The Society of Photo-optical Instrumentation Engineers Los Angeles Symposium Algorithms for Image Processing.
January, 1985, 5 pp.

554-06

System design for local neighborhood processing

Patrick F. Leonard*
Trevor N. Mudge**

*Synthetic Vision Systems, Inc., 3744 Plaza Drive, Ann Arbor, Michigan 48104
**Electrical Engineering and Computer Science Department, University of Michigan
Ann Arbor, Michigan 48109

Abstract

A class of processors which perform local neighborhood or cellular operations has well-documented applications in image processing. The architectural feature which unites these processors is their use of special-purpose hardware to compute image transformations based both on the values and on the spatial relationship of pixels in the input image. Typical operations include 3 x 3 convolution and mathematical morphology. Processing elements which compute cellular operations have been incorporated in a variety of system architectures. These range from single-processor, recirculating buffer systems to systems which have thousands of processors in a four-way interconnected mesh. Another feature which unites these processors is that they operate on images in the iconic domain. A new class of non-iconic processor is introduced here which uses image encoding schemes to reduce the number of operations required for certain classes of operations by between one and two orders of magnitude.

Introduction

The computation required to solve typical machine vision applications places rigorous demands on computer architectures for several reasons. The first is the volume of data to be operated on. For instance, a high quality, monochrome video signal can be digitized to produce approximately 7.8 Mbytes of data per second, with a single image containing as many as 256 Kbytes. The second reason is the amount of processing per image required. A typical machine vision algorithm may require 50 to 100 operations to be performed on each image to extract the desired information. Third, in high-volume manufacturing applications, these decisions must be made at a rate of up to several per second. Finally, there is the constraint on system cost imposed by the market. The combination of these constraints results in specifications for system architectures which must perform billions of operations per second at an extremely low cost per operation.

Several aspects of the image processing problem have been exploited to attempt to achieve the requisite specifications in both absolute performance and price/performance. The first is that since images regardless of the modality of the sensor, are typically regular arrays, computer architectures can be constructed which use this feature to simplify processor architectures. The second is that many image operations can be formulated as translation invariant operations, thus making it possible to exploit inherent parallelism and simplify the data flow problem. Thereby, a large class of useful image processing operations can be decomposed into local operations, making it possible to construct large processors by replicating regular processing elements, called cells, and interconnecting them in various patterns. The cells themselves differ in internal architecture from system to system, depending upon the type of computation desired. We will describe several different cells and the system architectures in which they occur. In addition, a new class of processors will be described which can compute many of the same operations on images with possible increases in efficiency of up to two orders of magnitude, depending upon the data being processed. This class of processors uses an image encoding scheme to reduce the amount of data by associating semantics with the scene as efficiently as possible.

Background

Cellular architectures for image processing have been the subject of research for a number of years, and several groups have built machines to exploit the inherent parallelism in image processing. A brief overview of four different types of architectures will be presented here. More detailed analyses of the strengths and weaknesses of the various approaches are given in [1], [2] and [3].

1. Simple cellular. This refers to systems which use a single cellular computation unit tightly coupled to a recirculating image memory to process data. A diagram of this type of system is given in Figure 1. Because of its ease of implementation, this architecture has found the most widespread commercial acceptance. A typical application of this type of architecture is to compute a 3 x 3 convolution kernel in parallel. An example of this type of processor is the Vicom VDP which includes software to decompose an

534-06

arbitrarily sized Finite Inpulse Response convolution kernel into 3 x 3 kernels for implementation on its cellular processor [4]. An example of this type of system architecture applied to non-linear kernels is the GLOPR machine [5]. This machine, and its descendants, compute new pixel values based on Golay transformations, in which the binary pattern formed by the neighbors of a pixel determine its new value [6]. Another example of this architecture is the PICAP II machine in which the cellular processor is only one of several special function units capable of being attached to a general-purpose data bus which provides high speed access to memories and mass storage and input/output devices [7].

2. Cellular arrays. Cellular array machines have their processor cells arranged in a mesh pattern, with processors communicating along the lines of the mesh. An example of this type of architecture is shown in Figure 2. Images are loaded into the array such that each pixel occupies a cell. This architecture therefore requires a great number of cells to be computationally efficient. This is generally made possible by constructing each processor as a bit serial device, thus allowing multiple processors to be integrated on a single VLSI chip. Machines constructed on this principle include the Clip4 [8] and the MPP [9]. Loading and unloading images with this type of processor typically represent a substantial amount of overhead in comparison to the actual time spent processing images. As a result, these machines are more suited for tasks where a lot of computation is done on a single image once it is loaded, as opposed to machine vision tasks where many images have to be processed in sequence.

3. Augmented cellular arrays. Attempts have been made to overcome the interconnection limitations of the cellular array architecture by augmenting the array with additional data paths. These are added to allow rapid movement of data across multiple cells. Several different organizations have been proposed to augment the cellular array [10]. One of the most interesting is the pyramid. In this arrangement, adjacent processors in the array are connected directly to processors in the next higher level. These processors are then connected to processors in the next higher level and so on until all processors are connected to a single processor which oversees the entire process. The theory is that each higher level of processor(s) will deal with the data at a higher level of abstraction, until the information needed to make a decision about the scene is communicated to the top-most processor. This eliminates the need to unload the image and process it further with a serial processor to extract information. Another example of augmented cellular arrays is the CAAPP architecture [11] in which a cellular array is augmented with a control processor capable of accessing the array associatively. In this arrangement, decisions based on processing pixels in a given state can be made by the control processor without examining the entire image. The ability to feedback global information into the cellular array facilitates the use of algorithms which require communication between iconic and symbolic levels of representation. The concept of a content addressable cellular processor is, in principle appealing but more work needs to be done to determine its applicability to machine vision.

4. Systolic cellular processors. Systolic cellular architectures have shown the most promise for solving machine vision problems in a cost-effective fashion. The term systolic refers to the manner in which data flows from memory through the processor and back to memory. Two of the benefits of this type of data flow organization are its ability to ensure a good match between the memory and processor bandwidths and its modularity. Implementation of systolic architectures for a variety of applications were explored by Kung [12]. The Cytocomputer, shown in Figure 3, is a linear systolic machine composed of a pipelined cellular processor [13]. Originally, this machine computed only neighborhood operations based on mathematical morphology but has recently been extended to compute some 3 x 3 linear convolution kernels and to perform data compression and expansion between iconic and non-iconic image domains.

Non-iconic image domains

Cellular architectures may differ in organization or function but they will have at least one thing in common: they all operate in the iconic domain. The iconic domain is defined as consisting of those images in which the image data is represented directly as an array of pixels. This representation can often be space consuming, but has the advantage that the spatial relationship between pixels is preserved in the array indices. We will define as non-iconic those image representations which can be readily obtained from an iconic image by an invertible mapping, but in which the spatial relationship between data values is not directly determined by their position in the data structure. Examples of mappings which map images from an iconic into a non-iconic domain include run-length encoding (RLE) [4], Fourier descriptors [14] and other edge encodings such as Hough transforms [15]. A brief overview of run-list processing approaches to cellular operations is given in [16]. These mappings are distinguished by the property that they are isomorphic to the iconic with respect to certain classes of operations. In other words, for each operation R in the iconic domain there is an equivalent operation R' in the non-iconic domain which has the same result. Thus an image may be mapped into a non-iconic domain, be operated by R', and then mapped back into the iconic domain; the result is the same as if the iconic version of the image were operated on by R. Depending upon the particular non-iconic domain selected, these operations can include convolution, mathematical morphology and general image-image and image-scalar arithmetic. Of course, the intent is to choose a non-iconic domain that allows the operation of interest to be performed with much greater efficiency than in the iconic domain. The paradigm here is the FFT.

Non-iconic image representations should be differentiated from purely symbolic image representations in which semantics are associated with images. Symbolic representations imply the loss of spatial information which is still available, albeit in encoded form, in non-iconic representations. Symbolic representations include high-order labeling operators such as object or region identification, curve labeling or surface modeling. Non-iconic domains, however, can be very useful as an intermediate representation when communication is required between the iconic and symbolic levels. An example of this type of processing could be used in the formation of Marr's primal sketch [17]. To form a primal sketch, a cellular processor can be used to generate raw features based on zero crossings, bars, blobs and terminations. These features can be passed on to a non-iconic processor where further processing can be accomplished to reinforce the features and tentatively classify them for a higher level symbolic processor. When the symbolic level processor detects a tentative feature of interest the information can be passed back down to the iconic level via the non-iconic processor to modify the program in the cellular processor to suppress unwanted features and enhance features of interest. This process can iterate until some desired level of confidence in the feature is obtained.

<u>Non-iconic operations</u>

In addition to their value as an intermediate level between iconic and symbolic levels, non-iconic image representations have the potential to offer significant computational advantages over iconic representations. The motivation for considering non-iconic domains is found in the basic paradigm for machine vision as stated in the introduction: to reduce the amount of data by associating semantics with the scene as efficiently as possible. The advantage of operating in a non-iconic domain becomes clearer when examined in light of a generic machine vision algorithm. One of the first tasks in many machine vision algorithms is to segment the features of interest from the rest of the scene. After this point, the non-feature pixels, which generally form the majority of pixels in the image, are of no interest to the algorithm. Cellular processors, due to constraints placed on them by their design as iconic processors, must continue to operate on every pixel in the image whether or not it contributes meaningfully to the result. It is at this point that the features of interest would be encoded and passed on to a non-iconic processor for further processing. Processing on the features would then continue as if they were still represented in the iconic domain, with the important exception that far fewer total operations need be performed to obtain the same result.

Analysis of a program which performs the logical AND of two RLE images shows that the worst case execution path requires three comparisons and two assignments per RLE start/stop pair for a total of five operations per pass. The total number of passes through the program is equal to the minimum of the number of start/stop pairs in the two input images. If I is the number of pixels in an iconic image, C is the compression ratio (ratio of number of pixels in the iconic image to the number of data items in the equivalent non-iconic image) and N is the number of edge points in the RLE image which is derived from the iconic image, then $N = I/C$ and the complexity of the computation of the RLE AND operation is given by $o(AND) = 5KN = FKI/C$. The multiplicative constant K arises because the complexity of atomic operations in the iconic domain may differ from those in the non-iconic domain. For RLE and for the operations discussed here, K is typically between one and two. It is clear from the equation that a compression ratio of less than ten is required to break even. Thus for this operation and type of encoding, if fewer than one out of every ten pixels are edge points, the non-iconic processor requires fewer operations per image than the iconic processor. Results from a study of edge frequency in a circuit pattern inspection task indicate that the average compression ratio far exceeds ten. Samples from three different multilayer thick film circuits, imaged at a resolution of about 10 pixels/design rule with 245,760 bytes of data per image before segmentation yielded an average of 6486 edge points per image with a standard deviation of 4504 edge points per image after segmentation using a coding scheme which is equivalent to RLE. This represents an average compression ratio of about 40, or four times greater than is required to break even. Because of their geometric nature circuit patterns probably represent an extreme case in terms of compression ratio. Encoding schemes currently under development for circuit inspection tasks take advantage of the geometric regularity to reduce the same image data as in the above example down to an average of 1199 points, with a standard deviation of 653 for a resulting compression ratio of 205. While this may represent an extreme example, the principle extends to other types of images in which the average frequency of feature points yields favorable compression ratios.

534-06

Similar savings can be realized in computation of mathematical morphology operations on images. It can be shown that there are "2D" morphic operations on RLE representations of images that are equivalent to the familiar morphic operations such as dilation and erosion in the iconic domain. Analysis of the computation for RLE images shows that, as in the AND operation, the number of operations is proportional to the number of edge points. In addition, the structuring elements used to implement the morphic operations do not have to be decomposed into 3 x 3 or smaller kernels to be implemented in the non-iconic domain, and as a consequence it can be shown that the number of computations needed to perform a dilation or erosion is also proportional to the number of edge points in the structuring element. This is in contrast to a cellular processor where the number of computations needed to perform a dilation or erosion is also proportional to the number of edge points in the structuring element. This is in contrast to a cellular processor where the number of computations needed to perform a dilation is proportional to the total number of pixels. Another possible advantage is that non-iconic morphology allows non-contiguous structuring elements, which, for some operations, can reduce the number of computations even further. A description of non-contiguous morphic operations is found in [18].

Computation of morphic operations in a non-iconic domain is a straight forward application of the rules of morphology. The RLE representation of the structuring element is translated to each edge point in the RLE representation of the image to be transformed and then merged using the AND or OR operator. Assuming a worst case of five operations per edge point and two operations per translation, the number of calculations required to perform a dilation is: $0(+) = 7NS$, where (+) is the dilation operator, N is the number of edge points in the non-iconic image and S is the number of edge points in the structuring element. For a large class of shapes, S is proportional to the square root of the total number of pixels in the iconic version of the structuring element, E, and in the most pathological case would not grow at a rate of more than twice E. Thus the compression ratio for structuring elements varies between E and 2E. The relationship of N to the number of pixels in the iconic image is as in the previous example. Considering the $K=2$ factor, this yields a compression ratio of, at worst, no more than 25 to break even. Typically, the figure is much lower (between one and two). For a circular structuring element of radius five, no compression of the input image yields a 20 percent decrease in the number of operations required to compute a dilation. Thus for the 40:1 compression of the earlier example, the net result would be a dilated image using one-fiftieth of the number of operations required for dilation in the iconic domain.

System design

System design for an image processing system incorporating a non-iconic processor with a Cytocomputer is shown in Figure 4. Data would be input either directly into the Cytocomputer or into a memory buffer. The Cytocomputer would perform filtering and feature extraction operations and pass the information on to the non-iconic processor through an encode/decode function. The non-iconic processor would perform its processing and pass the processed data on to a control processor which would make decisions based on the cues passed to it by the non-iconic processor. Note that the data paths are bidirectional. This hierarchical architecture allows adaptive algorithms to be implemented as discussed in the section on non-iconic image domains.

A proposed architecture for a non-iconic image processing element which calculates image-image and morphic operations of RLE data is shown in Figure 5. Memories A, B, C and D contain image operands and results. The a,b and c,d register pairs hold the start/stop points for each image while the e,f register pair acts as an accumulator. When conditions are satisfied for creation of a result start/stop pair, it is output to the memory through the h,i registers. The computation is performed by a logical ranker which is connected directly to a microsequencer which sets the multiplexors and initiates a memory read based upon the results of the ranking operation. This machine has been simulated in software and is currently processing images. The instructions which have been executed include a complete set of arithmetic and logical operations and morphic operations including dilation, erosion and thinning operators. Extensions to this architecture will include the ability to handle multiple level state encoded data and types of data encodings other than RLE.

Conclusions

Non-iconic processors show potential for augmenting current cellular architectures. While the inability to efficiently handle greyscale data prevents the non-iconic processor from operating as the sole image processor in all situations, tasks that are inherently binary can be processed. As a co-processor, the non-iconic machine can be used to increase the throughput of an image processing system while reducing the total hardware cost over the equivalent all-cellular machine needed to achieve the same performance. In addition, computations not possible on a cellular processor, such as Hough transforms and other edge-based algorithms can be introduced by working in a non-iconic domain.

Acknowledgments

The authors are grateful to David Symanow, Donald Rohrer, Donald Svetkoff and Steven Spencer, all of Synthetic Vision Systems, for their information regarding recent extensions to the Cytocomputer instruction set and their applications to circuit inspection. We would also like to thank Robert Copeland of Princeton, NJ, for his contributions to our understanding of image encoding and for the desgin of the RLE processor. We would also like to thank Dolly Gillett for her patient preparation of the manuscript.

References

1. Reeves, A. P., "Parallel Computer Architectures for Image Processing," Computer Vision, Graphics and Image Processing, No. 25, 1984.
2. Duff, M. J. B., ed., "Computing Structures for Image Processing," Academic Press, London, 1983.
3. Rutenbar, R. A., Mudge, T. N., Atkins, D. E., "A Class of Cellular Architectures to Support Physical Design Automation," IEEE Transactions on CAD of IC's and Systems, Vol. CAD-3, No. 4, October 1984, pp. 264-278.
4. Pratt, W. K., "Digital Image Processing," Wiley Interscience, 1978.
5. Graham, M. D., and Norgren, P. A., "The diff3^{m} Analyzer: A Parallel/Serial Golay Image Processor," Real-Time Medical Image Processing, Morio Onoe, Kendall Preston, Jr. and Azriel Rosenfeld, eds., Plenum Publishing Corporation, New York, N.Y., 1981.
6. Golay, M. J. E., "Hexagonal Parallel Pattern Transformations," IEEE Transactions on Computers, Vol. C-18, No. 8, August 1969.
7. Danielsson, P-E., and Levialdi, S., "Computer Architectures for Pictorial Information Systems," IEEE Computer, Vol. 14, No. 11, November 1981.
8. Duff, M. J. B., Watson, D.M., Fountain, T. J., and Shaw, G. K., "A Cellular Array for Image Processing," Pattern Recognition, Vol. 5, pp. 229-247.
9. Potter, J. L., "Image Processing on the Massively Parallel Processor," IEEE Computer, Vol. 13, No. 1, January 1983.
10. Uhr, L., "Algorithm-Structured Computer Arrays and Networks," Academic Press, Orlando, Florida, 1984.
11. Lawton, D., Levitan, S., Weems, C., Riseman, E., Hanson, A., and Callahan, M., "Iconic to Symbolic Processing Using a Content Addressable Array Parallel Processor," SPIE Applications for Digital Image Processing VII, San Diego, CA, August 21-24, 1984.
12. Kung, H. T., "Why Systolic Architectures?'" IEEE Computer, Vol. 12, No. 1, January 1982.
13. Sternberg, S., "Parallel Architectures for Image Processing," Proceedings of the Third International IEEE COMPSAC, Chicago, 1979.
14. Chellappa, R., and Bagdazian, R., "Fourier Coding of Image Boundaries," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 1, January 1984.
15. Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes," Pattern Recongition, Vol. 13, No. 2, 1981, pp. 111-122.
16. Gerritsen, F. A., and Verbeek, P. W., "Implementation of Cellular-Logic Operators Using 3*3 Convolution and Table Lookup Hardware," Computer Vision, Graphics, and Image Processing, No. 27, pp. 115-123, 1984.
17. Marr, D., "Vision," W. H. Freeman and Co., San Francisco, 1982.
18. Sternberg, S., "Machine Vision Non-Contact Gaging," Proceedings of the Third Annual Machine Vision Conference, RI/SME, Schaumburg, IL, February 27 - March 1, 1984.

554-06

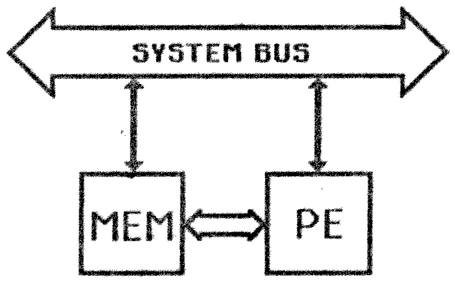

Figure 1. Simple Cellular Processor. A memory (MEM) is tightly coupled to a cellular processing element (PE).

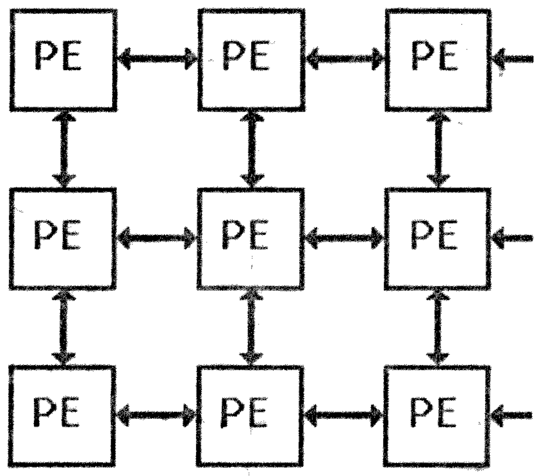

Figure 2. Cellular Array Processor. Cellular processing elements (PE's) are 4-way interconnected to form an array. Images are loaded through the PE's along one edge.

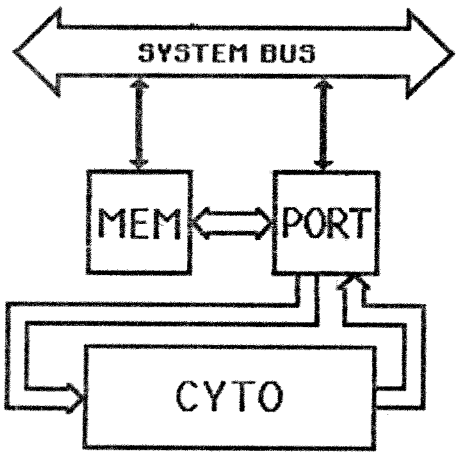

Figure 4. Systolic Cellular Processor. Image data flows from the memory (MEM) through the controller (PORT) to the multi-stage cellular processing pipeline (CYTO).

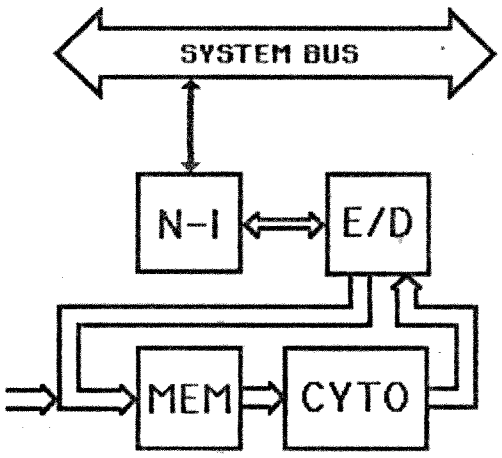

Figure 5. Non-iconic Processor. Images are input through the memory (MEM) and processed by the Cytocomputer (CYTO). Results are encoded (E/D) and passed on the non-iconic processor (N-1) for further processing.

53/-06

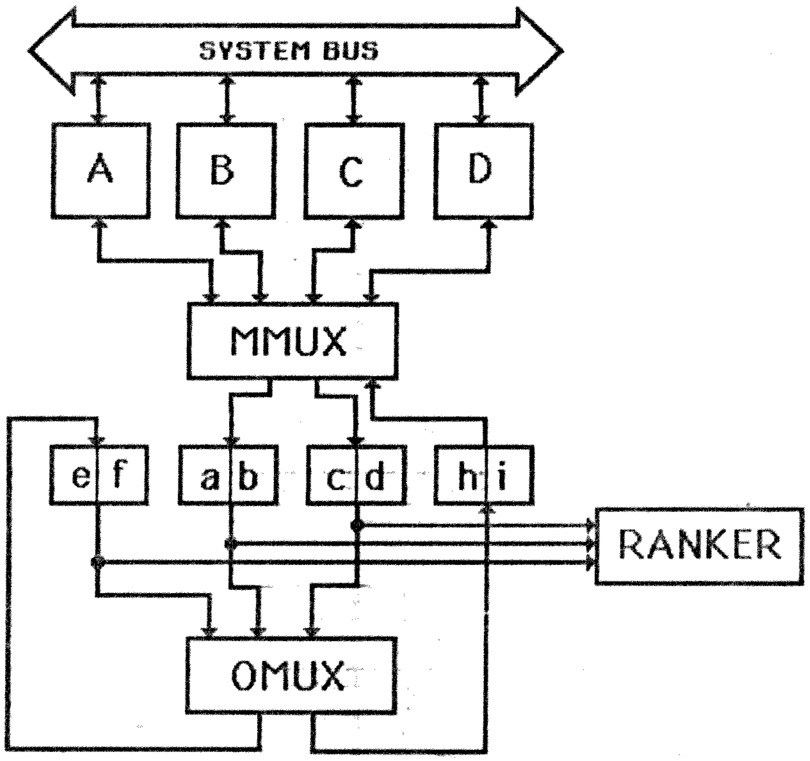

Figure 5. Non-iconic Processor. RLE data are stored in memories (A, B, C, D). Memory multiplexor (MMUX) selects start/stop pairs to be input to register pairs a,b and c,d. A comparator tree (RANKER) compares the start/stop pairs and stores the intermediate results in the g,f registers. When an output pair is calculted, the output multiplexor (OMUX) loads the result into the h,i register pair, to be then stored in the memory selected by MMUX.