HETERARCHICAL ARCHITECTURES

FOR PARALLEL PROCESSING

OF DIGITAL IMAGES

Adolfo Guzmán

Informe Técnico AHR-79-3, Laboratorio AHR

Informe Técnico PR-79-23, Laboratorio PR

Presented at the II Seminario Internacional sobre el Uso de los Sensores Remotos, México, D.F. 1979.

Computer Science Dept., IIMAS
National University of Mexico
Apdo. 20-726. México 20, D.F.
MEXICO

ARTICULO 38

Abstract

A novel computer architecture is proposed, suitable for high speed digital processing of LANDSAT and other images. The machine comprises a collection (20 to 100) of small active units (perhaps microprocessors), each of them performing a task on a subimage. The coordination among them is handled by a blackboard or distributor. This is a memory that contains addresses of tasks ready to be done, as well as destinations of the results of finished computations. The name "heterarchy" is used because there is no hierarchy among the processors (active units).

The machine exhibits some form of graceful degradation, and it is incrementally expandible.

The main components are: an image memory, where the image to be processed resides; the active units, that perform the work; the distributor or blackboard, a hardware that contains pointers to work yet to be done; and a bus that connects the machine to a general purpose computer, making it appear as a peripheral unit.

The machine is compared with a general purpose heterarchical machine.

The machine, if built, will enhance the data handling capabilities of the Proyect P.R. at IIMAS-UNAM.

Key words: parallel computing; picture processing; heterarchical; graceful degradation; incrementally expandible; microprocessors.

Acknowledgments

The machine herein described was inspired by the computer architecture being designed at the AHR Laboratory. Fruitful comments were made by most of its members, specially Angel Kuri, Luis Lyons and David Rosenblueth.

The application of this machine to data compression was strongly influenced by Rosa Seco.

Work herein reported was partially supported by Grant # 1632 from the Consejo Nacional de Ciencia y Tecnología (México).

CONTENTS

INTRODUCTION

The analysis by digital computer of multispectral images [2] and other kinds of image processing is continuously increasing in magnitude [21] and complexity [23]. Thus, parallel processing of the images is a viable alternative to more traditional serial computation, specially where the influence of context [14] does not extend beyond a few pixels, thereby making possible to process more or less independently and simultaneously several pats of an image --several subimages--.

On the other hand, computer architectures suitable for general parallel processing (general purpose parallel machines) are being actively investigated, specially because the aggregation of many microprocessors to form a larger machine (Project AHR [13]) offers reductions in cost and improvements in reliability.

This article describes the design of a machine that processes images in parallel, by dividing the task into smaller subtasks, each of which may be carried out by a single functional unit (usually, a general purpose microprocessor). The division of the task is both in time and space: a functional unit may (generally) carry on several subtasks of the whole task on several parts of the image. The units are not functionally specialized: any one of them may carry on any function, and during normal execution, would perform several distinct subtasks. They are not geographically rooted; any unit may process several parts of the image. Nevertheless, if necessary, by adequate programming, it is possible to specialize or root the units.

Disadvantages of actual systems

Several systems imported to Mexico are large machines that require huge memory sizes (IBM's Herman or Erips system) and expensive general

purpose machines; other systems (Detenal's Varian machine) come more or less as a black or gray box, and are difficult to learn. Some others (G.E. Image 100) can be purchased with hardware features (hardware cube classifiers; see Avilés' thesis [1]) that perform very few image operations.

The main disadvantages of conventional general purpose machines that process images are that they are slow (since usually one processor is used) and expensive (it requires a lot of memory). In addition, if the software system is imported, the user may depend technologically on the vendor, in a field where progress is fast and obsolescence occurs soon. This dependence is usually lessened if the user acquires the software locally developed [18].

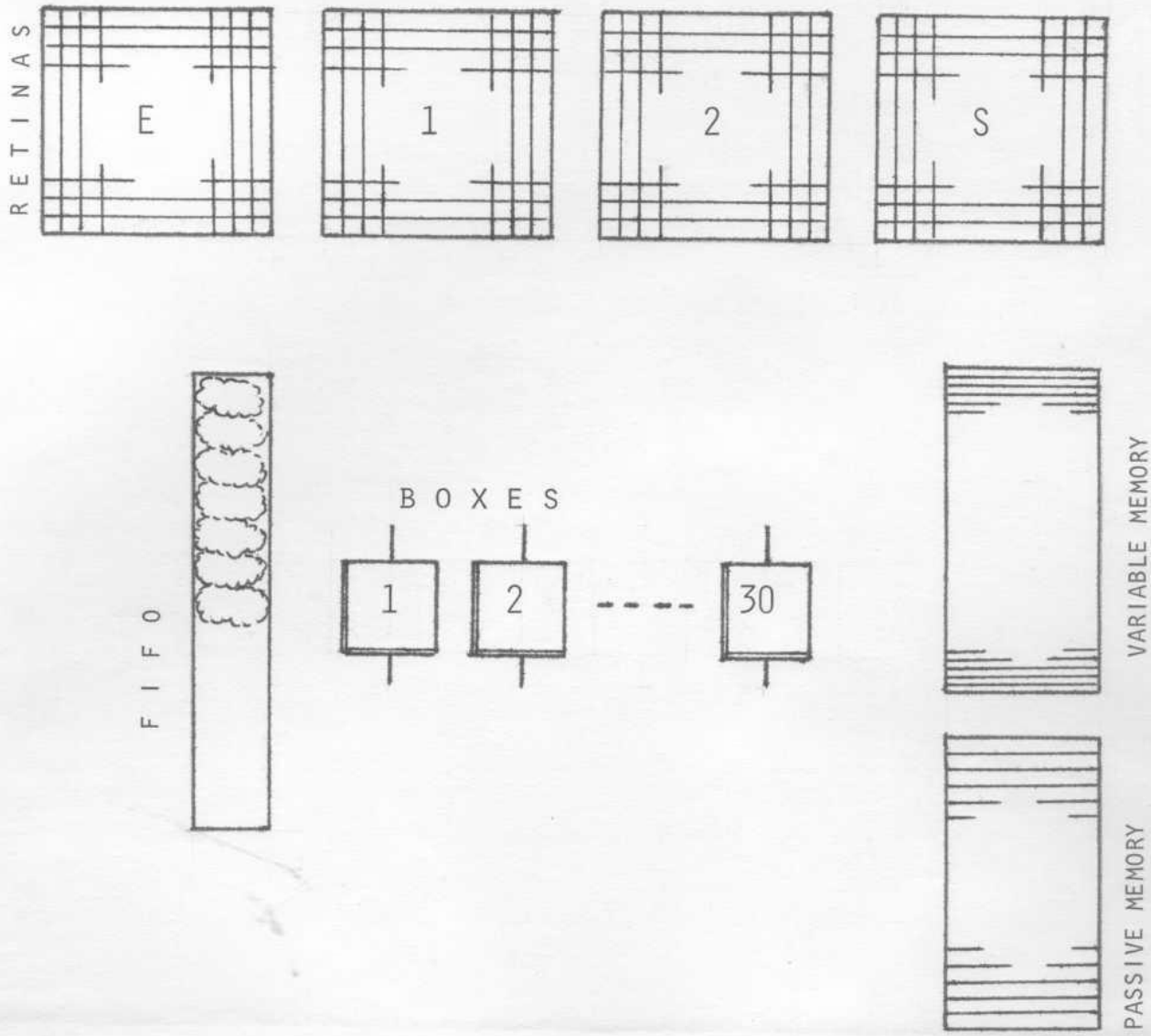

FIGURE 'THE PARTS OF THE MACHINE'

The boxes are functional units (microprocessors) that transform the input image stored in retina E into the output image stored in retina S, with the aid of intermediate retinas 1, 2, ...

Work proceeds in parallel, and it is buffered in the fifo, while the passive memory and the variable memory contain partial results and values obtained during execution of the programs.

ANATOMY OF THE MACHINE: ITS PARTS

Overview

The image to be analyzed is stored in a retina E of suitable dimensions (say, 256 by 256 pixels, each of 4 bytes). Partially processed images are held in other retinas 1, 2, 3, ... of similar or smaller size. The "output image" appears at the retina S. Partial data and results (scalar values of variables and lists) are stored in the passive memory and the variable memory, while the fifo holds the queue of jobs pending to be done. The boxes are the functional (or active) units that transform the input image into the output image (to be stored in retina S), using the other retinas as intermediate storage for images. See figure 'The parts of the machine'.

The complete machine interacts with its users through a general purpose computer (a minicomputer, perhaps), of which it becomes a slave or peripheral unit, capable of receiving unprocessed images and yielding processed results. The results may be images themselves, scalars (numbers) or lists. See figure 'The machine as a slave'. The machine shares its retinas with the minicomputer's main memory, thus allowing high speed data transfers between both machines.

The retinas

The retinas are large memories where the input, intermediate and output images reside. They can be thought of as bidimensional matrices (see figure 'The parts of the machine'), each having one or more bands. They form the storage media for images displayed in raster scan display systems (such as Comtal). Following the design of Luis Lyons' image display system developed at the Instituto de Ingeniería [20] (and a more recent X-ray digital display being built in Monterrey, N.L. [26]), the

retinas belong to the address space of the minicomputer that commands the image processor.

Typical sizes: retina E: 256 x 256 pixels, each of 4 bytes.
retina 1: 128 x 128 pixels, each of 8 bits.
retina 2: 64 x 64 pixels, each of 2 bits.
retina S: same as retina 2.

A given retina is never used by the same box as an input retina and also as an output retina; i.e., if a box "looks" into a retina i (and obtains information from it), then the results are not placed back in retina i, but in retina j (j>i) instead (if the result is an image), or in the variable and passive memories (if the result is a valid data object).

The fifo

The fifo (first in, first out) is a linear memory where work ready to be done waits for an idle box to take care of it. Each unit of work inside the fifo appears as a node; that is, in a particular format containing the name (number) of the solicited operation or task, a pointer to a window in the retina where the image data resides, and a pointer to the "environment" in the variable memory where values of variables are held.

Size of the fifo: 1024 words (nodes) of 6 bytes each.
Composition of the node: Number of the operation: 1 byte
Pointer to retina: 2.5 bytes (two 1-byte coordinates for beginning of window; 4 bits for number of retina).
Pointer to variable memory: 2.5 bytes.

Variable memory

The values that different variables attain in the course of the evaluation of a program are stored in a tree-structure, akin to the A-list

of LISP [13]. This tree resides in the variable memory. [24] gives details.

It is important to notice that, contrary to what happens in a monoprocessor, in our machine a variable can simultaneously have several values. For instance the "average grey level" of a part of an image may be at the same time 17, 87, 66 and 49 (because, in fact, four parts are being analyzed). The main purpose of the variable memory is to keep appropriate track of these environments (an environment is a mapping from variable names to data objects, that assigns to each variable a unique value). The memory itself is not "active" because the boxes update it. They also consult in this memory the value of a variable, when needed.

This memory is organized like an A-list or dictionary [16], to store lists.

The nodes stored in the fifo point to branches of this environment tree. Each branch represents an environment where values for variables may be found.

The values are atoms (including numbers) or lists. As in the case of the picture language "L" [23], no images or subimages can be stored as values.

Typical size: 32 k words.

Passive memory

This memory contains lists (although the use of lists by the image machine may be scanty) and code (although it is desirable that all needed code fits into the private memory of each microprocessor).

Typical size: 0 (zero).

In other fields [13], the passive memory has a substantial use.

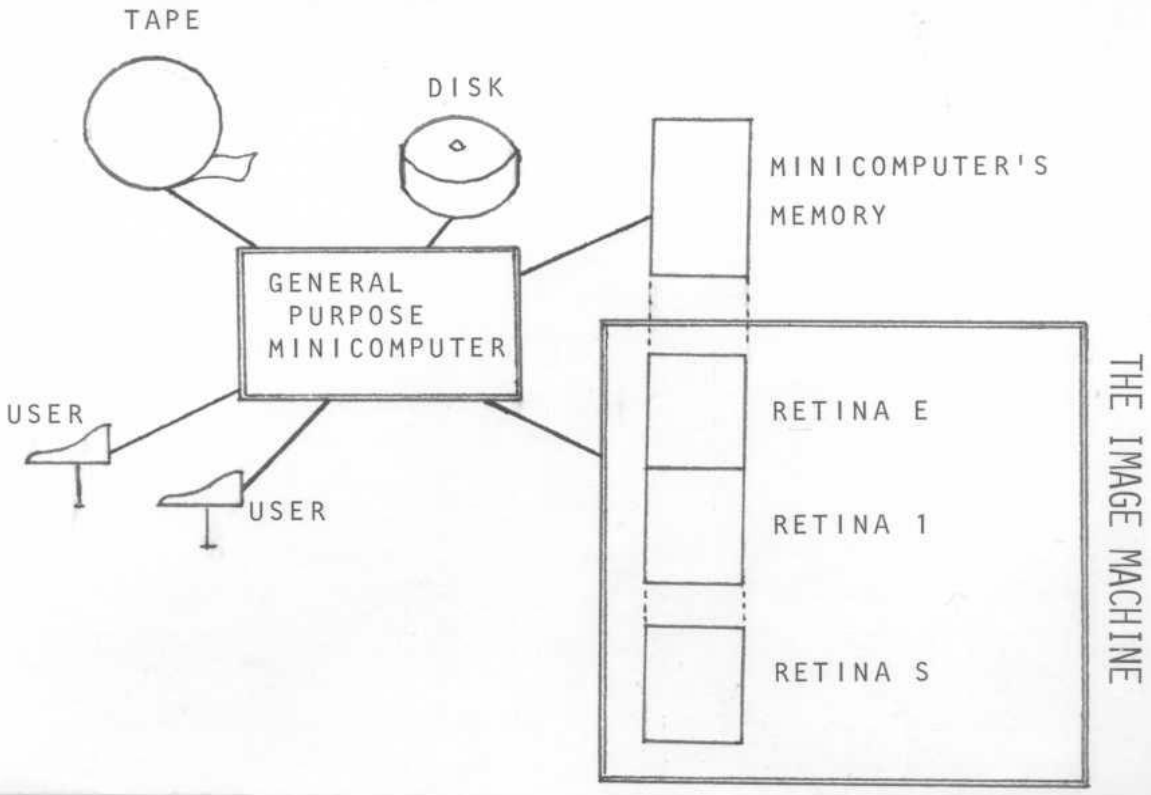

FIGURE 'THE MACHINE AS A SLAVE'

A normal general purpose computer is used to feed data to the retina E of the image machine, and to read back the results from retina S.

Thus, the image machine is a memory-to-memory processor (like the CDC 6600 and CYBER cpu's), while the normal computer serves the users, helps in the preparation of programs and handles the mass storage media.

The use of a mini or microcomputer with large address space (Interdata, Z-8000, Intel 8086, PDP-VAX) is recommended, because it will be able to access the retinas of the image machine as part of its own address space.

The boxes

The active elements of the image machine are units called boxes. Think of microprocessors. They contain the programs that perform the different transformations on the images. Each box posesses read only memory, where programs reside, as well as some writable random access (ram) memory, used for intermediate results. Since the machine described is experimental, it is advantageous to use ram memory to store the programs, too.

The programs stored in each box are of the form

$$P_i \rightarrow S_j$$

where P_i is a pattern to be located in retina i; S_j is a skeleton [12] to be placed in retina j (j>i) if P_i is found. Notice that, since j>i, each program "moves the image from the input to the output retina", while processing it. The meaning of this production rule is: if you (the box) find pattern P in retina i, replace in retina j the expression (bit pattern) or skeleton S. A program is typically formed by several of these production rules. More about this kind of programming using pattern matching can be found in [12, 15, 16].

The boxes are not wildly searching the retinas trying to match pattern P. Instead, they are guided (by the nodes from the fifo) to the places where work is. In addition, there are boxes that do look for certain patterns in the retinas.

The functioning of the machine is described in the next section.

It is not necessary for all boxes to be identical; the machine may have specialized units. But in the current design, all boxes look alike.

It is not necessary that the boxes are formed by microprocessors; special purpose hardware could also be used.

Even when only microprocessors are used as boxes, it is not necessary to have all of them of the same type, although this is a

convenience because the need for software writing is kept as low as possible.

Typical number of boxes: 30

Typical box: a microprocessor.

Typical size of memory: 4 k bytes for programs, 4 k bytes for scratchpad.

The programs

The programs run inside the boxes. Each box contains its own (private) programs. Each program is a collection of production rules. Each rule is of the form

$$P_i \rightarrow S_j \qquad j>i$$

or more generally, of the form

$$(P_i, P^1_{i+1}, P^2_{i+2}, \ldots) \rightarrow (S_j, S^1_{j+1}, S^2_{j+2}, \ldots) \qquad j>i$$

It means that if you (the box) see pattern P in retina i, and pattern P^1 in the same place in retina i+1, and pattern P^2 in the same place (that is the corresponding place in the image) in retina i+2, etc., then generate the configurations or skeletons S in retina j, S^1 in retina j+1, S^2 in retina j+2, etc.

Ideally, these programs are written in a high level language such as "L" [23] or CONVERT [12] and a compiler produces fast code for the microcomputers. In practice, and for efficient object code, some low level programming is necessary.

Each program refers to a "window" of the retina. This window is not fixed forever. Instead, the nodes of the fifo have pointers to the retina to be used, as well as to <u>where</u> in that retina the analysis should begin. In fact, the node is pointing towards a window in a retina.

For instance, the following program

111		111
101	→	111
111		111

"cleans" all isolates 0's, replacing them with 1's.

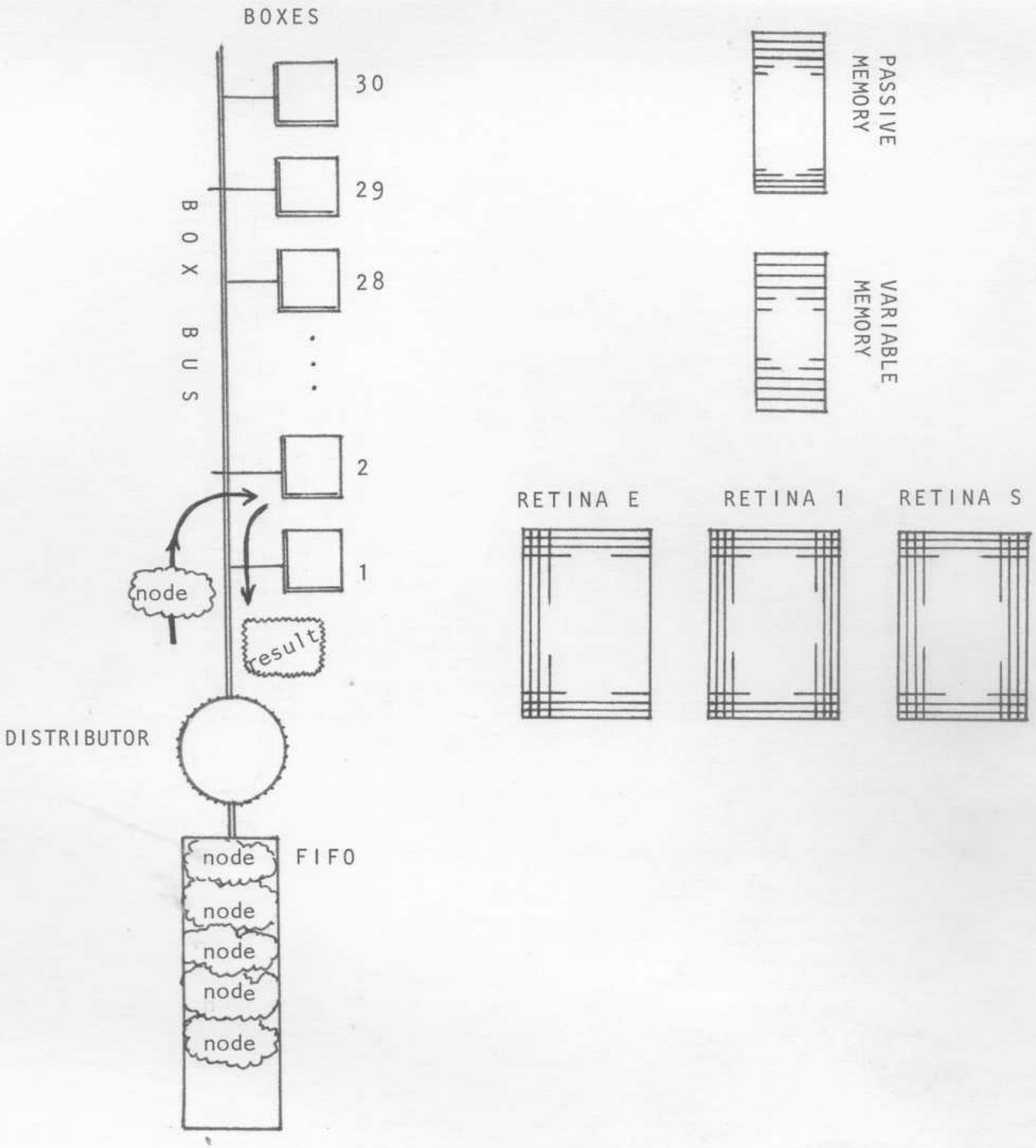

FIGURE 'HOW THE MACHINE WORKS'

Boxes 2 and 29 are idle and request for work to do. The other boxes are completely busy, each carrying on its own task, undisturbed and in parallel. Suppose the arbiter [22] (not shown) chooses box 2. The distributor then originates an exchange where the selected box (box 2) gives its {results} of the task just completed, and receives a {node} that contains the new task requested. The {nodes} are taken from the fifo. The {results} are taken from the boxes and placed into the retinas (connections not shown). In addition, during the evaluation of a {node}, the box has access to the passive and the variable memories (these connections not shown).

PHYSIOLOGY OF THE MACHINE: ITS FUNCTIONING

Overview

The image machine operates as a slave (a peripheral unit) of a general purpose minicomputer (cf. last section). The minicomputer stores the initial image (to be processed) in the retina E, and takes the results (the processed image) from retina S. See figure 'The parts of the machine'.

The processing units of the machine are the boxes. Each box is either busy or is asking for work to do. When a box finishes its work, it requests the fifo's attention because it wants to give back its result, and also wants to process another task (job).

The fifo contains within it a distributor that, upon noticing that a box asks for more work to do, performs the following operations:

1) the box's results (generally a 6 x 6 subimage) are taken from the box and placed in the corresponding retina(s);
2) a new piece of work (a node) ready to be done (but not processed yet) is extracted from the fifo and is given to the soliciting box.

Upon receipt of a new node to process, the soliciting box returns to the "busy" mode.

If more than one box requests new work from the fifo, an arbiter decides which box gets it; the unlucky boxes will keep waiting (and requesting) until they are served.

Thus, it is appropriate to see a box as a "node processing" unit [17], that works fully independently from the rest of the units, and communicates with the world by switching from a "busy" mode to an "idle" mode. The communication with the external world involves an interchange

of the results of the processing of the previous node with a new node taken from the fifo. See figure 'How the machine works'.

During the processing of a node, a box may generate additional work for other box(es) to do; this additional work is expressed as nodes that are stored in the fifo.

The image machine begins to work with a signal from the minicomputer, after the retina E has been loaded by the minicomputer and an initial "starter" node is forced into the fifo. After this, the image machine continues its execution, while the host minicomputer is free to devote its time to its users (under its own real-time or time-sharing operating system).

Work done by the image machine ends when the fifo is empty. This causes an interruption to the host computer, signaling that the results are ready.

Task sequencing, or how the work proceeds

The fifo contains nodes of the form shown in figure 'Node'. Each node can be considered as a specification of work that needs to be done. The fact that this node is (temporarily) stored in the fifo instead of being given immediately for evaluation, indicates that the number of available boxes is finite, so that work ready to be done must ocassionally wait. The fifo is the waiting line.

The distributor fetches a node (the node at the head of the queue, the "oldest" one) and gives it to any idle box (it follows the advice of the arbiter, if needed), or it is put back into the fifo, if all boxes are busy.

The soliciting box receives the node, processes it, and

(1) stores results in some retinas and perhaps in the variable and passive memories,

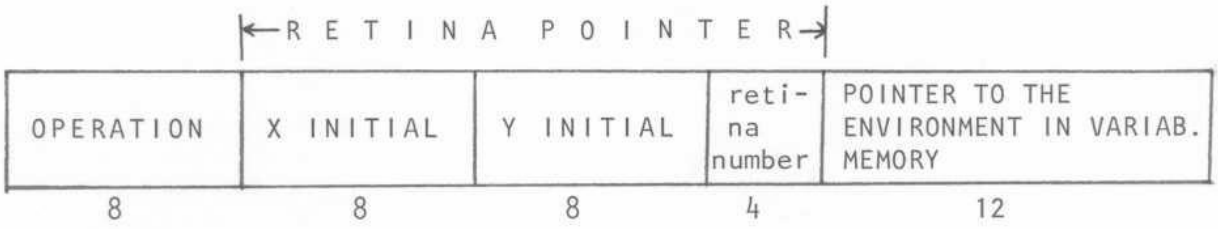

FIGURE 'N O D E'

Each node is a request for work to do. It contains the necessary pointers to identify the task (operation), where to do it (retina pointer) and what the current values (for this node) of the variables (pointer to variable memory) are.

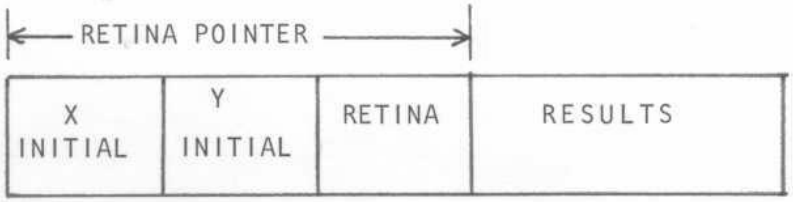

FIGURE 'RESULTS OF A BOX'

The distributor takes the results and places them in the indicated retina at the designated (x,y) window.

(2) it may place back into the fifo, one, or several new nodes, asking in this manner for some other box(es) to do additional processing,

(3) when it finishes, it signals the fifo that it has become idle and available for additional work.

It may also leave a result, which the distributor will place in the corresponding retina (part of the result indicates the target retina where the remainder part should be deposited).

a) If no node is placed in the fifo by the box, it means that, as far as this box is concerned, the box does not know "what else to do to this part of the image". Thus, the box does not ask for additional processing over this part of the retina.

Nevertheless, the partial results placed by the box in some retina i may contribute to the formation of some pattern or configuration P_i, which eventually may trigger (through a procedure explained later) the execution of process $P_i \rightarrow S_j$. Some box, in the future, will perform this process or production rule [16]. Nobody knows now what box will do it. It does not matter, because the nodes contain enough information for a "complete" processing by the production rule.

b) If this box places one node into the fifo, it means that this box wishes some other box to carry on with the operation indicated in the recently inserted node. That is, this box "knows" what other transformations to apply to this part of the image. It knows the continuation of the treatment. And it asks another box to do it.

If the boxes are universal actors (identical boxes, all know how to do everything), there is not much sense for this box to ask another one (through a node in the fifo) to perform an operation that the requester could do just as well. It is better for the requesting box not to ask for help, but to complete the work instead. (But see example 'Neighbor classifier' in Section 'Praxis of the machine: its applications'.)

c) If the box inserts more than one node into the fifo, means that this box requests several additional operations to be performed in parallel, if possible. It creates several sons.

These sons, when eventually executed, need not report back to their father. Their father does not exist. All the sons could do would be to create more sons.

Parts of a node

Each node (figure 'Node') has the following fields:

- Requested operation. It contains a tag (a number) that indicates what work (process, operation, production rule, subroutine) must be invoked.

- Retina pointer. It indicates the retina (not necessarily the input retina, E) that is to be processed. That retina containing the data needed for the requested operation. It also indicates the initial position (x,y) of the "window" in that retina. Thus, the box will not deal with the whole retina, but only with what is inside this window. The form and size of the window depends on the software, but it is generally a square of 6 x 6 pixels, for instance.

- Pointer to the environment. It points to a tree of variables and values, containing the additional parameters (values) required to process the node. The box will take, from this environment, the values of any free or global variables.

Since evaluation takes place in parallel, a variable may have at the same time several values, but a box will get only one: that given by the environment pointed by the node.

Results of a box

The results given back by a box may be placed by the box itself in retina j ($P_i \rightarrow S_j$), or it may be left as final output of the current process. In that case when the distributor is signaled that the box is idle, takes its results and places them in the appropriate retina, before inserting a new node into the box.

Thus, the final results (those to be placed by the distributor, and not by the box itself) have the form of figure 'Results of a box'.

Input/output of images

The image machine behaves as a peripheral unit of a minicomputer (figure 'The machine as a slave'). The latter transfers the images from its mass memory (tapes, disks) to the retina, and viceversa.

If the size of the retina E is smaller than the image to be processed, this is divided in several portions, each of the size of E. Each portion is serially fed to the image machine by the minicomputer and is processed in parallel by the image machine.

For this reason it is convenient for the retinas (or at least, retinas E and S) to form part of the address space of the minicomputer. There is an additional advantage: the minicomputer can use the (large) retinas as normal random storage when the image machine is idle. Another advantage is that raster scan displays [19, 20] can easily be connected to retinas E and S.

When the work is finished

When the image machine has completely processed the retina E

(1) the fifo is empty, or

(2) a bit or pixel in retina S has been set. This is controlled by

software. For instance, if the process "look for number 5" is being run by the image machine, the process should stop after the first "5" is found. The box that found it should store a 1 in pixel (1,1) (or in any other prearranged position), which will be enough to mean "I have finished". May be another "5" is found while this signal is attended. Should this be undesirable, the pixel (1,1) may be a memory address that signals the distributor to stop, as well as to the minicomputer to come to read the results from retina S.

Inter-processor (inter-box) communication

A box never signals to another box specifically. There is no need for direct communication. A box never knows how many other boxes there are, or what they are doing. A box "influences" the behavior of other boxes through changes in the retina and in the environment (variable and passive memories). A box may request "some other boxes" for additional work to do, by entering appropriate nodes into the fifo. But it never knows what boxes will digest these nodes.

A box does not know what box (process) created the node that it is currently processing.

A box never reports back to its "father" box; more likely, it (the father) is dead (completed).

The output of each box is "publicly available" and it is unknown who will use it.

The characteristics mentioned above simplify the programming of the boxes.

The processors (boxes) are not geographically distributed through the retina, i.e. a given box has no "southern neighbor" box.

The processing takes place in parallel; in general, there is no need to wait for the whole retina i to be processed, in order to start processing retina i+1. Processing of a retina starts as soon as there is something to do, and somebody to do it.

Dedication of a box to a fixed pixel or window

A box is not devoted to a particular area of the image. It processes any portion of it; more specifically, it processes the portion (window) indicated by the node that arrives from the distributor.

Priority among boxes

There is no precedence or priority among the boxes. If two boxes are free (idle) at a given instant, one of them (the box closer to the distributor) will receive the next node, and the other box will wait for a short time for its own node. But this is inconsequential, since the boxes are functionally identical, although they may differ in execution time, due to different software or different hardware. (The architecture of the image machine also works well for specialized boxes, but this is not to be discussed here).

In practice, box 1, the box closest to the distributor, is the busiest box; the most remote box is the least busy. If at some time we observe that box i is busy, it can be inferred that boxes 1, 2, ..., i-1 are busy too. Refer to figure 'How the machine works'.

The pattern matchers

How are patterns identified and found in a given retina? We need a procedure to identify certain patterns in the retina and to trigger the

appropriate programs. They are like the "daemons" [10] of real-time information systems.

First of all, the execution of a box may cause additional nodes to be placed in the fifo (cf. parragraph 'Task sequencing, or how the work proceeds'). This means that a given pattern P had already been recognized by the box (the box that placed the nodes), so that a rule $P \rightarrow S$ is now needed for its execution.

Many patterns, nevertheless, will be formed only with the agglutination of the partial results of several boxes; none of these boxes can detect that it is contributing to the formation of certain pattern.

For these patterns to be detected, it is necessary to program some boxes (or nodes) with a "search" procedure that reads as follows:

$$P_i \rightarrow S_j$$
$$P^1_k \rightarrow S^1_1$$
$$P^2_m \rightarrow S^2_n$$
$$\vdots$$

That is: search for patterns P, P^1, P^2, ... and execute actions S, S^1, S^2, ... These actions are the creation of nodes and their insertion into the fifo; these nodes will invoke appropriate "real" actions to be performed on the retinas. The "search" procedure determines where the work to be done is, sends a node to the fifo and keeps searching. It terminates when (for instance) a given retina was scanned a couple of times and no new nodes were created: none of the sought patterns were found [30].

PRAXIS OF THE MACHINE: ITS APPLICATIONS

Overview

This paper presents the design of a machine that has not been built. Therefore, the applications shown in this section try to find out how easy it is to program the machine to do certain "useful" things. Considerable more experience is needed before it can be said that this is a nice or useful architecture.

Picture compression

A recent paper [25] explores the use of a microcomputer to do picture compression for each band of a Landsat image. Our idea is to use the n microprocessors of the image machine to carry on the compression n times faster.

The basic algorithm is to find out how many repeated levels there are in a row, and to send (or store) a code that contains the number of repetitions.

For instance,

222233555555444 is coded as (4,2), (2,3), (6,5), (3,4)

meaning: four 2's, then two 3's, then six 5's, etc.

The proposed algorithm [25] can be coded in a program called "compress". This program will be given to each box. Assume there are 20 microprocessors. Let us begin with the retina E containing the full 256-row image, and with the fifo having 256 nodes of the type

compress	line i of E	not used

while the retina S is full of zeroes.

As each box finds itself idle, acquires a node from the fifo, and therefore processes a line. The results are placed in line i of S.

The work finishes when the fifo is empty. See figure 'Picture Compression'. Then the minicomputer will take the results away from S, presumably to write them in a disk file of suitable format.

Maximum likelihood classifier

To classify a multispectral image according to the spectral signature of each pixel, one needs [11] to find the distance of such pixel to every one of the mean values of representative classes: wheat, corn, alfalfa, etc.

Again, the work is simple, and resembles the picture compression task: we write a classifier able to classify a whole line, and use this algorithm instead of "compress" of the picture compression task.

Neighbor classifier

It is possible to take into account the influence of its neighbors in the classification of a pixel. This is a simple kind or relaxation method [29]. What is proposed [3] is to classify first according to the maximum likelihood rule, and then to take into account the neighbors of each pixel before the final veredict is rendered for each pixel.

There are two different processes:

1) Classify E according to the maximum likelihood rule, and place the results in retina 1.

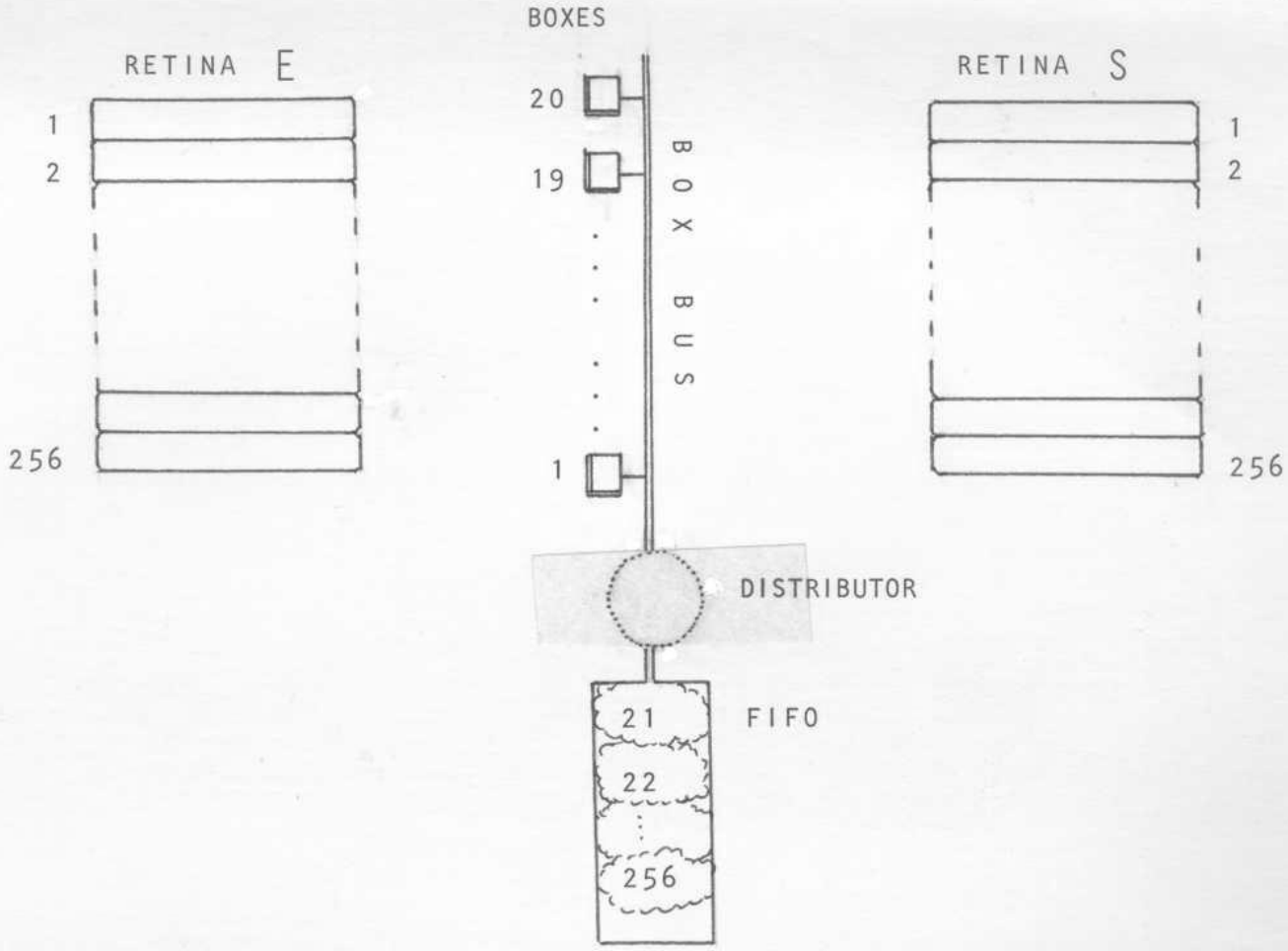

FIGURE 'PICTURE COMPRESSION'

The input image (in E) is considered to be a collection of 256 rows, each of them to be compressed according to algorithm "compress." Initially, 256 nodes, all with the order "compress" but with different line numbers, are placed into the fifo. As soon as a box finishes to compress a line and has stored its results in the retina S and a new node will be given to it. The machine halts when the fifo is empty.

2) Modify the class of each pixel of retina 1 according to some criteria that takes into account the classes (in retina 1) of its eight nearest neighbors. That is, observe retina 1 and place the results in retina S.

The process begins with the raw image in E, with 256 nodes in the fifo of the type

classify	line i of retina E	not used

and 256 nodes of the type

reclassify	line i of retina 1, from column j on	not used

In the processing of the first 256 nodes, the machine acts just like in the previous case, but it leaves the results in the intermediate retina 1.

The second 256 nodes instruct each box to reclassify line i of retina 1, from column j (initially set to 0) on. The results will be placed on retina S. Work concludes when fifo becomes empty.

The algorithm has been described as 2-step sequential: first all pixels are classified and then they are reclassified. But this need not be the case. Suppose the nodes get "mixed" in the fifo (for instance, if it were a stack and not a fifo); then some boxes will receive the order "reclassify" even if the work on classification still exists. It is only necessary a minor change in the "reclassify" program: the box has to test whether all the neighbors have been classified (none of them has 0 as class in retina 1) before a reclassification is done.

Once a pixel is reclassified, the pixel to its right (on the same row i) is tried: j:= j+1. But as soon as a pixel is "un-reclassifiable" (because some of its neighbors are not yet classified), the box quits and

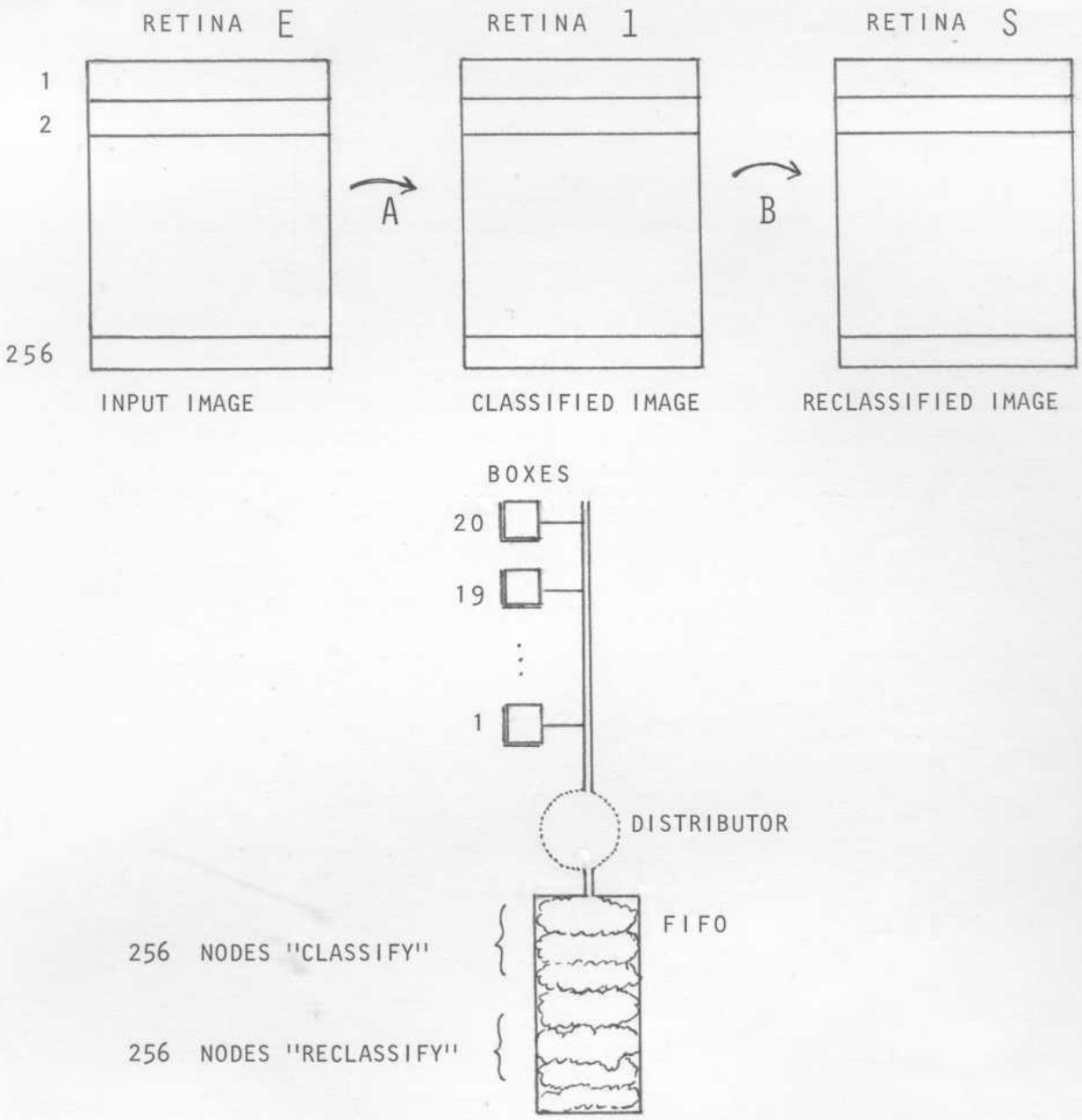

FIGURE 'NEIGHBOR CLASSIFIER'

The classification proceeds in two stages: (A) input retina E gets classified according to the maximum likelihood rule, the results appear in retina 1; and (B) the results in retina 1 are reclassified taking into account the classification of the eight nearest neighbors of each pixel, yielding the final results that go to retina S.

Notice that iterations on the retina S is a relaxation method [29].

returns to the fifo the node it was processing, but with the new value j. That is, the box says: reclassification work is still pending on row i from column j rightwards. This node is returned to the fifo. Later, some other box will fetch it and it may have more luck. Meanwhile, the box reports itself idle, and a new node will come about to give it new work.

Table lookup classifier

This classifier could be implemented in the image machine in a manner similar to the maximum likelihood classifier, but using the passive memory as the table [1] that dictates the classification.

What happens with the programs that work near the edges of the images and retinas? The algorithms herein described do not mention this. They need changes to operate well at the edges.

Other picture processing tasks

Tasks that will begin to fully use the capabilities of the machine are, for instance, linear feature detection, relaxation, interaction of shape and context [14] and scene analysis.

PRECURSORS OF THE MACHINE: PREVIOUS WORK

Other parallel machines

The field of parallel processing is rapidly expanding. The bibliography of [13] may be helpful. It is interesting to mention the Cm* [27] architecture and the Arcade machine [4], as well as the AHR Project [13, 24] at our AHR Laboratory (IIMAS-UNAM), as agglomerations of mini or microprocessors to perform macro tasks.

Other image processors

The cube hardware classifiers of GE Image 100, of Toshiba and of the color video processor being built at IIMAS-UNAM [19] are remote ancestors of the image machine. Duff [8] is building an interesting architecture where the boxes are geographically related, and can explicitly communicate with each other. The PM-4 machine [6] at Purdue University is an example where the program has to explicitly state "do this and that in parallel".

Our previous work: software

The P.R. System [2, 18] is a flexible interactive software intended for remote sensing applications. More recently, extensions [9] are made to couple it to a geographic data base.

Scene analysis [14, 17] work was performed some time ago, and more recently, shape similarity measurements [5] for two-dimensional silhouettes.

Pattern matching languages such as COMIT, SNOBOL or CONVERT [12, 15, 16] are well known and used.

The AHR Machine [13, 24] has been simulated in our AHR Laboratory.

Our previous work: hardware

In 1976, Luis Lyons [20] built a TV color image display at Instituto de Ingeniería, UNAM. In 1978 and 1979, a group of people is building an improved version [19].

In 1978, Peñarrieta [22] built a fast memory multiplexor.

Currently, the AHR Laboratory is used as a vehicle to design the AHR machine [24].

CONCLUSIONS

The marriage of parallel machines with picture processing seems to be a stable binding, due to the large amounts of data and to the relatively loose way in which these data (pixels) influence each other: usually, the interaction decays after a few neighbors. The diameter of the context is small [14].

This paper presents the design of a machine that processes images in a parallel fashion. The way the processors (boxes) are organized is called "heterarchy", because there is no box "above" others; all are equal and no one explicitly communicates with other boxes. Nevertheless, it is possible to do useful work under these conditions, as some examples of the section 'Praxis of the machine: its applications' show.

The programming of the machine tries to stay away from asking the programmer to explicitly state what things should be done in parallel. Instead, the programs have a pattern-matching style [12] of the form

$$P \rightarrow S$$
$$P' \rightarrow S'$$

"if you see pattern P, produce skeleton (output) S; if you see P', produce S', etc."

At the end of the paper some examples are given of how the image machine can be used for several common image processing tasks. No actuals programs have been written.

More work is needed, probably, to ascertain the use of this machine and to find out changes and improvements.

Advantages of the machine

= The boxes are never interrupted. Processing is asynchronous; each box takes its needed time; no box waits for somebody else to be ready.

= The boxes never wait (unless the fifo is empty).

= There is no priority in the tasks.

= There is no need for the programmer to write or think in parallel, that is, to explicitly write commands such as "do this and that in parallel"; "wait for your brothers to finish before you go on", etc.

= High speeds are attainable, since many boxes perform the processing.

= The machine is modularly expandible; no need to do large expenditures at once.

References

1. Avilés, R. Clasificador de búsqueda en tablas aplicado a percepción remota. B. Sc. Thesis, Electronics, ESIME-IPN. Mexico. 1979.

2. Barrera, R., et al. Detección y cuantificación de recursos agropecuarios mediante análisis por computadora de fotografías tomadas desde avión y satélite. México, IIMAS, serie naranja; 157. 1977

3. Baz, Guillermo. Clasificación bayesiana aplicada a la percepción remota. M. Sc. Thesis, IIMAS-UNAM, 1977.

4. Brandwajn, A., Kruchten, P., Hernandez, J-A. ARCADE- A system for research and education in computer science. Départment d' Informatique, Ecole Nationale Supérieure des Telecommunications., Paris, 1977. ENST-D-77014.

5. Bribiesca, E., and Guzmán, A. Shape description and shape similarity measurement for two-dimensional regions. Proc. 4th Intl. Conference on Pattern Recognition, Kyoto, Japan, 1978. 608-612.

6. Briggs, F.A., Fu, K.S., Hwang, K., Patel, J.H. PM^4: a reconfigurable multiprocessor system for pattern recognition and image processing. Purdue University, TR-EE-79-11. 1979.

7. Dennis, J. B. Packet communication architecture. Proc. 1975 Sagamore Computer Conf. on Parallel Computation.

8. Duff, M.J.B. A user's look at parallel processing. Proc. 4th Int'l. Joint Conf. on Pattern Recognition. Kyoto, 1978. 1072-1075.

9. Gaillat, J-P., Méndez, R. Diseño para un ayudante inteligente de fotointérprete. Memorias del II Seminario Intl. sobre el uso de los sensores remotos. Mexico. 1979.

10. El programa de recursos humanos en Informática: sistema automatizado. Subdn de Política Informática, Sist. Nal. de Información, S. P. P. México 1979.

11. Guerra, V. Clasificación por computadora de imágenes de satélite. México, IIMAS, Serie Naranja; PR--75-4. 1976

12. Guzmán, A., and McIntosh, H. V. "CONVERT." Communic. A.C.M. 9, 8, Aug 1966. 604-615.

13. Guzmán, A., and Segovia, R. A parallel reconfigurable LISP machine. Proc. Int'l Conf. on Information Sciences and Systems, Patras, Greece, 1976.207--211.

14. Guzmán, A. Analysis of curved line drawings using context and global information. Chapter 20 of Machine Intelligence VI, D. Michie and B. Meltzer (eds)., University of Edinburgh Press. 1970. 325-375.

15. Guzmán, A., and McIntosh, H. V. A program feature for CONVERT. Memorandum MAC M 305 (AI Memo 95), Project MAC, M.I.T. April 1966.

16. Guzmán, A. CONVERT - diseño de un lenguaje para manipulación simbólica

de datos y de su procesador correspondiente. B. Sc. Thesis(Electronics), ESIME-IPN. 1965.

17. Guzmán, A. Decomposition of a visual scene into three-dimensional bodies. Proc. AFIPS Fall Joint Computer Conf. 33, One, 291-304. 1968. Also as Part VI of: Computer Methods in Image Analysis (Aggarwal, J.K., Duda, R. O, and Rosenfeld, A., eds). IEEE Press. 1977. 324-337.

18. Guzmán, A., Seco, R., and Sánchez, V.G. Computer Analysis of LANDSAT images for crop identification in Mexico. Proc. Int'l Conf. on Informⁿ Sciences and Systems, Patras, Greece, 1976. 361-366.

19. Hermosillo, A., et al. Sist. de despliegue y proc. por hardware de imágenes digitalizadas. Mem. II Semʳ Int'l sobre el uso de sensores remotos. México 1979.

20. Lyons, Luis. Digital image display system. Instituto de Ingeniería. National Univ. of México. 1976. Unpublished.

21. McElroy, J.H., et al. CO_2 laser communication systems for near-earth space applications. Proc. IEEE 65, 2, Feb. 77. 221-251.

22. Peñarrieta, Luis Hugo. Ms.Sc. Thesis (Electronics). Div. de Est. Sup., Fac. de Ingʳ, Nat'l Univ. of Mexico. 1978.

23. Radhakrishnan, T., Barrera, R., Guzmán, A., and Jinich, A. Design of a high level language for image processing. Mexico,IIMAS, Serie Naranja. PR-78-21. 1979.

24. Rosenblueth, D., and Velarde, C. La máquina AHR para procesamiento en paralelo. Primera etapa. México,IIMAS. Serie Naranja. AHR-79-2. 1979

25. Seco, R., Radhakrishnan, T. Micro-processor controlled compression of remotely sensed data. Proc. Mini and microcomputers International Symp. California. Jan 1979.

26. Sistema para despliegue de imágenes digitales de radiografías. Sistemas Computacionales Avanzados, S. A. Monterrey, N. L. México 1979.

27. Swan, R.J., Fuller, S.H., Siewiorek, D. P. Cm* - a modular, multi-microprocessor. Proc. AFIPS Vol 46. 1977. 637--663.

28. Vital, R., Garza, M., Seco , R., Pérez, D., Jinich, A. Evaluación del uso del suelo en la cuenca del río Pánuco. México IIMAS Serie Naranja. PR-78-22. 1979.

29. Zucker, S.W., Hummel, R.A., Rosenfeld, A. An application of relaxation labeling to line and curve enhancement. IEEE Trans. Comp C-26, 4, Apr 77. 394-403.

30. A node is created if the pattern is found and a "processed" mark is off. Then, this mark is set, to prevent additional detections of this pattern in this same position.