCA's can perform local operations on the input image in parallel, so that the computation time required grows only with the complexity of the operation, not with the image size (at the price, of course, of requiring the number of processors to grow with the image size). Their advantage is somewhat reduced when we use them to compute properties of the image, or to make decisions on the basis of such properties ("acceptance", in automata theory terminology). Here the interprocessor communication structure (only neighbors can communicate) causes the computation time to grow with the image diameter. As a simple example, if we want to compute the histogram of the image (i.e., count how often each gray level occurs in it), we must send signals representing each value to a common location so they can be summed, and some of these signals must travel a distance on the order of the image diameter.

Some of the BCA algorithms for efficient property measurement or recognition are quite nonobvious (see, e.g., [5] for a collection of such algorithms). An example is the shrinking-and-shifting process that can be used to determine, for a given binary-valued input image, whether the 1's (or 0's) are connected, in time on the order of the image diameter. Labelling the connected components can also be done in O(diameter) time, but the algorithm is even less obvious [12]. Automata theorists have made important contributions to our understanding of how arrays of processors can be used efficiently for basic image processing tasks.

3. Extensions

3.1 Augmented memory

In a classical BCA, each cell has a bounded amount of memory, no matter how many cells there are. This implies, in particular, that a cell cannot know its address in the array, since its memory may not be large enough to hold that address; and a cell cannot explicitly address cells more than a bounded distance away. Historically, when the pattern generation capabilities of BCA's were studied, efforts were even made to minimize the number of states (i.e., the amount of memory) in each cell. Keeping the amount of memory per cell low does indeed reduce the hardware cost of a BCA; but from a practical viewpoint, there is no reason not to allow the memory per cell to grow, e.g., logarithmically, with the number of cells. Logarithmic growth makes operations such as histogramming easier (the values can be summed by a single cell, rather than using a set of cells as a counter), and also makes it easy to compute such properties as moments and the auto-correlation. It also facilitates connected component labelling (e.g., use the coordinates of a distinguished cell in each component as a label), run length coding, border coding, medial axis transformation, and quadtree construction, as well as computation of region properties such as area, perimeter, height, width, diameter, compactness, alongatedness, and convexity. For further details on algorithms for these tasks see [13].

3.2 Augmented topology

The speed of many BCA operations can be increased by allowing cells to communicate not just with their immediate neighbors, but with cells at distances 2, 4, 8,... [14,15]. (On neighborhood size tradeoffs in BCAs see [16]). This allows information to be sent to a common destination in time proportional to the log of the BCA's diameter. An alternative idea [5, Ch. 6; 17] is to use a "pyramid" of BCAs, each half of the size of the preceding (e.g., the sizes are $2^n x 2^n$, $2^{n-1} x 2^{n-1}$,...,2x2, 1x1), where each cell communicates not only with its "brother" neighbors on its own level of the pyramid, but also with four "sons" on the level below and with a "father" on the level above. Note that the total number of cells is $2^n x 2^n (1 + \frac{1}{4} + \frac{1}{16} \ldots) < 2^n x 2^n \times 1\frac{1}{3}$, not much greater than the number of cells in the base of the pyramid alone. Here the height of the pyramid is the log of the BCA's diameter, and tasks such as histogramming can be performed in O(log diameter) time using the apex node of the pyramid as the counter. Many of the algorithms for such pyramid BCAs require communication upward only, and require the memory in a cell to grow (at most) with its level in the pyramid, not (otherwise) with the number of cells, so that the total amount of memory is proportional to that of an ordinary BCA.

4. One-dimensional BCAs

Two-dimensional BCAs are rather expensive to build, and the largest ones now in existence are 128 x 128 arrays. For the same cost, one could build very large one-dimensional BCAs, consisting of tens of thousands of cells. (Indeed, the MPP has the option of operating as a 16,000-cell one-dimensional BCA.) Such BCAs could be used for fast parallel processing of various types of waveforms; one might, for example, use two of them alternatingly, so that one processes the previous waveform segment while the other loads the current segment. Further speedup is possible by using a "triangle" of such BCAs, each half of the size of the preceding, where each cell communicates with its two brothers on its own level, two sons on the level below, and one father on the level above; here the total number of cells is $2^n+2^{n-1}+\ldots<2^{n+1}$, and the processing time for many tasks is $O(n)$. In this section we briefly mention two other possibilities for using one-dimensional BCAs in image-related tasks.

4.1 Serial-parallel machines

A one-dimensional BCA can be used to scan an image one row at a time, operating in parallel on each row and "moving" sequentially from row to row. (Again, imagine two of them operating alternately on video data, one processing the previous row while the other loads the current row.) Algorithms for such "parallel-sequential" BCAs are given in [5, Ch. 7; 18]; as an example, if the cells have memory proportional to the log of the row length, it is easy to do histogramming in time proportional to the image diameter. Local operations, however, now also take O(diameter) time, rather than O(constant) time.

4.2 Chain code processing

One-dimensional BCAs can also be used to process border or curve information represented by chain codes. For example, such a BCA can determine, in O(length) time, the intersections (if any) of two given codes, and can determine the code(s) of the borders of the union or intersection of the regions having the given codes as borders. Various algorithms for chain code analysis using one-dimensional BCAs are given in [19].

5. Graph-structured BCAs

One-and-two-dimensional BCAs, pyramids, etc. are all composed of cells each of which communicates with a fixed number of neighbors (ignoring border effects). More generally, [20-22], one can consider graph-structured BCAs in which the neighbor relation defines an arbitrary graph of bounded degree. If the amount of memory per cell is allowed to grow logarithmically with the number of cells, one can also allow the degree of the graph to grow in the same way.

A BCA that has a fixed graph structure is of limited interest unless there are many sets of input data to be analyzed that all have the same graph structure (e.g., all images are arrays). In image analysis, various types of graph structures do arise (e.g., the adjacency graph of a segmentation of an image into regions), but they differ from image to image, and even vary in the course of processing a single image (e.g., if regions merge or split). Thus it is of greater interest to study graph-structured BCAs in which the initial graph structure can be defined arbitrarily and can then modify itself in the course of a computation.

A class of self-modifying graph-structured BCAs is defined in [23-25]. It is shown in [24,26,27] how such BCAs can be initially configured to represent a given segmentation of a given image, e.g., in terms of its region boundary segment graph or its quadtree. (The representation in terms of the region adjacency graph is simpler, but requires graphs of degree that can grow with the image size.) It is also shown how such a BCA can modify its configuration as the image representation changes, e.g., as regions split or merge, and can perform subgraph matching in parallel, avoiding the need for combinatorial search.

Arbitrarily graph-structured BCAs may not be easy to implement in hard-wired form unless they have regular structures (array, tree, etc.). On the other hand, conventional multiprocessor communication systems can be used to simulate BCAs that have other fixed or variable graph structures. For example,the ZMOB system [28], a collection of 256 Z80A microprocessors that communicate via a fast bus, can be used to simluate a reconfigurable graph-structured BCA having up to 256 cells. Such a simulation would be adequate for many real-world tasks involving region-level image processing.

6. Concluding remarks

Two-dimensional BCAs are a classical model for image processing and analysis at the pixel level. Such BCAs are beginning to be built in reasonable sizes, but are still quite costly, and are limited in speed for some tasks due to communication delays. Their performance can be speeded up by extending them in various ways: cell memory that grows logarithmically with the number of cells; connections to cells at power-of-2 distances, or the use of "pyramids" of BCAs. At the same time, there exist types of BCAs that can be built today at reasonable cost and that can solve practical problems - e.g., one-dimensional BCAs for processing waveforms or chain codes, or for row-by-row processing of images. Graph-structured BCAs whose structure varies in the course of a computation can be used for image analysis at the region level; here again, the number of cells required is not very great, and a variable-structure BCA can be simulated by a multi-microprocessor system that allows sufficiently flexible inter-processor communication.

References

1. S.H. Unger, A computer oriented toward spatial problems, Proc. IRE 46, 1958, 1744-1750.

2. S.H. Unger, Pattern detection and recognition, ibid. 47, 1959, 1737-1752.

3. A.R. Smith III, Cellular automata and formal languages, Proc. 11th SWAT, 1970, 216-224.

4. A.R. Smith III, Two-dimensional formal languages and pattern recognition by cellular automata, Proc. 12th SWAT, 1971, 144-152.

5. A. Rosenfeld, Picture Lanuages: Formal Models for Picture Recognition, Academic Press, NY, 1979.

6. R. Vollmar, Algorithmen in Zellularautomaten, Teubner, Stuttgart, 1979.

7. V. Aladyev, Mathematical Theory of Homogeneous Structures and their Applications, Valgus, Tallinn, 1980.

8. B.H. McCormick, The Illinois pattern recognition computer - ILLIAC III, IEEE Trans. EC-12, 1963, 791-813.

9. M.J.B. Duff, A cellular logic array for image processing, Pattern Recognition 5, 1973, 229-247.

10. P. Marks, Low-level vision using an array processor, Computer Graphics Image Processing 14, 1980, 281-292.

11. K.E. Batcher, Design of a massively parallel processor, IEEE Trans. C-28, 1980, 836-840.

12. S.R. Kosaraju, Fast parallel processing array algorithms for some graph problems, Proc. 11th STOC, 1979, 231-236.

13. C.R. Dyer and A. Rosenfeld, Parallel image processing by memory-augmented cellular automata, IEEE Trans. PAMI-3, 1981, 29-41.

14. R. Klette, A parallel computer for digital image processing, EIK 15, 1979, 237-263.

15. R. Klette, Parallel operations on binary images, Computer Graphics Image Processing 14, 1980, 145-158.

16. A.R. Smith, Cellular automata complexity trade-offs, Info. Control 18, 1971, 466-482.

17. C.R. Dyer and A. Rosenfeld, Triangle cellular automata, Info. Control, in press.

18. A. Rosenfeld and D.L. Milgram, Parallel sequential array automata, Info.Proc. Letters 2, 1973, 43-46.

19. T. Dubitzki, A. Wu and A. Rosenfeld, Parallel computation of contour properties, IEEE Trans. PAMI-3, 1981, in press.

20. P. Rosenstiehl, J.R. Fiksel and A. Holliger, Intelligent graphs: networks of finite automata capable of solving graph problems, in R.C. Read, ed., Graph Theory and Computing, Academic Press, New York, 1972, 219-265.

21. A. Wu and A. Rosenfeld, Cellular graph automata (I and II), Info. Control 42, 1979, 305-353.

22. A. Wu and A. Rosenfeld, Sequential and cellular graph automata, Info. Sciences 20, 1980, 57-68.

23. A. Wu and A. Rosenfeld, Local reconfiguration of networks of processors, TR-730, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, February 1979.

24. T. Dubitzki, A. Wu and A. Rosenfeld, Local reconfiguration of networks of processors: arrays, trees, and graphs, TR-790, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, July 1979.

25. A. Rosenfeld and A. Wu, Reconfigurable cellular computers, TR-963, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, October 1980.

26. T. Dubitzki, A. Wu and A. Rosenfeld, Region property computation by active quadtree networks, IEEE Trans. PAMI-3, 1981, in press.

27. A. Rosenfeld and A. Wu, Parallel computers for region-level image processing, Pattern Recognition, in press.

28. C. Rieger, J. Bane and R. Trigg, ZMOB: a highly parallel multiprocessor, Proc. IEEE Workshop on Picture Data Description and Management, 1980, 298-304.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 81 -0557	2. GOVT ACCESSION NO. AD-A1Ø2	3. RECIPIENT'S CATALOG NUMBER Ø59
4. TITLE (and Subtitle) CELLULAR ARCHITECTURES. FROM AUTOMATA TO HARDWARE		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Azriel Rosenfeld		8. CONTRACT OR GRANT NUMBER(s) AFOSR-77-3271
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Vision Laboratory, Computer Science Ctr University of Maryland College Park MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB DC 20332		12. REPORT DATE MAY 81
		13. NUMBER OF PAGES 14
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Cellular automata; image processing; pattern recognition; parallel processing.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Cellular automata have been studied for many years both as pattern generators and as acceptors for pattern languages. At the same time, computer architectures analogous to two-dimensional cellular automata have been proposed and used for image processing and recognition. In recent years, various extensions of the basic cellular automaton concept have been proposed. These extensions should be of interest in connection with the design of future hardware systems for image processing and analysis.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)