Parallel Image Processing

CHARALAMBOS D. STAMOPOULOS

Abstract—**Parallel image processing shows certain general similarities to retinal processing and shares with optical computing the characteristics of parallelism. Simulation on digital serial computers (DSC's), although possible in the simpler cases, very soon becomes almost uncontrollable, requiring large memories and very long programming and execution times. Advances in present circuit technology now permit one to construct large arrays of interconnected parallel logic elements resulting in real-time parallel machines where the action of a statement is simultaneous on all the points of the array. A typical processor of this kind is briefly described and an introduction to its code is given. Examples of symmetrical and directional functions and parallel algorithms are also presented. Possible applications of such processors in image processing, pattern recognition, and artificial intelligence are briefly discussed.**

Index Terms—**Artificial intelligence (AI), cellular logic, cybernetics, image processing, optical computing, parallel digital computing, parallel processing, pattern recognition.**

I. INTRODUCTION

NATURAL optical systems, as it is shown by various physiological studies of retinas and associated nervous networks, are inherently hybrid parallel systems with a mixture of optical and neural signal processing [10], [11], [13].

Many propositions have been made for cellular arrays showing at least some of the properties of the natural optical systems. However, most studies of proposed parallel logic systems for image processing, pattern recognition, and artificial intelligence (AI) have been theoretical, often employing a digital serial computer (DSC) for simulation of the parallel algorithms involved [7]–[9], [24]. This simulation, although possible in the simpler cases, very soon becomes almost uncontrollable, involving very long and tedious programming, huge memories, and very long execution times which, inspite the increasing refinements of the equipment used, do not always compare favorably with the inherently very slow nervous system. With rare exceptions [15], [6], it seems that hardware construction of digital parallel image processors has been neglected.

On the other hand, optical computers, which are two- and three-dimensional parallel processors, permitted some of the most remarkable results in optical image deblurring, coherent side-looking synthetic-aperture radar, correlative pattern recognition, and others [22], [14], [23]. Their unmatched efficiency has been correctly attributed to the parallelism of the processing.

Manuscript received June 12, 1974; revised November 4, 1974.
The author is with the Institute of Biomedical Sciences, University of São Paulo, São Paulo, Brazil.

Digital parallel processors (DPP's) show many advantages in image processing, pattern recognition, and AI. These advantages surpass by far their limitations which are rapidly being overcome and as DPP's are programmable they may substitute the DSC in hybrid optical-digital systems because their processing speed is much near to the speed of optical systems. Thus DPP's are relevant and closely related to natural optical systems and optical computing.

II. SIMULATION ON DSC's

A common argument against actual construction of parallel processors is that, in principle, one can simulate everything on a general purpose computer, already available, by making the necessary software effort. The DSC, as the name computer implies, has been developed in an effort to make, rapidly, precise calculations under control of a stored program. In this area of applications the DSC is quite useful. This is because the machine reflects very well the predominantly serial character of calculations, i.e., the machine is well adapted to the problem. It can also simulate other processes, although less efficiently. A serial computer simulating a parallel process requires, among others, many additional instructions and memory positions to store these instructions and the intermediate results. If one is willing to pay for the programming effort and additional memory and is satisfied with the long execution times involved, then the DSC can eventually simulate an inherently parallel process. The same, however, may be true for a parallel processor; under certain circumstances it can also make serial calculations, etc., although not as efficiently as the DSC [20].

The conclusion to this is that the term "general purpose machine" should be used with caution. For each area of applications, there is always a more appropriate hardware which cannot be simply substituted for by software efforts. In this respect [22] is strongly recommended; an interesting comparison is made of optical computing versus DSC methods for image processing involving some of the fastest computers in the world. Numbers comparing execution times, memories necessary, and other details are given.

III. PARALLEL PROCESSING

Advances in circuit technology now allow one to economically build relatively large arrays of interconnected parallel logic elements using integrated circuits. This makes it possible to match the form of the processor to the form of the data and to process plane data sets by using

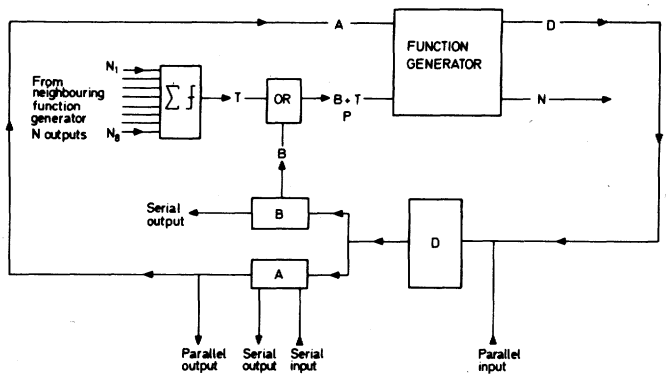

Fig. 1. The principle of the cellular logic image processor.

two-dimensional arrays of logic cells. Such arrays are well suited for image processing, pattern recognition, and AI either independently or in combination with optical methods.

DPP's, using cellular logic, are real-time machines where the action of a statement is simultaneous on all the points of the array. The interconnections between cells may vary, each cell propagating to and receiving information from its neighbors. These interconnections may be limited to the immediate neighbors, or they may involve the edge of the array or greater depths of the neighborhood. The edge may be used as an interface between various actual or virtual arrays. The action may be symmetrical or directional and the tessellations of various types, the most common being the square and the hexagonal ones.

Their actual limitation is the relatively small number of cells in the array. However, this limitation can be eliminated with large-scale integrated circuits and arrays of 20 000 cells or more are feasible. Their advantages include speed of processing, smaller memories in comparison to those of DSC, and simplicity of programming. Another interesting characteristic is that these processors are well adapted for the execution of parallel nonmathematical algorithms, using only logic, not less efficient or respectful than the mathematical ones. The last two items are very important if we consider the potential applications of these processors in areas of science where the users are not specialists in computer science or mathematics.

IV. THE CELLULAR LOGIC IMAGE PROCESSOR

An interesting example of this kind of machine is the parallel cellular logic image processor, CLIP 3, constructed by the Image Processing Group, Department of Physics and Astronomy, University College London [1]–[3]. This processor, using TTL logic and MOS memory, has a $16 \times 12 = 192$ logic cell array and permits square (S) and hexagonal (H) tessellations under program control. An idea of this processor is given in Fig. 1. Each cell includes a summation and threshold unit Σf, an OR gate, and a function generator with two inputs, P and A, and two independent outputs, N and D. The D outputs of the array may be stored in parallel in one of the 16 available locations of the D memory and/or displayed in the A or B displays via the corresponding A and B shift registers. A double beam oscilloscope is used for this purpose. For economy reasons, serial scanning and displaying devices are used. However, if necessary, they may be substituted by parallel devices. A common method of pattern input or change is via the A display by means of a light pen. The A register is parallel connected to the A inputs so that the image displayed in A is also the input image. One of the common uses of the B register is to display the output. In the S mode, the N output of each cell carries information to the 8 immediate neighbors of that cell. Thus each cell may send information to its neighbors and receive information from them ($N_1 - N_8$ inputs). In the H mode, the immediate neighbors used are restricted to 6.

The information propagated by the neighbors selected is summed and thresholded by Σf and the result, T, is ORed with the contents of the B register. The OR output $B + T$, is the P input to the function generator. The outputs N and D are Boolean functions of the two inputs P and A. The functions are set by 8 control lines, not shown in the diagram, and are valid for every cell in the array. Three control lines, associated with the threshold unit, select one of 8 threshold levels and 8 control lines select the neighbor inputs which are summed and thresholded. This enables both symmetrical and directional functions to be used. An additional control line sets the spare interconnections of the edge cells at either 0 or 1 ($E = 0$ or $E = 1$). In many cases it is convenient to consider the edge as a frame of imaginary cells, outside the 192 cell array, which may emit either 0 or 1. The propagation directions provided in the S and H modes are indicated in Fig. 2 together with the interconnections from one cell to its neighbors for hexagonal tessellation. Although the number of possible transformations in CLIP 3 is very large ($\cong 10^{308}$) [2], [4], only a relatively small part of them results in distinct useful operations. Instructions to the processor are contained in a random-access memory of 256 words of 24 bits each. Note that the array is iterative; propagation of information is allowed to proceed freely until a stable state is reached. Only then the D output is transferred to the D memory. Thus all parts of the input image contribute to all parts of the out-

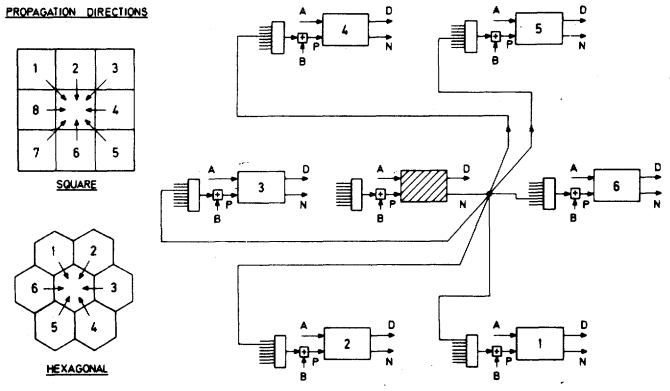

Fig. 2. Propagation directions and interconnections to neighboring cells.

put image. The parallel processing, at every iteration, proceeds at the maximum speed available in the integrated circuit used. This processor (the total cost of which is of the order of $10 000) is very simple to use and to program and is very fast. The propagation delay within one cell is approximately 30 ns while the propagation delay to neighboring cells in negligible. Thus functions that set all the cells to 0 or 1 or invert an image take approximately 30 ns. Functions where the output D depends only on the A input of the cell and its immediate neighbors need approximately 60 ns. Propagating functions, for a 192 cell array, may present a maximum propagation time of 3.12 μs but normally do not exceed 1 μs [5], [6]. These results have been verified by actual measurements. The approximate maximum propagation time t, for an array of L by M cells, is given by [5]

$$t = \tfrac{1}{2}L(M + 1)d$$

where $L > M$ and both L and M are even, d being the delay between and within consecutive cells. Using this relation one can find that for a $160 \times 120 = 19\,200$ cell array, with the same integrated circuits ($d = 30$ ns), the maximum propagation time would be 290.4 μs.

V. SOFTWARE

In order to better understand the examples presented in the next sections a few words on the software are necessary. For a detailed description the appropriate references are recommended [2], [6]. Programming is based on a mnemonic code, the D memory with 16 positions of 192 bits (for image data), two registers of 192 bits (A and B), and 256 instruction positions of 24 bits each (instruction memory). Statements are set on 24 toggle switches and are practically of three kind.

The first is a LOAD statement that essentially includes the addresses in the D memory from which the A and B registers are loaded, the eventual display or clearing of A and B, the memory address to which the results of the following process instructions are directed and the mode.

The second is a PROCESS statement that essentially includes the threshold used, the neighbors involved, the kind of information sent to and received from the neighbors, the resulting D output, the mode and the edge condition. Eventually, it includes logical operations with the results of previous process statements. The threshold indicates a minimum, out of a number of selected inputs, that must be present at Σf to give an output at T. The obvious values are 1,2,$\cdots$,8. In the code, however, these values are represented by 0,1,$\cdots$,7 because there are only 3 bits available for threshold purposes; to find the actual threshold value, add 1 to the value indicated in the code.

The third is a BRANCH statement, unconditional or conditional, subdivided into branch, branch to subroutine, and branch from subroutine statements. The author shares the opinion of those who used this processor that the code is very simple to use and to directly translate into its binary equivalent; it takes probably less time to set the instruction switches than to type the corresponding statement.

VI. EXAMPLES OF SYMMETRICAL FUNCTIONS

To illustrate the effect of simple symmetrical functions in parallel image processing some pictures taken on CLIP 3 are presented in Fig. 3 together with the statements used and the "naive" description of the results, where possible. The input (A) is shown in the lower part and the output (B) in the upper part of each picture. The absence of S implies H mode and the absence of E implies $E = 0$. For selected inputs and related propagation directions refer to Fig. 2. The detailed explanation of each statement follows.

Fig. 3(a)—$D = 0(1\text{–}6)A$, PA; Removal of Black Noise:

1) H mode, $E = 0$.
2) Threshold value $= 0 + 1 = 1$.
3) Selected inputs to Σf: 1 to 6. Hence it is a symmetrical function.
4) N logic: A. Cells give out a 1 to their immediate neighbors if they are black (A input is 1).
5) D logic: PA. Cells give out a 1 to their D output if they are black *and* receive a 1 from one or more immediate neighbors. Thus, the single black cells present in the input are eliminated from the output; they are the only black cells which do not receive a 1 from any immediate neighbor.

Fig. 3(b)—$D = 0(1\text{–}6)A$, $P + A$; Expand Function:

1) to 4) As in Fig. 3(a).
5) D logic: $P + A$. Cells give out a 1 to their D output if they are black *or* receive a 1 from one or more immediate neighbors.

Fig. 3(c)—$D = 0(1\text{–}6)P\bar{A}$, PA E; Contour of Objects:

1) H mode, $E = 1$.
2) to 3) As in Fig. 3(a).
4) N logic: $P\bar{A}$. Cells give out a 1 to their immediate neighbors if they are white ($\bar{A} = 1$) *and* receive a 1 from one or more immediate neighbors. In this case all edge cells receive a 1 ($E = 1$). Those of them which are white after 30 ns emit also a 1 to their neighbors and the process is repeated until black cells are reached and all edge-connected white cells emit a 1. Thus the edge initiates a

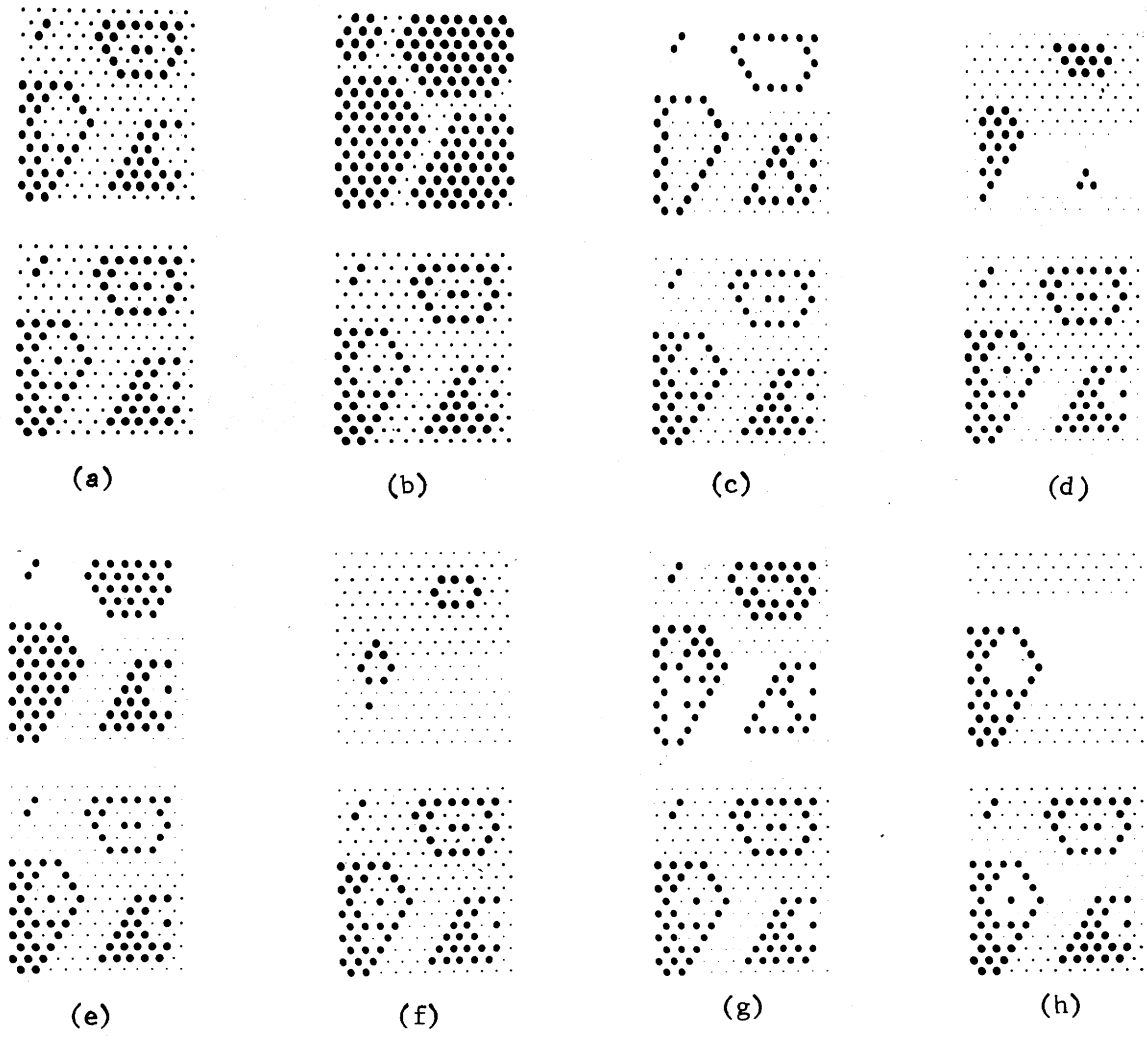

Fig. 3. Examples of symmetrical functions. (a) $D = (1\text{–}6)A,PA$; removal of black noise. (b) $D = 0(1\text{–}6)A,P + A$; expand function. (c) $D = 0(1\text{–}6)P\bar{A},PA \quad E$; contour of objects. (d) $D = 0(1\text{–}6)P\bar{A},\bar{P} \quad E$; cells inside contour. (e) $D = 0(1\text{–}6)P\bar{A},\bar{P} + A \quad E$; filling function. (f) $D = 0(1\text{–}6)P\bar{A},\bar{P}\bar{A} \quad E$; white cells inside contour. (g) $D = 0(1\text{–}6)P\bar{A},\bar{P}\bar{A} + PA \quad E$; contour plus white cells inside. (h) $D = 0(1\text{–}6)PA,PA \quad E$; black cells contacting the edge. For detailed explanation see the text.

propagation of 1's through edge-connected white cells.

5) D logic: PA. Cells give out a 1 to their D output if they are black *and* receive a 1 from one or more edge-connected immediate neighbors.

Fig. 3(d)—$D = 0(1\text{–}6)P\bar{A}, \bar{P}\ E$; Cells Inside Contour:

1) to 4) As in Fig. 3(c).

5) D logic: $\bar{P}$. Cells give out a 1 to their D output if they do not receive a 1 from any immediate neighbors.

Fig. 3(e)—$D = 0(1\text{–}6)P\bar{A}, \bar{P} + A\ E$; Filling Function:

1) to 4) As in Fig. 3(c).

5) D logic: $\bar{P} + A$. Cells give out a 1 to their D output if they do not receive a 1 from any immediate neighbors *or* are black.

Fig. 3(f)—$D = 0(1\text{–}6)P\bar{A},\bar{P}\bar{A}$; White Cells Inside Contour:

1) to 4) As in Fig. 3(c).

5) D logic: $\bar{P}\bar{A}$. Cells give out a 1 to their D output if they do not receive a 1 from any immediate neighbors *and* are white.

Fig. 3(g)—$D = 0(1\text{–}6)P\bar{A}, \bar{P}\bar{A} + PA\ E$; Contour Plus White Cells Inside:

1) to 4) As in Fig. 3(c).

5) D logic: $\bar{P}\bar{A} + PA$. Cells give out a 1 to their D output if being white they do not receive a 1 from any immediate neighbors *or* being black they receive a 1 from one or more edge-connected immediate neighbors.

Fig. 3(h)—$D = 0(1\text{–}6)PA, PA\ E$; Black Cells Edge Connected:

1) to 3) As in Fig. 3(c).

4) N logic: PA. Cells give out a 1 to their immediate neighbors if they are black *and* receive a 1 from one or more immediate neighbors. In this case all edge cells receive a 1. Those of them which are

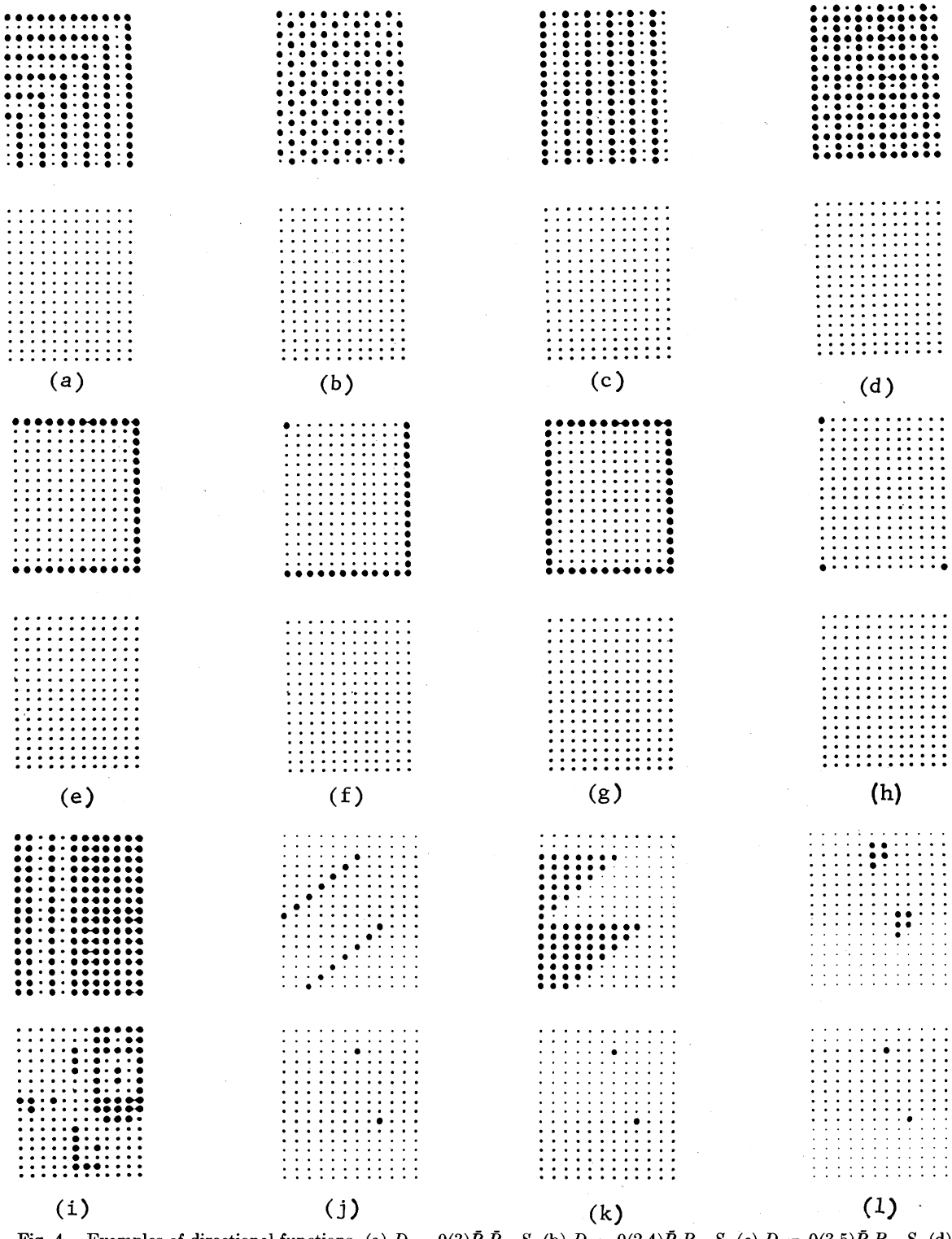

Fig. 4. Examples of directional functions. (a) $D = 0(3)\bar{P},\bar{P}$ S. (b) $D = 0(2,4)\bar{P},P$ S. (c) $D = 0(3,5)\bar{P},P$ S. (d) $D = 0(2\text{–}4)\bar{P},P$ S. (e) $D = 1(3,5)1,\bar{P}$ S. (f) $D = 1(3,5,7)1,\bar{P}$ S. (g) $D = 2(1,3,5,7)1,\bar{P}$ S. (h) $D = 2(2,4,6\text{–}8)1,\bar{P}$ S. (i) $D = 0(2,6)P + A,P$ S; Detection of empty vertical lines. (j) $D = 0(3)P\bar{A},\bar{P} + A$ SE; propagation of lines in direction 3). (k) $D = 1(3,4)P\bar{A}$ $\bar{P} + A$ SE; propagation of black points between directions 3) and 4). (l) $D = 3(3\text{–}6)\bar{A},\bar{P} + A$ SE; production of points in given directions. For detailed explanation see the text.

black, after 30 ns, emit also a 1 to their neighbors and the process is repeated until white cells are reached and all edge-connected black cells emit a 1. Thus the edge initiates a propagation of 1's through edge-connected black cells.

5) D logic: PA. Cells give out a 1 to their D output if they are black *and* receive a 1 from one or more edge-connected immediate neighbors.

VII. EXAMPLES OF DIRECTIONAL FUNCTIONS

Directional functions offer interesting possibilities increasing the flexibility and purpose of parallel image processing. A few examples are presented in Fig. 4 under the same conditions given in Section VI. The detailed explanation of each statement follows.

Fig. 4(a)—$D = 0(3)\bar{P}, \bar{P}$ S:

1) S mode, $E = 0$.
2) Threshold value $= 0 + 1 = 1$.
3) Selected inputs to Σf: direction 3).
4) N logic: $\bar{P}$. Cells give out a 0 to their immediate neighbors in direction 3) if they receive a 1 from (one or more of) their immediate neighbors in direction 3). Otherwise they give out a 1.
5) D logic: P. Cells give out a 0 to their D output if they receive a 1 from (one or more of) their immediate neighbors in direction 3). Otherwise they give out a 1.

Fig. 4(b)—$D = 0(2,4)\bar{P}, P$ S:

1) to 4) As in Fig. 4(a) but for directions 2) and 4).
5) D logic: P. Cells give out a 1 to their D output if they receive a 1 from one or more of their immediate neighbors in directions 2) and 4).

Fig. 4(c)—$D = 0(3,5)\bar{P}, P$ S:

1) to 5) As in Fig. 4(b) but for directions 3) and 5).

Fig. 4(d)—$D = 0(2\text{–}4)\bar{P}, P$ S:

1) to 5) As in Fig. 4(b) but for directions 2)–4).

Fig. 4(e)—$D = 1(3,5)1, \bar{P}$ S:

1) S mode, $E = 0$.
2) Threshold value $= 1 + 1 = 2$.
3) Selected inputs to Σf: directions 3) and 5).
4) N logic: 1. Cells give out a 1 to their immediate neighbors in directions 3) and 5).
5) D logic: $\bar{P}$. Cells give out a 0 to their D output if they receive a 1 from two (or more) of their immediate neighbors in directions 3) and 5). Otherwise they give out a 1.

Fig. 4(f)—$D = 1(3,5,7)1, \bar{P}$ S:

1) to 5) As in Fig. 4(e) but for directions 3), 5), and 7).

Fig. 4(g)—$D = 2(1,3,5,7)1, \bar{P}$ S:

1) S mode, $E = 0$.
2) Threshold value $= 2 + 1 = 3$.
3) to 4) As in Fig. 4(e) but for directions 1), 3), 5), and 7).
5) D logic: $\bar{P}$. Cells give out a 0 to their D output if they receive a 1 from three or more of their immediate neighbors in directions 1), 3), 5), and 7). Otherwise they give out a 1.

Fig. 4(h)—$D = 2(2,4,6\text{–}8)1, \bar{P}$ S:

1) to 5) As in Fig. 4(g) but for directions 2), 4), 6)–8).

Fig. 4(i)—$D = 0(2,6)P + A, P$ S; *Detection of Empty Vertical Lines:*

1) S mode, $E = 0$.
2) Threshold value $= 0 + 1 = 1$.
3) Selected inputs to Σf: directions 2) and 6).
4) N logic: $P + A$. Cells give out a 1 to their immediate neighbors in directions 2) and 6) if they are black *or* if they receive a 1 from one or more immediate neighbors in directions 2) and 6). In this case there is a propagation of 1's along those vertical lines which have one or more black cells. The propagation is initiated by any black point. Vertical lines without black cells do not propagate 1's.
5) D logic: P. Cells give out a 1 to their D output if they receive a 1 from one or more immediate neighbors in directions 2) and 6).

Fig. 4(j)—$D = 0(3)P\bar{A}, \bar{P} + A$ SE; *Propagation of Lines in Direction* 3):

1) S mode, $E = 1$.
2) Threshold value $= 0 + 1 = 1$.
3) Selected inputs to Σf: direction 3).
4) N logic: $P\bar{A}$. Cells give out a 1 to their immediate neighbors in direction 3) if they are white *and* receive a 1 from their immediate neighbors in direction 3). A propagation of 1's along direction 3) is initiated by $E = 1$. Finally all edge-connected white cells give out a 1 in direction 3).
5) D logic: $\bar{P} + A$. Cells give out a 1 to their D output if they are black *or* receive a 0 from their neighbors in direction 3).

Note that the same result may be obtained by $D = 0(3)P + A, P + A$ S.

1) S mode, $E = 0$.
2) to 3) As above.
4) N logic: $P + A$. Cells give out a 1 to their immediate neighbors in direction 3) if they are black *or* if they receive a 1 from their immediate neighbors in direction 3). A propagation of 1's is initiated at the black points.
5) D logic: $P + A$. Cells give out a 1 to their D output if they are black *or* if they receive a 1 from their neighbors in direction 3).

Fig. 4(k)—$D = 1(3,4)P\bar{A}, \bar{P} + A$ SE; *Propagation of Black Points between Directions* 3) *and* 4):

1) S mode, $E = 1$.
2) Threshold value $= 1 + 1 = 2$.
3) Selected inputs to Σf: directions 3) and 4).
4) N logic: $P\bar{A}$. Cells give out a 1 to their immediate neighbors in directions 3) and 4) if they are white *and* receive a 1 from their two immediate neighbors in directions 3) and 4). Finally all edge-connected white cells, satisfying the above condition, give out a 1.
5) D logic: $\bar{P} + A$. Cells give out a 1 to their D output if they are black *or* if they receive a 0 from any of their immediate neighbors in directions 3) and 4).

Fig. 4(l)—$D = 3(3\text{–}6)\bar{A}, \bar{P} + A$ SE; *Production of Points in Given Directions:*

1) S mode, $E = 1$.
2) Threshold value $= 3 + 1 = 4$.

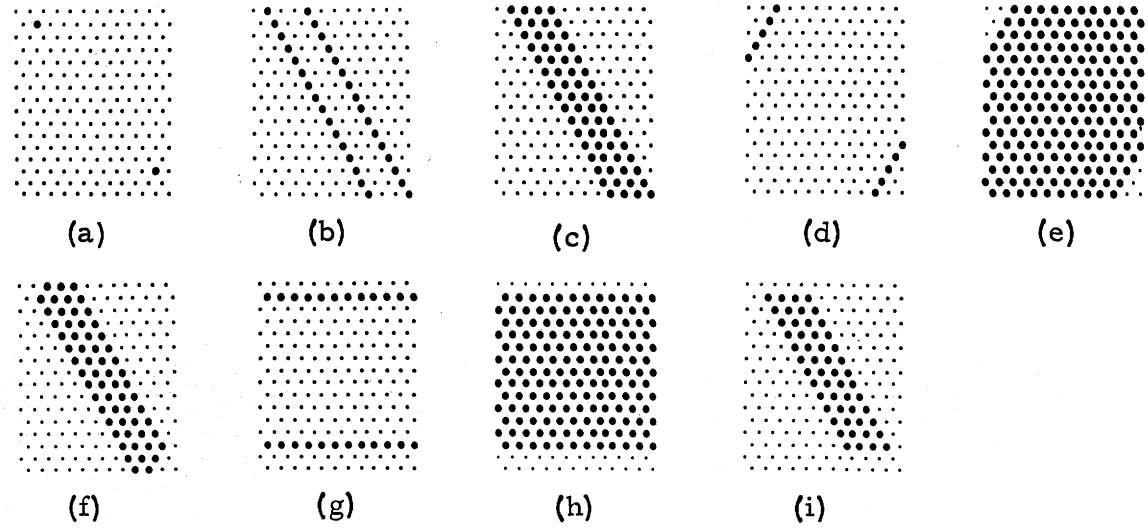

Fig. 5. The principle of the ANDing part of the algorithm for joining two points. (a) Input. (b) Propagation of lines along direction 1). (c) Filling with black points. (d) Propagation of lines along direction 2). (e) Filling with black points. (f) ANDing of (c) and (e). (g) Propagation of lines along direction 3). (h) Filling with black points. (i) ANDing of (f) and (h) results in minimal core.

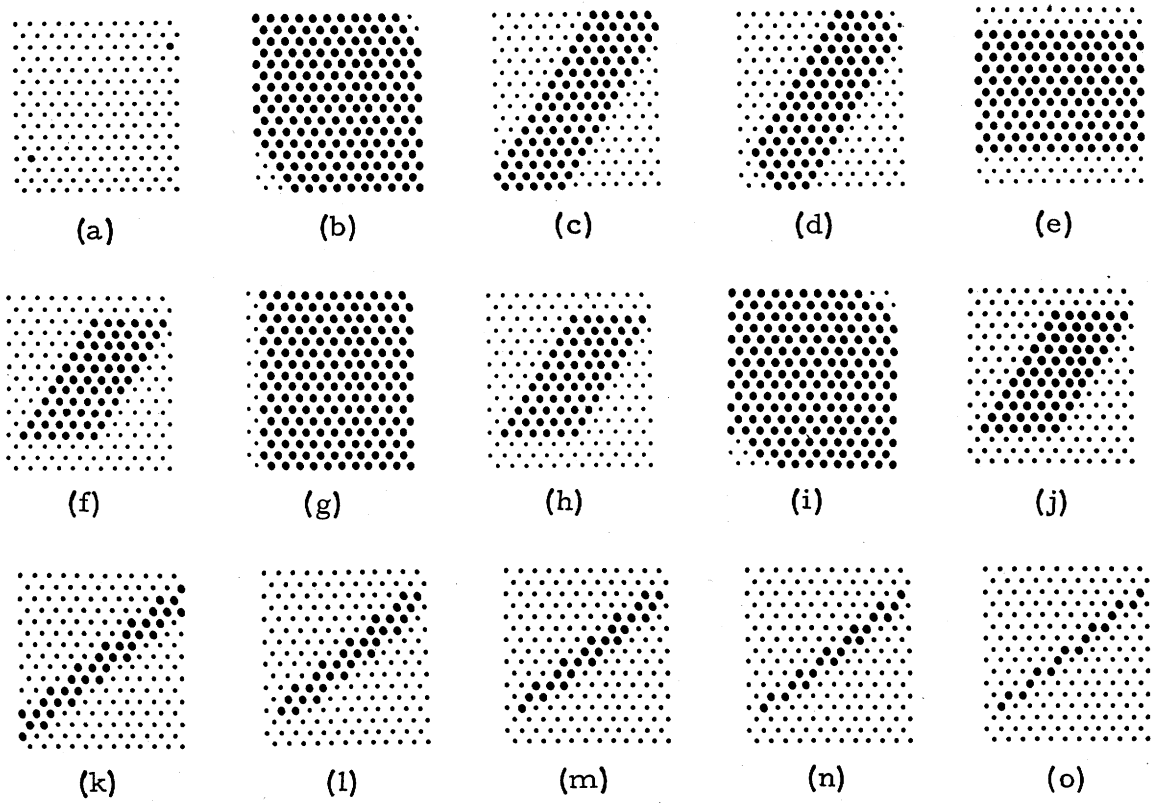

Fig. 6. A complete sequence of the algorithm for joining two points with AND and thinning sections. To obtain a smaller core, before thinning, the vertical and 45° directions are introduced by an instant switching to S mode (under program control). (a) Input. (b) Propagation of lines and filling along direction 1). (c) Propagation and filling along direction 2). (d) ANDing of (b) and (c). (e) Propagation of lines and filling along direction 3). (f) ANDing of (d) and (e). (g) Propagation and filling along the vertical direction. (h) ANDing of (f) and (g). (i) Propagation and filling along the 135° direction. (j) ANDing of (h) and (i). (k) Propagation and filling along the 45° direction. (l) ANDing of (j) and (k) resulting to small minimal core. (m) and (n) Thinning section. (o) Result.

3) Selected inputs to Σf: directions 3)–6).
4) N logic: $\bar{A}$. Cells give out a 1 to their immediate neighbors in directions 3)–6) if they are white.
5) D logic: $\bar{P} + A$. Cells give out a 1 to their D output if they are black *or* receive a 0 from any of their immediate neighbors in directions 3)–6).

VIII. EXAMPLES OF PARALLEL ALGORITHMS

Parallel nonmathematical algorithms are suitable for cellular logic processing. Two examples are presented [18], [19], [21].

1) Given two points it is required to join them by a straight-line segment. Note that nothing is said about

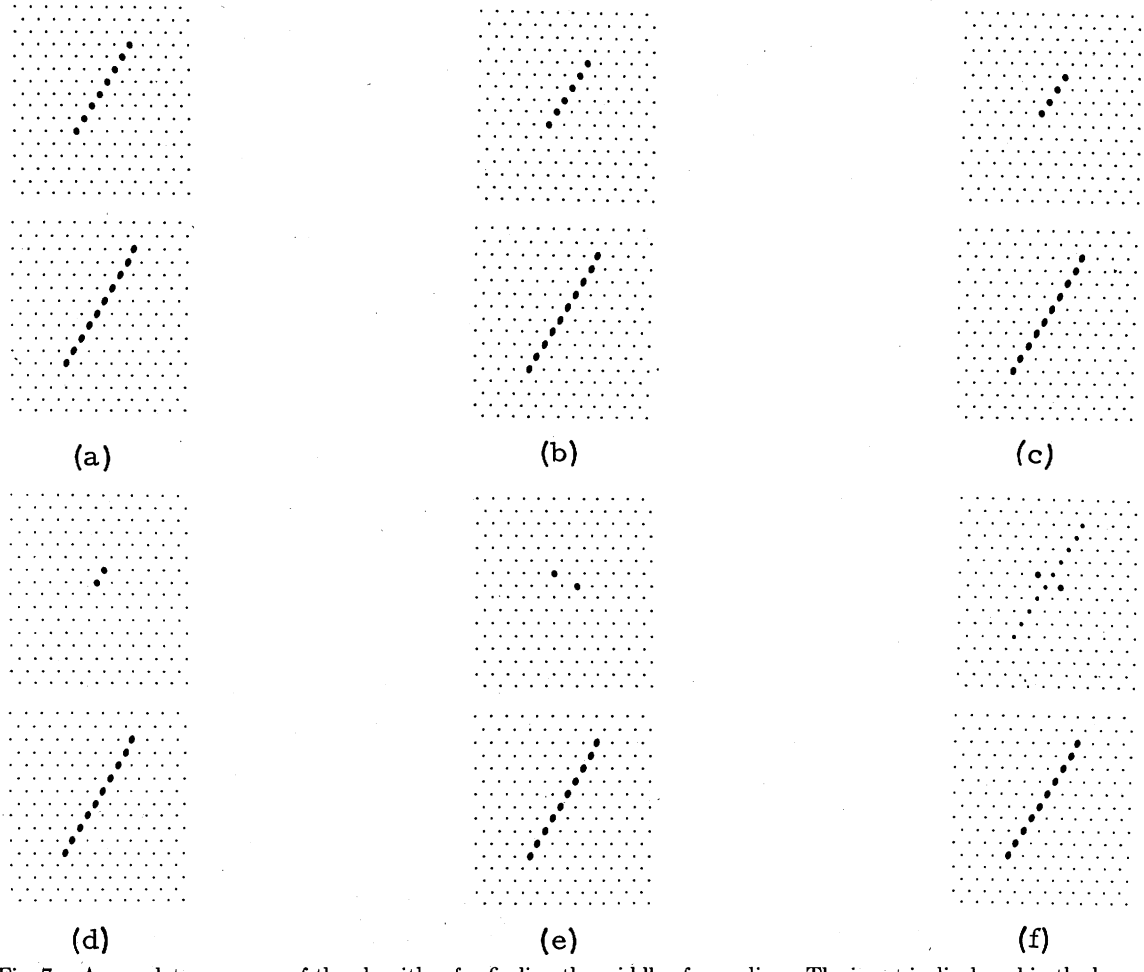

Fig. 7. A complete sequence of the algorithm for finding the middle of open lines. The input is displayed in the lower part in all pictures for reference purposes. (a)–(d) Successive detection and elimination of the two endpoints. (e) Elimination of the last two points and production of markers. (f) Display of markers (strong) indicating the middle together with the line (faint).

their coordinates, the directions or quadrants involved, solution of equations, etc. Such an algorithm should work properly over the whole area of the array and should not destroy the connectivity [16]. The principle of the algorithm consists of two main sections: the first, the ANDing section, may be described as follows. Successively and along half of the available directions of the array, straight lines passing from the two points are propagated, the area between the lines is filled with black points and the results are ANDed, thus obtaining a minimal symmetrical core. The sequence is shown in Fig. 5. The second section consists of a rotational thinning subroutine which eliminates all those points which satisfy the thinning condition—black points with 3 white given neighbors and 3 black given neighbors. The remaining black points form the required join which is actually the best possible compromise of a straight-line segment for the array given. A more complete case, including vertical and 45° directions, is shown in Fig. 6. Note that with increasing number of available directions the minimal core becomes less and less, and with a sufficiently large number of available directions the thinning process would become unnecessary.

2) Given any open line or lines it is required to find the middle of them. Note that nothing is said about their number, length, direction, measurement, division, etc. The principle of the algorithm may be described as follows. In a continuous repetitive process the endpoints of the lines are detected and eliminated. In the case of lines with an odd number of points the last remaining point is the wanted middle. In the case of lines with an even number of points, before the elimination of the last two points, markers indicating the middle of the line are generated. Both, the lines and their middles, are displayed in the output, the former in an intermittent way and the latter continuously. This results in an output with a faint line and a more pronounced middle point. A complete sequence is shown in the photographs of Fig. 7. Some additional results are shown in Fig. 8.

IX. POSSIBLE DEVELOPMENTS AND APPLICATIONS

There are various ways of exploring the possibilities of parallel processors. One is by the construction of new kinds of cellular logic, taking advantage of the experience we have gained from natural systems. Nature has given several solutions for each problem and the surviving solutions are in general very efficient, although the components used are not so efficient. Another approach is to refine

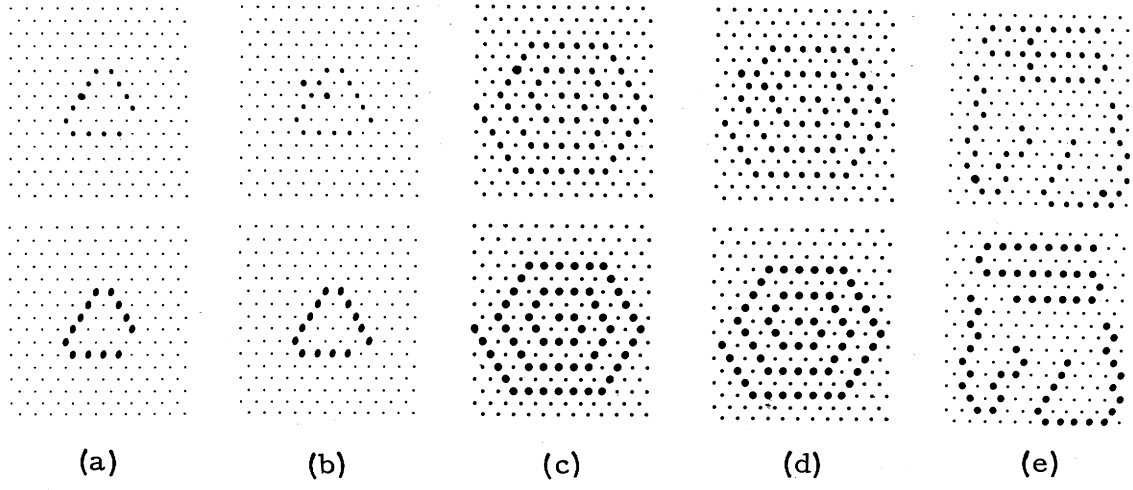

Fig. 8. Various examples of detection of the middle of open lines. The input is displayed in the lower part and the resulting output in the upper part of each picture. (a) Line with odd number of points. (b) Line with even number of points. (c) Line with odd number of points. (d) Line with even number of points. (e) The algorithm acting on various open lines at the same time.

and expand the promising cases, by increasing the resolution, i.e., the number of array cells, to values comparable, as far as possible, to those of natural systems, introducing grey scale and color sensitivity, adaptability to light conditions, etc. However, with increasing number of cells the heat dissipation problem will become more severe. This problem can be eased if one uses micropower integrated circuits sacrificing and exchanging speed with less heat production. Natural systems are excellent guides in this respect. DPP's are the natural companions of purely optical systems. They are complementary and compatible with them and both are parallel and very fast devices. A possible combination of them into hybrid optical/DPP will result in programmable processors of increased flexibility and capability.

Possible scientific, technological, artistic, and military applications of real-time DPP, alone or in hybrid combinations, are many and promising in all areas of image processing, pattern recognition, and AI. To mention just a few of them: real-time detection of nucleated and nonnucleated cells, open cells, concavities, protuberances, chromosome recognition, form detection, feature extraction of patterns and pictures, classification of objects according to their area, perimeter, projections or shape, real-time detection and separation of targets, production of artistic patterns for static and moving pictures and others.

X. CONCLUSION

The construction of DPP's seems to be today quite feasible due to the new circuit techniques which permit large scale fabrication of integrated circuits at low prices. As the construction expenses become of the same order with the expenses of simulation on DSC's, the argument against such construction, because of economic reasons, is not valid. An additional advantage is the acquisition of know-how related to the construction of such processors which will be reflected in further progress. The number of real-time applications of parallel processing is only limited by the imagination.

ACKNOWLEDGMENT

The author wishes to thank Drs. M. J. B. Duff and D. M. Watson, Image Processing Group, Department of Physics and Astronomy, University College London, for having made available their facilities and CLIP 3.

REFERENCES

[1] "CLIP 3 technical specification (1)," Image Processing Group, Dep. Phys. Astron., Univ. College, London, England, Rep. 73/1.
[2] "CLIP 3 operating manual," Image Processing Group, Dep. Phys. Astron., Univ. College, London, England, Rep. 73/4.
[3] "CLIP 3 technical specification (2)," Image Processing Group, Dep. Phys. Astron., Univ. College London, England, Rep. 73/5.
[4] M. J. B. Duff, "Cellular logic and its significance to pattern recognition," presented at the Agard Avionics Panel 21st Tech. Symp. Artificial Intelligence, Rome, Italy, May 24–28, 1971.
[5] M. J. B. Duff, D. M. Watson, T. J. Fountain, and G. K. Shaw, "A cellular logic array for image processing," *Pattern Recognition*, vol. 5, pp. 229–247, 1973.
[6] M. J. B. Duff and D. M. Watson, "CLIP 3—A cellular logic image processor," presented at the NATO Advanced Study Inst. Conf. New Concepts and Technologies in Parallel Information Processing, Capri, Italy, June 17–30, 1973; also in *NATO ASI Series*. Groningen, The Netherlands: Noordhoff, 1973.
[7] M. J. E. Golay, "Hexagonal parallel pattern transformations," *IEEE Trans. Comput.*, vol. C-18, pp. 733–740, Aug. 1969.
[8] S. B. Gray, "Local properties of binary images in two dimensions," *IEEE Trans. Comput.*, vol. C-20, pp. 551–561, May 1971.
[9] J. K. Hawkins and C. J. Munsey, "A parallel computer organization and mechanizations," *IEEE Trans. Electron. Comput.*, vol. EC-12, pp. 251–262, June 1963.
[10] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, Jan. 1962; also *IEEE Trans. Military Electron.*, vol. MIL-7, pp. 98–103, Apr. 1963.
[11] D. H. Hubel, "The visual cortex of the brain," *Sci. Amer.*, Nov. 1963.
[12] "Image processing in cellular arrays," Image Processing Group, Dep. Phys. Astron., Univ. College, London, England, Rep. 72/1.
[13] K. Kirschfeld, "Das neurale Superpositionsauge," *Fortschritte der Zoologie*, vol. B21, H2/3, pp. 229–257, 1973.
[14] W. E. Kock, "Holography can help radar find new performance horizons," *Electronics*, vol. 43, pp. 80–88, Oct. 12, 1970.
[15] B. H. McCormick, "The Illinois pattern recognition com-

puter—ILIAC III," *IEEE Trans. Electron. Comput.*, vol. EC-12, pp. 791–813, Dec. 1963.
[16] A. Rosenfeld, "Connectivity in digital pictures," *J. Ass. Comput. Mach.*, vol. 17, pp. 146–160, 1970.
[17] C. D. Stamopoulos, M. J. B. Duff, and D. M. Watson, "Some aspects of the logic functions of CLIP 3," presented at the NATO Advanced Study Inst. Conf. New Concepts and Technologies in Parallel Information Processing, Capri, Italy, June 17–30, 1973; also in *NATO ASI Series*. Groningen, The Netherlands: Noordhoff, 1973.
[18] C. D. Stamopoulos, "Parallel algorithms for two points join," Inst. Comp. Sci. Doc., ICSI 506, Sept. 1973.
[19] ——, "A parallel algorithm for finding the middle of open lines," Inst. Comp. Sci. Doc., ICSI 510, Sept. 1973.
[20] ——, "Cellular logic image processing," presented at the 1st Int. Optical Computing Conf., Zürich, Switzerland, Apr. 9–11, 1974.
[21] ——, "Parallel algorithms for joining two points by a straight-line segment," *IEEE Trans. Comput.*, vol. C-23, pp. 642–646, June 1974.
[22] G. W. Stroke, "Optical computing," *IEEE Spectrum*, vol. 9, pp. 24–41, Dec. 1972.
[23] A. Vander Lugt, "Signal detection by complex spatial filtering," *IEEE Trans. Inform. Theory*, vol. IT-10, pp. 139–145, Apr. 1964.
[24] S. H. Unger, "Pattern detection and recognition," *Proc. IRE*, vol. 47, pp. 1737–1752, Oct. 1959.

Charalambos D. Stamopoulos was born in 1918 in Athens, Greece. He received the Diploma in physics from the University of Athens, Athens, Greece in 1940.

After passing the Grad. Brit. IRE examination he became in 1956 an Electronic Engineer and in 1967 a Chartered Engineer registered in London. Since 1954 he occupied various teaching posts at the Faculty of Medicine of the University of São Paulo, São Paulo, Brazil, and is now a Fellow Professor at the Department of Physiology, Institute of Biomedical Sciences, University of São Paulo. During 1972 and 1973 he spent 16 months as a Visiting Professor at the Institute of Computer Science of the University of London. Most of the courses given by him, including Electronics for Biology, Logic Circuits and Computer Science, are at postgraduate level. He has done research on electronic instrumentation, logic systems, and image processing and has published over 40 papers. His actual interests are concentrated on image processing, pattern recognition, artificial intelligence, and computer assisted instruction, especially as related to biomedical applications.

Prof. Stamopoulos is a member of the Institution of Electronic and Radio Engineers, London.