Computer Architectures for Image Processing

Larry S. Davis
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

TR-145 May, 1980

This paper will be presented at the Asilomar Workshop on Picture Data Description and Management, August, 1980.

This research was supported in part by funds derived from the National Science Foundation under Grant ENG-7904037.

1. Introduction

The analysis of large, complex images is a computationally demanding task. The sheer bulk of data in even a monochromatic TV frame (more than 2^{20} bits) results in even the simplest image analysis operations (histogram construction, thresholding) requiring several seconds of processor time for moderately large conventional processors. For a variety of biomedical, industrial and military applications, this results in a totally unacceptable throughput rate.

One of the most promising approaches towards improving this throughput rate is the design and development of novel architectures possibly using multiple processors and memories, which enable many parts of an image to be processed in parallel or which operate at very high speeds.

There are at least three distinct types of such architectures which can profitably be distinguished; each is accompanied by its own specific set of advantages, disadvantages and theoretical and practical problems:

1) focal plane architectures, which are integrated into sensors at the focal plane and which are capable of operating at high-quality television data rates (7.5 MHz).

2) cellular arrays of simple, bit serial processing elements (PE's) (a special class of fixed-interconnection single-instruction stream-multiple data stream, SIMD, machines).

3) General single- or multiple-instruction stream-multiple data stream machines (S/MIMD) with many general-purpose processors, memories and a flexible interconnection network.

As one moves from architectures of type 1 through type 3 there is a significant decrease in speed. Focal plane architectures can compute a relatively complex computation (e.g., a 5x5 convolution) at the rate of 100 ns/pixel. A single logical operation on a cellular computer can take between 2-10μs, depending on the technology used to construct the PE's. Even ignoring the time required to load a subset of a 6-bit TV frame into a "large" cellular array of 64x64 PE's, and assuming that at least 1000 logical operations are required to compute an arbitrary 5x5 convolution, the cellular machine would take approximately .4 seconds to compute the convolution (5 us/operation x 1000 operations/ 64x64 subimage x 64 subimages/frame). The time per pixel would thus be $\frac{4 \times 10^{-1}}{2.5 \times 10^{5} \text{ pixels}}$ = 1.5μs/pixel, which is 5 times slower than the focal plane architecture. Finally, consider the more general S/MIMD architecture (which would operate as an SIMD machine to compute a convolution). If we ignore the time required to load the image into memories and the time lost to memory contention (which may be non-negligeable) and even assume that the processors operate at their maximum data transfer rate (about 5 us for a standard microprocessor), a collection of 256 microprocessors organized as an S/MIMD machine would probably require at least 1-2

seconds to compute the convolution, a factor of 5 times slower than the cellular machine.

Balancing this decrease in processing efficiency is an increase in processing generality. Focal plane architecture is functionally quite rigid. It cannot, e.g., be used to apply iterative algorithms to an image (such as shrink/expand) unless the number of iterations is known a priori. Even then, it requires duplication of circuitry (e.g., the median of median operation computed by a TI VLSI architecture [1]). The cellular arrays are more general, since their PE's are ordinarily capable of computing any Boolean function over a single bit plane of a point and a simple function of its four or eight neighbors. However, for non-logical operations, the PE's are very difficult to program due to their "low-level" instruction set. S/MIMD machines composed of many microprocessors are still more general, since not only are the microprocessors' machine instructions ordinarily quite powerful, but compilers are available for translating high-level languages (such as PASCAL or FORTRAN) into the machine language of the microprocessors.

The remainder of this paper explores these three classes of architectures in more detail, and suggests the class of image analysis operations for which each is most relevant.

2. Focal plane architectures.

Focal plane architectures integrate processors into sensors at the focal plane and are capable of operating at high quality television data rates (7.5 MHz.). There is no parallelism but the image is processed as quickly as it is obtained. This is especially relevant for applications which require analyzing sequences of images in real time such as tracking objects from frame to frame, or awaiting the onset of a critical event.

Hughes Research Laboratory [2] has designed and constructed charged-couple devices (CCD) and metal oxide semiconductor (MOS) processing architectures that can be integrated into TV cameras. For example, they have constructed a test chip which is capable of computing several simple image analysis operations and is designed to be integrated into the sensor at the focal plane. All of the operations are defined over 3x3 neighborhoods of 4 bit pixels and operate at 2 MHz. The operations include a simple edge detector, a low pass spatial filter and a Laplacian. The circuit utilizes 2 CCD delay lines when operated from a camera in order to store 3 lines of video (one can be received directly from the camera).

More recently [2] Hughes has been developing a more complex chip whose capabilities include a 26x26 element convolution process for computing the "Mexican hat" (Gaussian weighted Laplacian) operator which plays a central role in David Marr's theory of low-level vision (see, e.g., [3]). The Mexican hat

operator requires 8 bits of dynamic range, since the weights in the convolution have a range of 150:1. Hughes will capitalize on the circular symmetry of the Mexican hat to build only a 26x13 bi-polar convolution filter. The most ambitious design goal of this chip, however, is the development of programmable processing Kernals - i.e., a capability for dynamically changing the weights defining the convolution. At the present time, no information is available concerning the degree of success which Hughes has experienced with this chip.

The circuits which Hughes has constructed to implement convolutions involve charge-sensing and charge accumulation - i.e., the convolution is computed based on an analog representation of the image signal. Texas Instruments [1,4] has been investigating an alternative approach based on VLSI technologies.

In general, the convolution of a sequence $X = \{X_i\}_{i=0}^{n}$ with a sequence of weights $W = \{W_i\}_{i=0}^{n}$ is defined by

$$C(i) = \sum_{j=0}^{n} W_j X_{i+j} \qquad (1)$$

Now, if we express X_n as

$$X_n = \sum_{b=0}^{r} X_{n,b} * 2^b \qquad X_{n,b} \in \{0,1\} \qquad (2)$$

then (2) can be substituted into (1) to obtain

$$C(i) = \sum_{b=o}^{r} \left(\sum_{j=0}^{n} W_j X_{i+j,b} \right) 2^b$$

Thus, C(i) can be computed using a total of about rn shifts and adds. However, time can be saved by prestoring all values of $\sum_{j=0}^{n} W_j X_{i+j,b}$ in a 2^{n+1} by $B_w + \log_2(r+1)$ bit memory where B_w is the number of bits required to store the maximum W_j. Now, the computation of C(i) takes r+1 table look-ups in the memory, and r+1 shifts and adds. This technique is called the ROM-accumulator (RAC) technique.

Texas Instruments has constructed a breadboard version of the above convolution algorithm for 9 point convolutions. The breadboard consists of 9 input latches which buffer the input data paths into the RAC circuitry, and also serve as an 8 bit wide shift register for sliding window operations. Data in the input latches are clocked in parallel into parallel-in-serial-out shift registers which serve to form 8 sequential 9-bit addresses to the partial product memory. The partial product memory is composed of 12, 30 ns static 1K x 1 MOS RAMs. Finally, the 8 partial products are combined to form the convolution by shift and accumulate circuitry. TI estimates that for 3x3 sliding window convolutions, the maximum

input data rate will be 2.5 MHz. To achieve 7.5 MHz TV data rates, 3 boards must be operated in parallel. Each board would then compute a non-sliding convolution.

The advantage of TI's VLSI design versus Hughes' CCD design is that the dynamic range of the convolution weights can be increased with only a small increase in the size of the partial product memory, while there are practical limitations on the dynamic range of the Hughes' architecture. On the other hand, the VLSI approach is impractical for large convolutions such as the "Mexican Hat". Even a small 10 x 10 convolution would require a ROM which is 2^{100} x $(B_w + 2^{r+1})$ bits, which is clearly impossible. If one adopted the blocking schemes suggested in [4] (essentially break the large memory into several smaller memories and then combine the results with additional circuitry), then the architecture would become too slow.

3. Cellular architectures

Cellular logic arrays were the earliest special purpose architecture proposed for image processing (Unger [5]). Duff [6] outlines the defining features of such arrays:

1) The image is divided into a set of cells (pixels) and a unique processor is assigned to each cell.

2) Each processor cell has certain memory and logic capabilities.

3) There is a fixed interconnection structure which enables "adjacent" processors to communicate. "Adjacency" here usually corresponds to picture adjacency - i.e., 4-neighbors or 8-neighbors for a rectangular grid.

4) The logic function which each processor computes is set simultaneously and identically using control lines which run in parallel to each processor.

CLIP 3 [6] is an array of 16x12 processing elements which are constructed using TTL MSI circuits. The architecture of the cell is illustrated in Figure 1 (from Cordella, Duff and Levialdi [7]). Here f_d and f_n are circuits capable of computing any Boolean function of their inputs. The circuit in G is capable of counting how many neighbors, from a specified subset of neighbors (the subset is controlled by the control lines $G_1,\ldots,G_8$), have value 1 (CLIP is capable of operating on grey scale images, but operates on a single bit at a time), and comparing that count

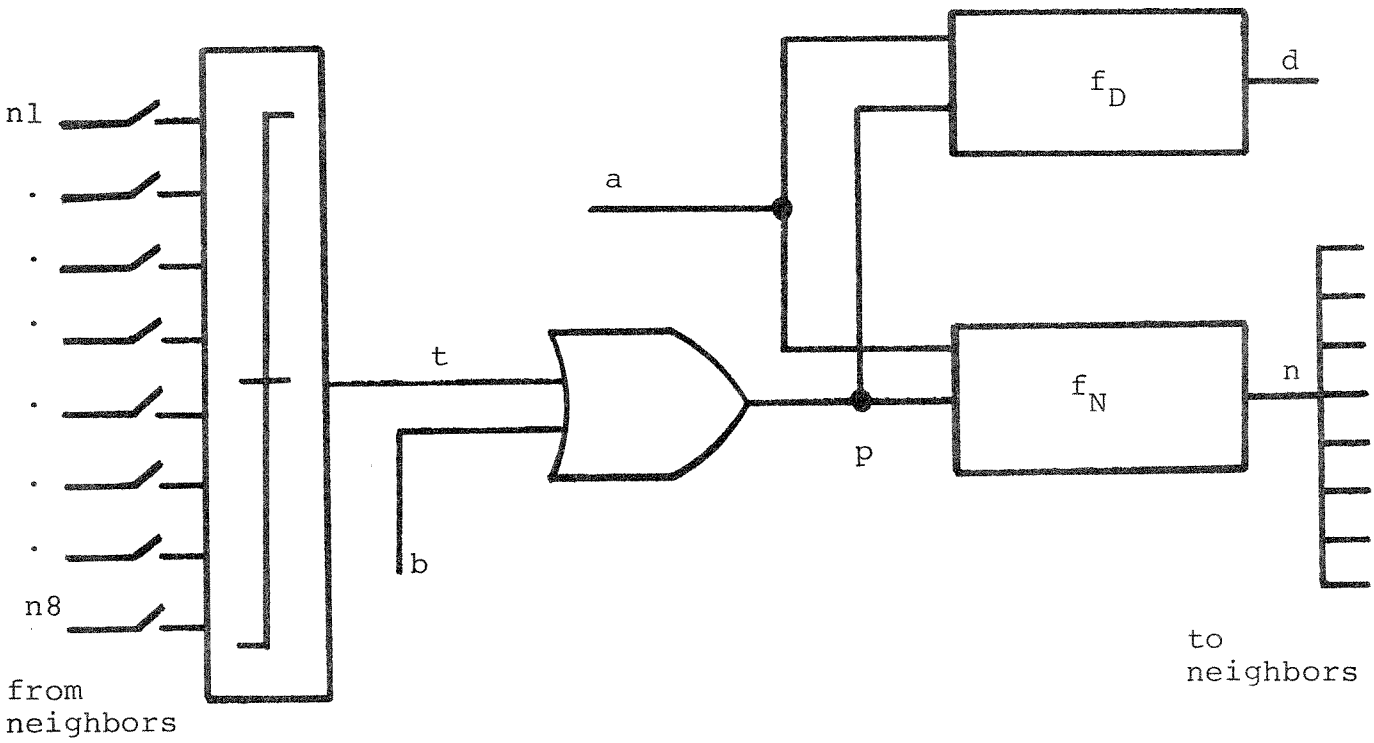

Figure 1. The CLIP cell (from [7]).

against a threshold. So, for example, G can determine if at least 2 out of 4 4-neighbors of a point have value 1. The values of "a" and "b" are extracted from the intersection of 2 bit planes with the processing element. The bit planes are chosen under program control. The outputs of the processing elements are n and d. The output n becomes an input to the G circuit of neighboring cells. The output d is stored in one of the bit planes at the processing element.

CLIP 4 is a larger cellular array (96x96) built using N-MOS LSI. Eight processors are incorporated in a 4x2 chip which is mounted on a 40 pin dual-in-line package. It is more easily programmed to perform arithmetic operations than CLIP 3 since it contains an extra buffer for automatic carry. Although CLIP 4 is 5 times slower than CLIP 3, it can process 50 times as many pixels per instruction as CLIP 3. Duff estimated that there would be time for 100-200 operations between successive television frames.

One of the difficulties with the CLIP machine, as well as other cellular processes, is that the very low level of the individual instructions makes it very cumbersome to write non-trivial image analysis programs. For example, the detailed analysis of the computational requirements of a histogram-based thresholding algorithm, described by Cordella, et al [7] show that the programmer of the CLIP array must have intimate knowledge of the logical organization of the PE's. It seems clear that if

such cellular arrays are to have widespread applications in image processing it will be necessary to construct a virtual machine over the array which describes picture operations at a higher level of abstraction, and provides a more powerful set of control structures than the conditional branches in the "machine language" of CLIP.

Several other cellular architectures have been designed and/or constructed. For example, Reeves [8] discusses the design of a Binary Array Processor (BAP) which is very similar to the CLIP machines.

One other cellular array which should be mentioned is GLOPR [9]. The GLOPR machine contains only a single PE and shift registers for piping image windows through the PE, but the machine is programmed as if it were truely a cellular array.

The GLOPR machine is based on a hexagonal interconnection structure between pixels and operates only on binary images. There are only 14 distinct immediate neighborhoods for a hexagonal array, independent of rotations. Each GLOPR instruction specifies a subset of these 14 patterns which are matched against the neighborhood of each pixel. Circular shift registers are employed to match patterns independent of orientation. The new value computed at each pixel is then a Boolean function of the original value and the result of the neighborhood matching.

GLOPR was subsequently integrated by Perkin-Elmer into

a commercial white blood cell analyzer called "diff3" (Preston, et al [10]). Diff3 was capable of operating on 8 pixels simultaneously. The circular-shift registers and associated neighborhood matching logic of the GLOPR design were replaced by a 64x14 ROM which was addressed by the 6-bit neighborhood of a pixel.

Preston, et al [10] also contains an extensive discussion of other cellular logic machines, as well as a discussion of the theory of cellular automata and a review of applications of cellular machines to image processing problems in biomedicine.

During the past few years a large number of image analysis algorithms have been developed which operate on a pyramid image representation [11,12]. The pyramid is a "stack" of images, where the image at level i is obtained (ordinarily) by averaging and sampling the image at level i-1. For our purposes, we will assume that the bottom or 0^{th} level of the pryamid is a 2^n x 2^n image and that the i^{th} level is a 2^{n-i} x 2^{n-i} image obtained by averaging non-overlapping 2x2 windows of the image at level i-1. A pixel in the level i image is defined to be adjacent to:

1) Its 4 or 8 neighbors at level i,

2) The 4 pixels at level i-1 which were averaged to form this pixel, and

3) The pixel at level i+1 which this pixel (along with three others) contributes to forming.

One of the principle advantages of the pyramid is that an image can first be analyzed at a coarse resolution, and the

results of that analysis can be used to selectively guide the application of subsequent operations to finer resolution versions of the image. For example, Kelley [13] detected the boundary of human faces in a coarse resolution image, and used the results of that computation to compute a finer description of the boundary at the shape which was finally used to predict the locations of facial features which would otherwise have been difficult to find. Tanimoto and Pavlidis [14] describe an edge detection algorithm which uses many levels of the pyramid rather than only two. Other image analysis operations for which pyramid-based algorithms have been proposed include image segmentation by functional approximation (Pavlidis [12], Blumenthal, Davis and Rosenfeld [15]), by texture analysis (Chen and Pavlidis [16]) and by thresholding both monochromatic images (Shneier [17]) and multispectral images (Ohlander, Price and Reddy [18] which is a generalization of Ohlander's [19] recursive segmentation procedure to pyramids). In addition, a large number of algorithms for operating on a pyramid-like representation of binary images, called a quad tree, have been developed (e.g., Samet [20], Dyer, et al [21], Davis and Roussopoulos [22]).

Although it is straightforward to define a theoretical generalization of cellular arrays to pyramids (see Dyer [23]), constructing an actual pyramid in which all nodes of all levels can be simultaneously active is not straightforward. In what follows we

briefly present a simple extension of cellular architectures which allow them to operate as pyramids; however, only one level of the pyramid is active during any instruction cycle. We should mention that it might be preferable to implement a pyramid using one of the more general architectures described in Section 4.

Consider an ordinarily cellular array i composed of n^2 PE's. Any PE is addressed by its location (i,j) in the array $0 \leq i, j \leq n-1$, $n-1 = 2^m$.

Such a cellular array can be modified to act as a pyramid in the following way: To perform an operation at the r^{th} level of the pyramid, the only PE's which are <u>active</u> are those located at positions whose addresses represented as binary numbers are $(X^{m-r}O^r, Y^{m-r}O^r)$ where X^s and Y^s are strings of 0's and 1's of length s, and O^r is a string of 0's of length r. So, for example, if the base of the pyramid is $2^3 \times 2^3$ (i.e., m=3), then all PE's are active at level 0, all PE's with even-numbered addresses are active at level 1, only PE's at locations (0,0), (0,4) (4,0) and (4,4) are active at level 2, and only PE (0,0) is active at level 3 (see Figure 2). The local memory requirements of a PE are linearly proportional to the number of levels in the pyramid to which it belongs. When operating the pyramid at level r, the machine would be r times slower than it would at level 0, since the "neighbors" of a level r PE are all located r+1 direct connections away.

	0	1	2	3	4	5	6	7
0	0-3	0	0-1	0	0-2	0	0-1	0
1	0	0	0	0	0	0	0	0
2	0-1	0	0-1	0	0-1	0	0-1	0
3	0	0	0	0	0	0	0	0
4	0-2	0	0-1	0	0-2	0	0-1	0
5	0	0	0	0	0	0	0	0
6	0-1	0	0-1	0	0-1	0	0-1	0
7	0	0	0	0	0	0	0	0

Figure 2 - PE (i,j) is active for the level numbers shown in position (i,j).

Furthermore, the instruction set at the array would have to be enhanced to include at least two new instructions:

1) A reduction instruction which loads the pyramid at level r based on the image data in the pyramid at level r-1, and

2) A projection instruction, which enables a node at level r to communicate data to its adjacent nodes at level r-1.

These instructions essentially specify protocals for passing data through some path of direct connections in the underlying cellular array. The array is thus being used to simulate other interconnection networks (see Siegl [24] for a relatively complete comparison of various interconnection networks).

4. General S/MIMD Architectures

The last class of architectures considered are S/MIMD architectures. These are designed based on a large set of relatively powerful microprocessors, a large set of memory modules and some flexible method for establishing processor-processor and processor-memory communication.

The utilization of microprocessors rather than simple processing elements such as those incorporated in CLIP facilitates writing programs which implement complex image analysis operations. However, these more general machines have a much more complex overall organization than a cellular array.

Siegl et al [25] have been designing a partitionable S/MIMD system, called PASM, at Purdue University for image processing and pattern recognition. Figure 3 (from [25]) contains a block diagram of the system. The heart of the system is the Parallel Computation Unit (PCU), which contains a set of N microprocessors, N memories and an interconnection network which provides for communications between processors and memories.

The interconnection network is a multi-stage implementation of the PM2I [26] network. At stage j of the network, processor i can be directly connected to any subset of processors i, $i+2^j$, $i-2^j$ (arithmetic modulo N). If $N = 2^n$, then the network is a n-stage network. The delay incurred in communications between processors j_1 and j_2 is linearly proportional to the number of

ones in the binary representation of $|j_1-j_2|$. This makes it especially well suited for image processing. For example, if the 2^n processors are logically organized as a $2^{n/2} \times 2^{n/2}$ array, then the 4 neighbors of a processor (i,j) are (i,j-1), (i,j+1), (i-1,j) and (i+1,j). In the linear ordering of processors, (i,j) is the $(i2^{n/2} + j)^{th}$ processor. Its horizontally adjacent neighbors are the $(i2^{n/2} + j-1)^{st}$ and the $(i2^{n/2} + j+1)^{st}$ processor, which can arc directly linked to $(i2^{n/2} +j)$. Its vertically adjacent neighbors are the $((i-1)2^{n/2} + j)^{th}$ and $((i+1)2^{n/2} + j)^{th}$ processors, which are also directly linked to $(i2^{n/2} + j)$ in the PM2I network. The diagonal neighbors of (i,j) are distance 2 away in the interconnection network (1 horizontal followed by 1 vertical movement). Siegl [25] describes applications of PASM to image analysis operations such as smoothing, histogram construction, and FFT computation.

It should be pointed out that PASM, with its PM2I interconnection network, could easily support the pyramid representation discussed in Section 3. Again, suppose that the 2^n processors are logically organized as a $2^{n/2} \times 2^{n/2}$ array. At the 1^{th} level of the pyramid, the horizontal neighbors of a point (i,j) are at location $(i,j-2^r)$, $(i,j+2^r)$ which are directly connected by the

r^{th} stage of the network. The vertical neighbors are at $(i-2^r,j)$ and $(i+2^r,j)$. These correspond to addresses $n_d=(i-2^r)2^{n/2} + j$, $n_u=(i+2^r)2^{n/2} + j$. Since the address of (i,j) is $a = i2^{n/2} + j$, we see that, e.g. $|n_d-a| = i2^{n/2} - 2^{r+n/2} + j - i2^{n/2} - j = 2^{r+n/2}$. So, vertical neighbors at the r^{th} level of the pyramid are directly connected at the $(r+n/2)^{nd}$ stage of the PM2I network.

Finally, the MASK instruction of PASM which selectively enables/disables a subset of the processors can be used to signal which level of the pyramid is currently active. The complexity of many pyramid-based image analysis algorithms (such as [14-19]) suggest that PASM would be a more appropriate architecture than a cellular array of simple PE's.

Rieger [27] discusses the design of a ring network architecture for image processing and distributed problem-solving research called ZMOB. ZMOB is a collection of 256 identical Z80A microprocessors which communicate using a high-speed, synchronous "conveyor belt".

Each microprocessor is a mail stop on the conveyor belt, which can be designed to make one complete "revolution" every 5.25μ seconds; since this is the maximum data transfer rate of the Z80A, the conveyor belt can logically support n/2 non-blocking, direct processor-processor communications for a mob of n processors. The messages on the conveyor belt are formatted to include sender

address, recipient address, one byte of message and various control bits. Each processor has its own place on the conveyor belt in which it places its messages, although it can accept a message from any bin on the conveyor belt.

The image analysis algorithms which motivated the design of ZMOB were relaxation processes [28,29]. A cellular architecture based on simple PE's cannot support such computations since each iteration of a relaxation process might require up to 1000 multiplies/pixel. One complete iteration of such an algorithm on a 512x512 image would therefore require about 250,000,000 multiplies, and a comparable number of adds. Rieger [27] reports that, based on hand-simulated timing results, ZMOB would require about 100 seconds/iteration.

5. Conclusions

We have distinguished three types of architecture for image processing: focal plane architectures, cellular arrays of simple processing elements and more general networks of microprocessors configured as S/MIMD machines. Focal plane architectures are well suited for position-invariant local operations, cellular arrays for iterative local operations on binary and grey scale images and S/MIMD machines for more complex image analysis operations such as relaxation. We have also briefly discussed how pyramid machines may be constructed as either cellular arrays or more flexibly connected networks. The wide-spread application of pyramid-based algorithms to image analysis problems indicates that interest should increase in the design and construction of pyramid machines.

REFERENCES

1. W. Eversole, D. Mayer, F. Frazec and T. Cheek Jr., "Investigation of VLSI technologies for image processing", Proc. Image Understanding Workshop, Palo Alto, CA., April 1979, pp. 159-163.

2. G. Nudd, S. Fouse, T. Nussmeier, P. Nygaard, "Development of custom-designed integrated circuits for image understanding", Proc. Image Understanding Workshop, Los Angeles, CA., Nov. 1979, pp. 1-9.

3. D. Marr and E. Hildreth, "The theory of edge detection", Proc. R. Soc. Lond. B, to appear.

4. W. Eversole, D. Mayer, F. Frazec, and T. Check Jr., "Investigation of VLSI technologies for image processing", Proc. Image Understanding Workshop, Los Angeles, CA., Nov. 1979, pp. 10-14.

5. S. Unger, "A computed oriented towards spatial problems", Proc. IRE, 46, 1958.

6. M. Duff, "CLIP 4: A large scale integrated circuit array parallel processor", Proc. 3rd Int. Joint Conf. on Pattern Recognition, Coronado, CA., 1976, pp. 728-733.

7. L. Cordella, M. Duff and S. Levialdi, "Thresholding: A challenge for parallel processing", Comp. Graphics and Image Processing, 3, 1977, pp. 207-220.

8. A. Reeves, "A systematically designed binary array processor", IEEET-C, 29, 1980, pp. 278-287.

9. K. Preston Jr. and P. Norgren, "Interactive image processor speeds pattern recognition", Electronics, 45, 1972, p. 89.

10. K. Preston Jr., M. Duff, S. Levialdi, P. Norgren and J. Toriwaki, "Basics of cellular logic with some applications in medical image processing", Proc. IEEE, 67, 1979, pp. 826-856.

11. A. Klinger and C. Dyer, "Experiments in picture representation with regular decomposition", Comp. Graphics and Image Processing, 5, 1976, pp. 68-105.

12. T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, New York, 1978.

13. M. Kelly, "Edge detection in pictures by computer using planning", Machine Intelligence, 6, 1971, pp. 379-409.

14. S. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing", Comp. Graphics and Image Processing, 4, 1975, pp. 104-119.

15. A. Blumenthal, L. Davis and A. Rosenfeld, "Detecting natural 'plateaus' in one-dimensional patterns", IEEET-Computers, 26, pp. 178-179.

16. P. Chen and T. Pavlidis, "Segmentation by texture using a cooccurrence matrix and a split-and-merge algorithm," Proc. 4th Int. Joint Conf. on Pattern Recognition, Kyoto, JAPAN, 1978, pp. 565-569.

17. M. Shneier, "Using pyramids to define local thresholds for blob detection", Proc. Image Understanding Workshop, Los Angeles, CA., Nov. 1979, pp. 31-35.

18. R. Ohlander, K. Price, R. Reddy, "Picture segmentation using a recursive region splitting method", Comp. Graphics and Image Processing, 7, 1978, pp. 313-333.

19. R. Ohlander, "Analysis of natural scenes", Carnegie-Mellon Univ. Comp. Sci. Ph.D. Thesis, 1975.

20. H. Samet and A. Rosenfeld, "Quadtree structures for region processing", Proc. Image Understanding Conference, Los Angeles, CA., Nov. 1979, pp. 36-41.

21. C. Dyer, A. Rosenfeld and H. Samet, "Region representation: boundary codes from quad trees," Comm. ACM, March 1980,

22. L. Davis and N. Roussopoulos, "Approximate pattern matching in a pattern database", to appear in Information Systems, 1980.

23. C. Dyer, "Augmented cellular automata for image analysis", Ph.D. Thesis, Computer Science, Univ. of MD., 1979.

24. H. Siegl, "A model of SIMD machines and a comparison of various interconnection networks", IEEET-Computers, 28, 1979, pp. 907-917.

25. H. Siegl, L. Siegl, R. McMillen, P. Mueller Jr., S. Smith, "An SIMD/MIMD multimicroprocessor system for image processing and pattern recognition", Proc. Pattern Recognition and Image Processing Conference, Chicago, IL., 1979, pp. 214-224.

26. Y. T. Feng, "Data Manipulating functions in parallel processors and their implementations", IEEET-Computers, 23, 1874, pp. 309-318.

27. C. Rieger, "ZMOB: A mob of 256 cooperative Z80A-based microprocessors", in Proc. Image Understanding Conference, Los Angeles, CA, 1979, pp. 25-30.

28. A. Rosenfeld, R. Hummel and S. Zucker, "Scene labelling by relaxation operations", IEEET-Systems, Man and Cybernetics, 6, 1976, pp. 420-433.

29. L. Davis and A. Rosenfeld, "Cooperating processes for low-level vision: A survey", Univ. of Texas Computer Sciences TR-123, Jan. 1980.