NETWORK AND ARRAY ARCHITECTURES FOR REAL-TIME PERCEPTION

by

Leonard Uhr

Computer Sciences Technical Report #424

March 1981

Network and Array Architectures for Real-Time Perception

Leonard Uhr

Computer Sciences Dept., Univ. of Wisconsin

Abstract

This paper examines the computing demands that must be met by a system capable of scene description and perception of real-world moving objects. Then a particular architecture that appears to combine a number of attractive features is suggested as appropriate for at least a certain class of scene description systems. A brief survey is made of the major different kinds of computer systems that have been built, or designed, and of the different sources of potential speed-up of processing that have been exploited. Finally, a number of alternative possible hardware architectures that might be capable of handling real-time perception of moving objects are suggested, and examined.

Network and Array Architectures for Real-Time Perception

Leonard Uhr

Computer Sciences Dept., Univ. of Wisconsin

Summary Outline

I. The requirements and specifications for computer systems capable of perceiving moving objects in real time:
- A) The size of a real-world scene
- B) The size of the living eye's retina and visual system
- C) Estimates of the amount of processing needed:
  1. times needed by today's computer programs
  2. estimates from these times
  3. times needed by human beings
- D) Estimates of the additional time needed for moving pictures' frames subsequent to the first frame

II. The proposed architecture:
- A) Large array
- B) Smaller scanning array
- C) Network of MIMD processors (might combine with B)
- D) One powerful processor to:
  1. control (though might use a smaller one)
  2. store image and transforms (though might eventually disperse this, storing where appropriate)
  3. simulate processors not yet physically present

III. Types of parallel systems:
- A) Networks like c.mmp, cm*, X-trees, pyramids and n-cubes
  1. speed-ups that can reasonably be expected from plausibly realizable networks
  2. future, and ideal, possibilities
- B) Fast scanners
- C) Pipelines
- D) True parallel arrays

IV. Sources of parallel speed-up:
- A) Pipeline (e.g., Cytocomputer)
- B) Parallel fetches of near-neighbors
- C) Parallel fetches of more bits for grey-scale, weight, number
- D) Parallel application of several operations (e.g., PICAP)
- E) Parallel processors (e.g., CLIP, DAP, MPP)

V. Interesting possible architectures:
- A) Very large array
- B) Fairly large array of (slow and cheap) scanners
- C) Relatively smaller array of (faster, more expensive) scanners
- D) Very fast and well architectured pipe-line scanner
- E) A small number of very powerful pipe-lined scanners
- F) Sets of arrays, in series
- G) Mixtures of SIMD and MIMD architectures

Introduction

This paper examines the computing demands that must be met by a system capable of scene description and perception of real-world moving objects. Then a particular architecture that appears to combine a number of attractive features is suggested as appropriate for at least a certain class of scene description systems. A brief survey is made of the major different kinds of computer systems that have been built, or designed, and of the different sources of potential speed-up of processing that have been exploited. Finally, a number of alternative possible hardware architectures that might be capable of handling real-time perception of moving objects are suggested, and examined.

The Demands on a System for Real-Time Perception of Moving Objects

Real-world objects can move at any speed (up to the speed of light). And arbitrarily large numbers of objects might be moving, in the same scene, in arbitrarily different directions and manners. We might pose the "general problem of the perception of motion" as one of recognizing any number of objects moving in any number of trajectories at any constant or changing speeds. Or we might restrict the problem to only a manageable few objects moving at some speed less than some arbitrary amount (e.g., 1,500 miles per hour, viewed at some standard distance).

But it seems most reasonable to ask that the system handle those objects, moving at those speeds, that a human being can perceive; for example, a galloping horse but not a bullet. This is not to insist upon modelling human perception, but rather to point out that the "general" problem is potentially infinite in its size, speed, and complexity, and that we must cut it down to reasonable proportions.

The Enormous Computing Burden to Perceive Objects in Motion

Most researchers developing "vision systems" today pose their ultimate problem as one of handling a scene on the order of 25Ø by 25Ø, 5ØØ by 5ØØ, 1,ØØØ by 1,ØØØ or 3,ØØØ by 3,ØØØ "pixels" (picture elements), as resolved by a television camera. For simplicity, let's assume the scene is resolved in a 1,ØØØ by 1,ØØØ array, giving 1,ØØØ,ØØØ pixels, each containing a 1Ø-bit value (representing color or grey-scale intensity). This is an extremely large amount of information. But I should note that a human retina contains roughly 1Ø,ØØØ,ØØØ cones (for shape and color vision) and 1ØØ,ØØØ,ØØØ rods (for change, motion, and grey-scale vision).

To date only a few attempts have been made to have a computer program try to recognize and describe the objects in a full television picture (usually on the order of 5ØØ by 5ØØ). These programs have taken from 5 or 1Ø minutes to 1Ø or 2Ø hours of CPU time on rather large computers (e.g., Univac 111Ø; DEC pdp-1Ø). From these times (for programs that are just barely beginning to handle a single static scene) we might extrapolate that several hundreds, or even thousands, of hours of CPU time might be needed to perceive the 3Ø television frames that a human being is able to perceive in a single second. Reddy has suggested that for segmentation alone 1,ØØØ processes might be needed per pixel, and therefore from 1 to 1Ø billion instructions per second (BIPS) are needed to segment a single scene in one second.

Some Possibilities of Speed-Ups Using Parallel Programs

We can use the more parallel(-serial) programs that have been developed with parallel computers specifically in mind, such as the "pyramid" and "cone" systems that I, and a number of other researchers, have been developing (Hanson and Riseman, 1974, 1978; Klinger and Dyer 1974, Levine and Leemet, 1976, Levine 1978; Tanimoto, 1976, 1978; Uhr, 1972, 1974, 1976, 1978; Uhr and Douglass, 1979; Douglass, 1977, 1978; see Tanimoto and Klinger, 1980) to give alternate estimates of the time needed. I would estimate that from 5 to 50 transforms of the sort that a suitably designed multi-computer network of parallel processors could execute in a single machine instruction might be needed in each of from 10 to 50 layers. This would need a sequence of from 50 to 2500 instructions. Thus, for example, if each processor took 10 microseconds per instruction such a system would need only .50 to 25 milliseconds for an entire (static) scene. (But large amounts of time can easily be eaten up by the serial processes used by most other kinds of vision systems.)

Empirical Evidence that Human Perception is Highly Parallel

The human perceptual system throws some extraordinarily interesting light on this problem. The "basic meaningful instruction time" for the brain appears to be from 1 to 2 milliseconds, if you will accept the following simple and to my mind rather compelling argument:

The retina is excited by light energy, sending trains of impulses through the neurons of the visual system and the cerebral cortex. The synaptic junctions between neurons compute functions of the impulses firing into them, and fire out their results. That is the underlieing microstructure with which the brain "perceives" and "thinks." Therefore the time needed for a synaptic firing, plus the time needed for a neuron to carry the resulting information to the next synapse, can be thought of as the "basic meaningful instruction time" of the brain.

Now a synapse takes about 1.5 milliseconds to do its job, and an impulse moves along a neuron in a time appreciably smaller than that. So 1.5, or, to be conservative, 1 to 2 milliseconds, is a quite reasonable estimate. Note how slow this is compared to a computer.

Psychologists have also collected extensive data on the time a human being needs to recognize and perceive objects in a wide variety of different scenes. Without bothering to pin this figure down precisely, it lies somewhere between 5Ø or 1ØØ milliseconds and 1 second. This gives us a quite amazing result: *The human brain perceives with a serial sequence of processes from 25 (or fewer) to at most 1,ØØØ deep*!

A television camera typically takes a new picture every 3Ø milliseconds. This is necessary to give a smooth flowing picture of moving objects, because of the "critical fusion frequency" of the human visual system which (depending upon illumination, resolution, type of object, and a variety of other factors), lies in the range of from 1Ø to 3Ø static images per second.

This suggests that each new static picture might have to be processed in 3Ø milliseconds, which would force us to lower the lower extreme given above to *only 15 to 3Ø serial processes when* _scenes of moving objects must be perceived_. It seems more likely, however, that a well-designed visual system (including living visual systems) would not need to process each new scene in its entirety, but rather would use what it had gathered from the moving scene of the most recent past to direct, and drastically cut down on, its processing.

There are at least two additional factors that are almost entirely conjectural:

1) How complex is each individual process in the brain? We know that hundreds, or often many thousands, of neurons synapse on a single neuron. So, conceivably, the functions a neuron computes could be very complex indeed. But somehow it seem intuitively more plausible (at least to me) to assume that much of this is a matter of redundancy and stabilization, and that the functions aren't all that complex - that the degree of parallelness in the function computed by the individual neuron is on the order of 5 or 1Ø, or a few hundred at most. Indeed there is experimental evidence that cortical neurons onto which, typically, 1Ø,ØØØ or so neurons synapse, will fire when only two or three of those neurons fire into them.

2) How much can previous knowledge cut down on the processing needed for each new input frame, and how great a speed-up in processing can this effect? Here I think we know so little that we have virtually no grounds for estimating. Estimates must wait until we have developed computer systems that exhibit some of this speed-up, while performing reasonably well. But somehow it seems quite unlikely that significantly more than one order of magnitude speed-up could be attained, and it does not seem un-

likely that there might be little or no speed-up at all, since this is hard to achieve in parallel systems.

A Network-Array Architecture for Perception, Cognition and Intelligent Systems

We will now examine an architecture for a network-array of processors that appears to be an appropriate system on which to run programs for perception and cognition, along with preliminary specifications of the actual first-approximation to this longer-term goal that we have been considering. We will then examine the different possible sources of increased power and speed, and alternative types of networks and arrays that might exploit them.

A) Information is input from the sensing device into a large buffer memory, and then processed by a sequence of instructions that are executed by a very large hardware-parallel array of relatively simple processors specially designed with the near-neighbor operations most people feel are appropriate for the first stages of image processing, image enhancement, and scene analysis.

This suggests a cellular array like CLIP4 (Duff, 1976, 1978), DAP (Flanders et al., 1977; Reddaway, 1978, 198Ø) or the MPP (Batcher, 198Ø).

B) Next follows a sequence of instructions executed by an appreciably smaller array of processors, each of which is appreciably more powerful and has access to an appreciably larger neighborhood of pixels. Since today it would be premature to contemplate an array larger than 96 by 96 (CLIP4) or 128 by 128 (the projected MPP), it seems preferable to think in terms of a second-level array built from the very powerful bit-slice processors, rather than from the far cheaper micro-computers on a chip - for the following reasons: The 96 by 96 array of information will have been appreciably reduced by the sequence of instructions effected by the CLIP array, and can now most appropriately be stored in a 48 by 48 or 32 by 32 array. This means that a 12 by 12 or 16 by 16 array of bit-slice processors (see C) below) could quite adequately handle things, with each processor scanning a sub-array of 2 by 2 to 4 by 4 (or even more) cells.

This is a small enough number to make the bit-slice processors economically feasible. Here we are estimating that reasonably powerful processors of type C) below would cost about $2,000 each to build, if each is able to execute a different set of instructions, and is a full-blown processor-plus-controller. But we have designed a system (Uhr, Thompson and Lackey, 1980) that has a 4 by 4 array of 16 such processors, with 16 slave processors tied to each, giving a 16 by 16 array in toto. This will give a 16 processor MIMD system, where each processor is a 16 processor SIMD system. It should further mean (if we can overcome a number of hardware design problems without too many compromises) that this resource will also serve as a relatively general-purpose 16 processor network.

(Compare this with alternative F) below, with a sequence of successively smaller CLIP arrays. The CLIP array that we propose for the "low-level" processes will serve to replace the first 2 or 4 of these layers. Then the network-array will handle the intermediate-level layers).

Finally, a general-purpose computer will handle the deepest layers of processing. Information will be transferred to this computer at whatever layer the programmer feels is appropriate (because the degree of potential speed-up from parallel processing has become so small as to be relatively unimportant, whereas the advantages of allowing a processor to make global assessments of information no matter where it might be have become sufficiently great).

In our particular embodiment of this type of system, (we would use a VAX with several million bytes of memory) this transfer may well occur relatively early (removing much of the burden of programming a network efficiently). The VAX will also serve several other crucial purposes: The raw image (along with intermediate outputs from the array and from the network-array) will all be stored in the VAX high speed memory, and accessed over the UNIBUS or, possibly, its very fast (13.3 million bytes per second) SBI bus. (It would be preferable to have direct interconnections along with buffer memories between the separate resources; but that would entail a good bit more hardware, and expense. With this system delays will be relatively small, and we will be able to estimate quite precisely how much a better-designed system would increase speeds.) The VAX will be used to handle whatever processing the programmer chooses not to assign to the array or the network-array. Thus only when we discover how to program processes in parallel need we move them off the VAX to the other resources. And we can simulate new kinds of resources on the VAX, including ones we are contemplating acquiring, or trying to design.

The Major Types of Parallel Systems Built or Proposed

Clearly, computers must handle scenes much faster than presently possible, to hope to succeed at real-time real-world vision. The alternatives that have been proposed include faster processors, concurrent processors, networks, scanners, pipelines, and arrays. Let's examine these briefly:

Speeding up Processors by Brute Speed of Faster Technologies

Today's technology has given us machines that execute meaningful picture-processing instructions in from 1ØØ nanoseconds to 1Ø microseconds. A serial computer must iterate such an instruction; a parallel array need execute it only once, _at each and every cell of the array_. (I am ignoring extremely fast and expensive technologies, and assuming a wide range of rather unusual - for ordinary computer users - parallel near-neighbor image-processing instructions.)

It is conceivable that we will see cryogenic computers with speed-ups on the order of two, or even three, orders of magnitude during the next ten years. But that will be the end, since we will have hit the limiting speed of light. And it seems likely that these will depend upon expensive, bulky and limited (in size) computers, since they will have to be kept within an environment that maintains their temperatures close to absolute zero.

Speeding up Processing by Improving Architecture

We can, in theory, speed up processing to whatever degree we desire, by building special architectures that compute arbitrarily more complex functions in parallel. That is, any serial program (computer) can be converted into a parallel program (computer) - if only by building a system with separate hardware modules to sense each possible input state, and output the resulting appropriate symbol-string. (I should note that this is what Rosenblatt appears to have been suggesting with his 1-layer perceptrons, or at least what Minsky and Papert took him to be suggesting in their book on perceptrons.)

But this is an exponentially explosive procedure. Worse, it quickly grows out of the bounds of reasonable economy and well-balanced architecture. Today we see people building machines capable of parallel fetches of 9 (Duff, 1976), or even 25 (PICAP II Kruse and Danielsson, 198Ø) nearest-neighbors (and of course for arithmetic operations 32 or 64 bit numbers are often fetched in parallel).

But it seems unlikely that anybody will ever want to build systems that would fetch thousands, or even a few hundred, pieces of information in parallel in order to compute complex structural functions over them. Thus we can expect to get one order of magnitude increases in speed from parallel input of the separate components of a picture processing function. But it seems unlikely that we can get two, and certainly not more than two.

When arithmetic is appropriate (as in handling grey-scales, intensities, and weights) we can get roughly 1Ø, 3Ø or 6Ø fold improvements. (This is already done in serial computers, which fetch and transform up to 64 bits of numerical information in parallel. But the large parallel arrays of today work with only 1, or at most 4 bits of information at a time; otherwise the cost of hardware would be excessive, since thousands of processors are involved.)

By building a special-purpose scanner we can increase speed by 2 to 10 times. Appropriate architectures for the whole system, encompassing the buffer memory that contains the tv image of the scene, the large array (or scanner), and a general-purpose computer, can also increase speed considerably. E.g., Kruse's (1976, 1978) PICAP I may well gain one order of magnitude, or more, in speed from a specially designed scanner, appropriate hardware for picture-processing operations, and a capability to scan over a sample of the large (raw or transformed) image as a function of what the program has found out so far.

Still another very interesting and powerful system can, potentially, gain 100-fold in speed because of its pipeline of 118 processors (Sternberg's Cytocomputer, Sternberg, 1978, 1980). But it seems unlikely that such a long pipeline can be fully exploited unless it has very large and expensive temporary memories associated with it to store intermediate results.

Networks of Concurrent Processors

Rather than on the one hand using one traditional single-CPU serial computer, or on the other hand using what to many people appear to be unacceptably specialized large arrays or scanners, we can try to couple a number of individual computers (each with one CPU and at least a little bit of its own memory) into a network of processors. (Note that all three alternatives are actually "general-purpose" in the sense that they all are equivalent to "universal Turing machines" and therefore, potentially, they all can compute exactly the same set of functions, given enough time. But the crucial problem is one of efficiency: Are they fast enough? Are they as cheap as possible?)

Today people have designed or built networks of 5, 16 (e.g., Wittie and van Tilborg, 1980; Despain and Patterson, 1978) or at the most 50 (Swan et al., 1977) processors of this kind. [See Wittie, 1976, 1978; Sullivan et al., 1977, Goodman and Sequin, 1981; Uhr, 1981 for descriptions of networks that might have thousands, or millions, of processors.] A large variety of different interconnection patterns have been proposed that might, potentially, allow thousands, or even millions, of processors to be connected together. But the problems of developing operating systems, programming languages, and, worst of all, actual programs that make efficient use of very large numbers of processors of this sort appear to be extremely difficult, and relatively little progress is being made. Results with the small networks already built (chiefly c.mmp, Fuller, 1976, and cm*, Swan et al., 1977, Carnegie-Mellon) suggest that 5 or 15 processors will speed up processing only 2 or 4 times, and then only on programs that are highly parallel, where there is virtually no need to pass messages between processors.

These first early results may merely reflect the difficulty of the problems of designing, building and programming networks, rather than revealing any inherent and fundamental limitations. But for the moment there is little reason to count on large increases in speed from large numbers of processors connected in a typical network. One order of magnitude improvement seems possible, two orders of magnitude seems quite unlikely.

The Most Promising Alternative Architectures for Computers Capable of Real-Time Perception of Scenes of Moving Objects

Increases in brute speed of the hardware, increases in the number of bits of information that the individual processor can fetch in parallel, improvements in the architecture of the processors so that they are designed as appropriately as possible for picture processing operations, pipipelining, and improvements in the design of the overall system can all effect significant increases in speed. But the only way we can continue to increase speed, with no potential absolute limits (other than the number of particles in the universe) is by adding more processors; and the best way to handle profligate increases in processors appears to be with relatively simple, regular and well-structured architectures.

During the next few years judicious combinations of improvements from this large set of sources of possible speed-up may well prove to be sufficient. Kruse's PICAP is an unusually successful, and instructive, example of how to gain enormous amounts of power and efficiency at a relatively cheap price by careful design not only of the scanning processor but also of the total system. And networking together 10 or 20 much-faster-technology general purpose computers (as Binford and Reddy have suggested doing at Stanford and at Carnegie-Mellon) may give similar improvements of from 2 to 4 orders of magnitude in speed. This suggests that combining all these sources of speed-up might, conceivably, give 4 to 8 orders of magnitude improvements.

But the true parallel array (which will probably have to sacrifice some, but by no means all, of these different subsidiary types of improvements, because their increased costs would become excessive when multiplied by the thousands or millions of processors now involved) appears to be the only way to continue to increase speed at the same time that increasingly larger scenes are processed.

The very large array can come in several models, and take advantage of at least some of the other potential sources of speed-up, as follows:

A) A pure array would have one processor assigned to each pixel in the raw input scene, thus giving the most extreme form of parallelism. If we want to contemplate numbers as high as 100,000,000 (to model the 100,000,000 rods in the human visual system) each individual processor will have to be as simple and as cheap as possible, and there would seem to be little opportunity to take advantage of other sources of speed-up from more expensive technology, wider parallel input busses, or elaborately designed architectures. But, as we see in the CLIP arrays (Duff, 1976, 1978), near-neighbor logical operations can be built into the hardware.

It seems unlikely that anybody would seriously contemplate building 100,000,000 processors (unless a technology developed that made this very simple and cheap, e.g., by growing crystals). And since such a system would be very inefficient at the later stages of perception it would almost certainly be combined with other types of resources, as discussed above.

More plausible would be a smaller array of this sort that itself was scanned over the larger retinal array. For example, we can today conceive of building a 1,000 by 1,000 CLIP-type array (by 1985 this should be quite possible at a cost appreciably less that $1,000,000). This would need only 100 iterations to scan the larger 100,000,000 rod array. Similarly, even when we want to process smaller pictures (as will usually be the case) it will often be preferable to have a small array scan over the larger picture array. Thus Duff (personal communication) contemplates using a 100 by 100 or a 500 by 20 CLIP4-type array to scan over a 500 by 500 television image.

We can assume that this type of array will perform its basic operation in one microsecond rather than the 11 microseconds needed by CLIP4 (or even faster if faster and more expensive technologies are used). The speed demanded by the particular range of programs that the system will be expected to execute will determine how many such iterations can be tolerated. But only after we have gained experience in writing such programs will we have a clear idea of how deep their serial sequence of processes might be. For now we have only the rough estimates given above - that if we can find the right programs they should have a serial depth of only a few thousand, or even a few hundred, instructions.

Each CLIP processor today costs roughly $5 to $25. But extrapolating into the future, we might use the basic cost of a chip - about $20 or $100 for a new chip if produced in thousands, but going down to $1 or less in quantities of millions - to predict future prices. It may be possible with VLSI technologies to put significantly more than 8 processors on each chip (though not too many more, because the limited number of pins to each chip will not accommodate the continually increasing number of connections that would be needed). But it seems more reasonable to pack more memory and more processor power on the chip instead. So let's assume that only 16 (appreciably faster and more powerful) processors are placed on each chip. The overall hardware cost might then reach 1Ø cents a processor, and the cost of a 1,ØØØ,ØØØ processor system approach $2ØØ,ØØØ (assuming large numbers could be sold, to drive the price down).

B) A large array might be built from conventional off-the-shelf micro-computers, e.g., z8Øs, z8ØØØs, or whatever z8...Øs are developed in the future. It seems reasonable to assume that these will be of the same technologies, price and speed as the specially-designed CLIP chip. The differences would be that the CLIP chip would have 16 processors rather than only one, with the CLIP processors more appropriately designed for picture processing (e.g., with near-neighbor fetches in parallel and appropriate logical operations built into the hardware), but with much less memory and a much smaller total repertoire of hardware-embodied instructions associated with each CLIP processor. We might therefore expect a 16-fold decrease in costs from CLIP (which would come if and only if the appreciable costs of designing and fabricating the CLIP chip and building the total CLIP system were amortized over large numbers of successfully sold CLIP systems). In addition, the more appropriate design of the CLIP processors should further increase speed and power.

On the other hand, the larger memory and wider range of instructions (especially including arithmetic instructions) with the more conventional micro-processor would, when they were useful, lead to significant improvements on their side. And it might well be that in 5 or 10 years "conventional" microprocessors will be designed to do more and more parallel operations, or, possibly, come in an increasingly wider variety of less-conventional-from-the-perspective-of-1981 architecture. So let's just give CLIP the 16-fold improvement from its processors, and assume that the other factors cancel one another out. Therefore, if CLIP-like systems can be bought easily they will probably be preferable. If they are exotic, or must be designed and developed by the individual, the problems may easily be greater than the expected benefits. This suggests that it is important to make it as attractive as possible for companies to build and to market such systems, by developing as large as possible a group of potential users.

C) A somewhat (probably appreciably) smaller array might be built from bit-slice processor chips. These are, today, often used as building blocks for very fast and powerful large CPUs. They are relatively easily configured and constructed into processors, and, because they are micro-coded, they can be given a wide variety of carefully tailored machine language instructions. A typical processor might need roughly 20 chips (PICAP I has about 45), and cost, roughly $200 to $1000 (for raw hardware). If many processors were built the design costs per processor would go down appreciably.

But nobody appears to have built a system of this sort with more than 16 processors, or even contemplated one with more than 1000 or so. (Siegel et al., 1979; Bogdanowicz, 1977; Briggs et al., 1979) Probably $500 to $2000 per processor is a reasonable estimate for the cost in an array of from 50 to 200 processors built from off-the-shelf ICs. Such a system might well be 10 times faster than either of the above alternatives in its basic speed. It might gain appreciably more when doing arithmetic and, because of its microcode and potentially good ability to fetch information in parallel, might gain up to one more order of magnitude increase in speed.

This would appear to be a very interesting alternative when the size of the image to be processed is reasonably small. For example, a 16 by 16 array of bit-slice processors might scan a 96 by 96 array of pixels by having each processor scan a 6 by 6 sub-array. This might well be done at virtually the same speed as a 96 by 96 CLIP array. It would probably cost appreciably more. Its main advantage would appear to be its wider range of possible instructions and its great (potential) flexibility (if it can be microcoded and programmed appropriately) to effect a wide variety of processes, including more traditional MIMD network processes.

D) A very powerful and appropriately designed single scanner would be far slower than the three alternatives described above. Kruse's PICAP is an already built and tested example of such a system. It needs one microsecond per pixel (using a much faster technology than that of CLIP4, but that is quite reasonable since only one processor is involved). It will inevitably need increasing amounts of time as the size of the picture to be processed grows. For a 100 by 100 it may well need only 100 times as much time, gaining 100-fold in speed because of the good architecture and overall design features touched on above. But for a 1,ØØØ by 1,ØØØ it would need 1Ø,ØØØ times as much time.

Such a system might be extended by making the single processor into a pipeline of processors (as Sternberg has done in the Cytocomputer). But as the pipeline grows longer the need for more memory for intermediate results grows greater. Sternberg's 118 processors already appear to be hard to use completely, in every instruction. It seems possible that a hundred-fold increase can be got in this way (though possibly only at a very high cost in hardware), but probably little more.

E) An array of several specially designed processors, possibly including pipelining, appears to be an attractive alternative to the more conventional arrays envisioned in C), and especially in B). Here we see obvious trade-offs in terms of the power and cost of each particular processor. To give rough estimates: the PICAP processor might cost $1Ø,ØØØ; the Cytocomputer processor might cost $5Ø,ØØØ. A more conventional bit-slice processor might cost from $2ØØ to $2,ØØØ. There is a need for a great deal of empirical examination of the costs and benefits of different design trade-offs for such processors. Almost certainly a great variety of different individual processors, and total arrays and networks, should be built, and almost certainly there will be no single "right answer."

F) A series of arrays of the sort described above might be built, to serve as a giant pipeline each of whose processor stages was itself a very large array. This might take one of several forms:

1) The entire system of arrays might be built by simply dupli-

cating the single array. For example, 20 96 by 96 CLIP4s might be put in series.

2) A single array might be programmed to simulate such a series, by decomposing it into a number of sub-arrays of equal size, and shifting information, when appropriate, from one sub-array to the next. (This would be extremely wasteful of time in a conventional CLIP4 array, though changes in the interconnection architecture might overcome many of the problems.)

3) A single array might be used, but one that was re-configurable at the hardware level, under program control. Siegel et al.'s, 1979, proposed 32 by 32 MIMD-SIMD system can be reconfigured into different sub-systems whose sides are powers of 2, and Lipovski, 1977, is now building a 16-processor system that is reconfigurable at the hardware level, in order to demonstrate that its extra costs for switches and slower message passing can be kept reasonably small.

4) Rather than have all arrays the same size, arrays of different sizes might be used, where appropriate. This appears to me to be an attractive alternative for picture processing, since the very large array of raw information initially input to the system by the tv camera or other sensing device is successively abstracted and reduced as processing progresses. That suggests starting with a very large array but then transferring information to successively smaller arrays as they become appopriate, giving an overall pyramid structure.

5) Different kinds of processors, and different kinds of arrays, might well be appropriate at different layers of such a

system. It seems likely that the early "image processing" operations on near-neighbor cells (for which CLIP for example is especially well designed) can be handled by simpler processors that can be built into larger arrays. "Higher-level" processing might best be done by processors capable of adressing more global sets of information. At the most global level, when information has been greatly reduced, a single powerful processor might be most appropriate.

G) Each of the above possibilities has its own range of uses for which it is best suited, and its own set of limitations. This strongly suggests that a judicious combination of several of the above alternatives might produce a network of mixed resources that could take advantage, if multi-programmed over a suitable mix of user programs, of each of its sub-systems strengths.

Summary and Conclusion

The particular configuration we have been considering and (tentatively) planning to build is only one of several possible architectures for parallel arrays and networks that promise orders of magnitude increases in speed and power for picture processing and other cognitive tasks. Our choices are dictated by a variety of economic and practical issues, and it seems important that a number of other, competing, architectures be implemented.

It seems compelling that a general system for the perception of scenes of moving objects must be orders of magnitude faster (and more powerful) than any conceivable single-CPU computer. And it seems much more sensible to build a system that has many processors working in parallel, rather than try to push as much power as possible out of one serial machine. Human perception uses millions of processors arranged in a parallel-serial layered network of arrays, and handles perception in a surprisingly small serial depth of 25 or fewer processes to a few hundred at most.

A variety of different sources for speeding up processing are available to us, and many of these can be combined. But the only way in which we can continue to speed up our systems is to add more physical processors. This paper examines some of these sources of potential speed-ups, and some of the network and array architectures that appear the most promising ways to exploit them. It also describes and discusses the particular system that we hope to build: a large array of simple processors, a smaller network-array, and a powerful serial computer.

Probably the best way to view such a system is as a network of diverse resources with which to learn how to explore and extend such networks - by simulating, designing, and building them, then programming them to see how well they work. This raises many major problems, in developing operating systems and languages to expedite the smooth use of the total system, handling different kinds of nodes in the total network, and developing appropriate algorithms and programs. But these are extremely interesting, and important, problems in themselves. To the extent they are not solved, or handled well, such a system will still be usable, in its separate components, and in the limited combination of these components for which it has been expressly designed.

References

Batcher, K.E., Architecture of a massively parallel processor, Proc. 7th Annual Symp. on Computer Arch., ACM, 1980, 168-174.

Bogdanowicz, J. F., Preliminary Design of a Partitionable Multimicroprogrammable Microprocessor System for Image Processing, Tech. Report, EE 77-42, School of Electrical Engineering, Purdue Univ., 1977.

Briggs, F., Fu, K. S., Hwang, K. and Patel, J., PM4 - a reconfigurable multimicroprocessor system for pattern recognition and image processing, Proc. AFIPS NCC, 1979, 255-265.

Despain, A.M. and Patterson, D.A., X-tree: a tree structured multi-processor computer architecture, Proc. Fifth Annual Symp. on Computer Arch., April, 1978, 144-151.

Douglass, R. J., Recognition and depth perception of objects in real world scenes, Proc. IJCAI-4, 1977, 656.

Douglass, R. J., A computer Vision Model for Recognition, Description, and Depth Perception in Outdoor Scenes. Unpubl. Ph.D. Diss., Computer Sciences Dept., Univ. of Wisconsin, 1978.

Duff, M. J. B., CLIP4: a large scale integrated circuit array parallel processor, Proc. 3d Int. Joint Conf. on Pattern Recog., 1976, 4, 728-733.

Duff, M. J. B., Review of the CLIP image processing system, Proc. National Computer Conf., 1978, 1055-1060.

Flanders, P.M., Hunt, D.J., Reddaway, S.F., Parkinson, D., Efficient high speed computing with the Distributed Array Processor, In: *High Speed Computer and Algorithm Organization*, New York: Academic Press, 1977, 113-128.

Fuller, S. H., Price/performance comparisons of c.mmp and the pdp-10, *Proc. Third Symp. on Computer Arch.*, 1976, 195-202.

Goodman, J.R. and Sequin, C.H., Hypertree, a multiprocessor interconnection topology, submitted for publication, 1981.

Hanson, A. R. and Riseman, E. M., Pre-processing cones: a computational structure for scene analysis, COINS *Tech. Rept. 74-7*, Univ. of Mass., 1974.

Hanson, A. R. and Riseman, E. M., Visions: A Computer System for Interpreting Scenes, In *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (Eds.), New York: Academic Press, 1978, 303-334.

Klinger, A. and Dyer, C., Experiments on picture representation using regular and decomposition, *TR Eng. 7497, UCLA*, 1974.

Kruse, B. The PICAP picture processing laboratory, *Proc. 3d Int. Joint Conf. on Pattern Recog.*, 1976, *4*, 875-881.

Kruse, B. Experience with a picture processor in pattern recognition processing, *Proc. National Computer Conf.*, 1978.

Kruse, B., Danielsson, P.E., and Gudmundsson, B., From PICAP I to PICAP II, In: *Special Computer Architectures for Pattern Processing*, K.S. Fu and T. Ichikawa (Eds.), New York: CPR Press, 1980.

Levine, M. D., A knowledge-based computer vision system, In:

Computer Vision Systems, A. Hanson and E. Riseman (Eds.), New York: Academic Press, 1978, 335-352.

Levine, M. D. and Leemet, J., A method for nonpurposive picture segmentation, Proc. 3d Int. Joint Conf. on Pattern Recog., 1976, 494-498.

Lipovski, J., On a varistructured array of microprocessors, IEEE Trans. Computers, 1977, 26, 125-138.

Reddaway, S.F., DAP - a flexible number cruncher, Proc. 1978 LASL Workshop on Vector and Parallel Processors, Los Alamos, 1978, 233-234.

Reddaway, S.F., Revolutionary array processors, In: Electronics to Microelectronics, W.A. Kaiser and W.E. Proebster (Eds.), Amsterdam: North-Holland, 198Ø, 73Ø-734.

Siegel, H.J., et al., PASM: A Partitionable Multimicrocomputer SIMD/MIMD System for Image Processing and Pattern Recognition, School of Electrical Engineering TR-EE 79-4Ø, Purdue Univ., West Lafayette, 1979.

Sternberg, S.R., Cytocomputer real-time pattern recognition, paper presented at Eighth Pattern Recognition Symp., National Bureau of Standards, April, 1978.

Sternberg, S.R., Language and architecture for parallel image processing, Proc. Conf. on Pattern Recognition in Practice (E.S. Gelsema and L.N. Kanal, Eds.), Amsterdam: North-Holland, 198Ø.

Sullivan, H., T. Bashkov, and D. Klappholz, A large scale, homogenous, fully distributed parallel machine, In: Proc. Fourth Annual Symp. on Computer Arch., 1977, 1Ø5-124.

Swan, R.J., S.H. Fuller and D.P. Siewiorek, Cm* - A modular, multi-microprocessor, Proc. AFIPS NCC, 1977, 637-663.

Tanimoto, S. L., Pictorial feature distortion in a pyramid, Computer Graphics Image Proc., 1976, 5, 333-352.

Tanimoto, S. L., Regular Hierarchical Image and Processing Structures in Machine Vision, In: Computer Vision Systems, A. R. Hansen and E. M. Riseman (Eds.), New York: Academic Press, 1978, 165-174.

Tanimoto, S. and Klinger, A. (Eds.), Structured Computer Vision, New York: Academic Press, 198Ø.

Uhr, L., Layered "recognition cone" networks that preprocess, classify and describe. IEEE Trans. Computers, 1972, 21, 758-768.

Uhr, L., A model of form perception and scene description, Computer Sciences Dept. Tech. Rept. 176, Univ. of Wisconsin, 1974.

Uhr, L., "Recognition cones" that perceive and describe scenes that move and change over time, Proc. Int. Joint Conf. on Pattern Recog., 1976, 4, 287-293.

Uhr, L., "Recognition cones" and some test results. In: Computer Vision Systems (A. Hanson and E. Riseman, Eds.), New York: Academic Press, 1978, 363-372.

Uhr, L., Computer Arrays and Networks: Algorithm-Structured Parallel Architectures, in progress, 1981.

Uhr, L. and Douglass, R., A parallel-serial recognition cone system for perception, Pattern Recognition, 1979, 11, 29-40.

Uhr, L, Lackey, J. and Thompson, L., A 2-layered SIMD/MIMD Parallel Pyramidal "Array/Net", Computer Sciences Dept. Tech. Rept., Univ. of Wisconsin, 1980.

Wittie, L.D., Efficient message routing in mega-micro-computer networks, Proc. Third Annual Symp. on Computer Arch., New York: IEEE, 1976.

Wittie, L.D., MICRONET: A reconfigurable microcomputer network for distributed systems research, Simulation, 1978, 31, 145-153.

Wittie, L.D. and van Tilborg, A.M., MICROS, a distributed operating system for MICRONET, a reconfigurable network computer, IEEE Trans. Computers, 1980, 29, 1133-1144.