Parallel, Hierarchical Software/Hardware Pyramid Architectures

by

Leonard Uhr

Computer Sciences Technical Report #646

June 1986

Parallel, Hierarchical Software/Hardward Pyramid Architectures

Leonard Uhr

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706

Technical Report #646

June 1986

This paper will be published as the first chapter in the proceedings of the NATO Workshop on Pyramids held in Maratea, Italy, May 1986 (V. Cantoni and S. Levialdi, Eds., *Pyramid Image Processing Systems*, Berlin: Springer-Verlag).

PARALLEL, HIERARCHICAL SOFTWARE/HARDWARE PYRAMID ARCHITECTURES

Leonard Uhr
Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706, USA

Introduction

This paper examines pyramid-structured software/hardware systems, both programs and multi-computers. It explores how efficient parallel computers can be designed. It poses the extremely difficult problem of perception, and briefly surveys the major alternative approaches. It examines the basic structures with which pyramids can be built, and describes pyramid multi-computers. It sketches out how pyramid hardware can be used to execute programs, with special emphasis on the "recognition cone" structures being developed to model living visual systems. It briefly suggests how pyramids can be augmented and embedded into larger software/hardware structures. In addition, it gives a short history of massively parallel hierarchical pyramid structures.

Serial and Parallel Computers, General-Purpose and Specialized

The classical serial 1-CPU Von Neumann computer is capable of scanning over any possible program graph (simply because any graph can be decomposed into a set of 1-node graphs). But it is guaranteed to take the longest possible time.

The only way the single-CPU Von Neumann computer that has for the past 40 years been such a tremendous success can be substantially further improved in power and in speed is to combine large numbers of individual computers into parallel networks. This is because the VLSI technologies that are about to pack increasing millions of transistors onto each tiny silicon chip are fast approaching the speed of light.

A great variety of different types of multi-computers are being investigated, designed, and built. But it is not at all clear whether it will be possible to design and build multi-computer networks that will execute any and all programs with high efficiency and sufficient speed. It may well be necessary to settle for reasonable trade-offs between power, generality, efficiency, speed, and cost-effectiveness.

True Data-Flow Hardware That Is Isomorphic to the Program's Structure

Possibly the ideal is to map the program into an isomorphic data-flow hardware. This would be equivalent to building the appropriate finite state automaton that was equivalent to that program. Brains appear to be systems of this sort, since they are networks of neurons through which data flow. However, it is not at all clear that such a system could be generally usable over a wide range of different programs.

There are a number of other interesting possibilities. They are just beginning to be investigated, and much work will be needed to develop any of them.

Systems that Flow Data Through a Smaller Sub-Graph

Rather than build hardware whose structure is isomorphic to the entire program graph, a much smaller hardware graph can be constructed. Now that graph can be used to scan over the larger program graph. This is exactly what is today done with the single-CPU computer. Since it is a 1-node graph any program graph can be decomposed into it, for successive processing. (Note that this gives the slowest possible execution, since the single processor can do only one thing at a time, serially.)

In order to increase speed, the number of nodes in the hardware graph must be increased substantially. It is not known whether this is possible in the general case, although there is one striking commercial success - the linear pipeline of processors that give today's "supercomputers" their great power at the vector processing so common in scientific computing.

Systems that Use a General Topology that Can Execute ALL Programs Efficiently

The program can be represented as a graph, and the multi-computer can be represented as a graph. The problem becomes one of finding a multi-computer graph into which all program graphs can be mapped with reasonable efficiency. Since it is not known what different graph topologies are needed for different programs of interest, it is impossible to judge whether such a system can be realized. It may well be that there are a small number of basic structures that programmers could use to compose large programs.

Systems that Use Switches to Reconfigure Among Appropriate Topologies

Switches might be incorporated into the hardware network, so that it could be reconfigured to handle each different program input to it. A generally reconfigurable system is in theory possible. But the cost in switches would probably be too great. (These costs increase as the network grows larger, and will quickly dominate the cost of the entire system.)

Networks that Contain Suitably Specialized Regions

Several different structures can be combined together, to build the larger network. Reconfiguring switches can be used to link the individual structures together in a variety of different ways, as directed by the program.

Some General Issues

Different programs will, inevitably, be of different sizes and with different amounts of parallelness. Therefore the only way to use a general-purpose multi-computer efficiently is to multi-program, so that different programs will use otherwise idle resources. This further complicates the problem.

However, it may well be that percent utilization is unimportant. Transistors and chips get cheaper and cheaper. Increasingly important will be speed and power.

This paper concentrates on program-structured specialized systems, along with attempts to make them more general. The perception program motivates and suggests the hardware.

The Perception Problem Posed, and its Extreme Difficulty

The perceptual recognition and description of complex real-world objects is an extremely difficult problem. Probably half of the many billions of neurons in brains of higher mammals are involved with perception. The speed at which perception takes place is incredibly fast.

The Massively Parallel and Shallowly Serial Perceptual System

The human visual system has about 10,000,000 cones and 100,000,000 rods in each eye. The cone system is sensitive to color and shape. The rod system is sensitive to change, whether of intensities in the spatial domain or of position in the temporal domain. Cones are concentrated in the central fovea, where shapes are perceived. Rods extend out to the periphery, and thus serve as an early warning system.

The living perceptual system is capable of recognizing and describing complex scenes of complex objects in from 30 to 800 milliseconds. But the basic cycle time of a neuron - that is, the amount of time it takes to bridge the synapses over which neurons fire other neurons - is roughly 1 or 2 milliseconds. This means that the serial depth of the perceptual system is only a few hundred steps at most, and possibly only a few dozen. The visual system appears to be organized in layers of neurons that converge and diverge information, as they transform that information, moving inward from retina to primary and then to a number of secondary

visual areas, and then other parts of the cortex (see Uhr, 1980, 1986c for more extended examinations of these issues).

The System Must Recognize and Describe Scenes of Complex Moving Objects

To recognize, describe, and track real-world objects as they move about and interact with one another in real time, the perceptual system must be able to process each input (e.g., each frame in a succession of TV images) in only 30 milliseconds or so. This means that the serial depth when processing moving images is especially shallow.

The System Must Handle Enormous Amounts of Information in Complex Scenes

Real-world scenes typically contain a number of different objects. Psychological experiments indicate that we human beings are able to recognize objects and their distinguishing and salient characteristics when there are dozens in the single scene. This is a largely non-conscious process that takes place in a few hundred milliseconds.

The System Must Recognize Highly Structured Objects Over Unknown Variations

Each object is complex, with a large number of individual details, some of which are of importance and must be noticed. Each instance of each object can vary in a wide variety of ways. The linear transformations (translation, magnification, rotation) are only the simplest. Far more important, and more difficult to handle, are the unknown non-linear transformations effected by atmospheric distortions, muscles that twist, emotions, and aging. Consider the many ways that a face can change as it smiles, frowns, or looks tired or discouraged.

The Major Alternative Software Approaches

A large number of programs have been written to recognize objects (variously called image processing, pattern recognition, and computer vision). Most of these fall into one or another of the following general types.

A) Programs that Combine Results of Independent Procedures (Feature Detectors)

In the simplest case, the image is processed with a number of procedures (often called feature-detectors), and their results combined into a decision as to what might be there.

A highly parallel 2-dimensional array structure is often used to realize such systems.

When executed on hardware arrays, these systems typically use a variety of simple, usually local, features. (Non-local features can take extremely long and inefficient sequences of operations.)

These systems are usually considered to be bottom-up, first applying feature detectors, then combining their results, and finally choosing among alternative possibilities. But they can also apply feature-detectors in a top-down manner, or be given a top-down component.

They can be extremely fast. But they are relatively weak, and suited only for the recognition of small, simple objects that vary little.

B) Programs that Assess Syntactic-Like Structures Over Primitives

A number of programs have been developed that attempt to use 2-dimensional extensions of grammars, giving tree-like structures. However, there are major unsolved problems when trying to generalize grammars that have been developed for linear strings to handle 2-dimensional objects. In a linear string only one substring can be linked (concatenated) to the previous substring. That is, each substring touches only two other substrings, one on either side. In sharp contrast, in a 2-dimensional space a thing (feature or object) might join to a potentially infinite number of other things.

Such syntactic recognition systems usually expect and can handle only images made up of perfectly connected simple primitive curves. This means that they work only on carefully pre-processed or toy data (e.g., line drawings of simple objects).

C) Programs that Match Stored Models with Appropriate Regions in the Image

Probably the most commonly proposed type of system for "high-level" computer vision is one that attempts to match models of each possible object in a top-down manner. Typically, this kind of system serially matches models (commonly called "frames," or "schemata") to sub-regions of the image. The model is, essentially, a graph - and, for real-world objects, this will be an extremely complex graph. The image must be processed, usually in a bottom-up manner, until it is converted into a similar graph. This usually means that its edges, angles, contours, and regions must all be identified and correctly linked to one another.

This is extremely difficult (if not impossible - consider the prefectly recognizable images that can be constructed by randomly erasing most of the regions and edges). In any case, the system is, essentially, faced with a graph-matching problem. (Note that in the general case this is an NP-complete problem, since the graph that models the object must be matched with a sub-graph of the graph into which the image has been transformed.)

This kind of system is typically given only a few models, and matches them separately, one by one. It is usually considered to be top-down. Such systems can handle only a few ob-

jects, and are inevitably slow.

D) Programs that Hierarchically Examine Successively Larger Structures

Highly parallel, logarithmically serial "pyramid/cone" structures appear to offer real hope of overcoming the problems of other types of vision systems.

They can make use of local operations that are executed in parallel everywhere. This means that they can be quite general-purpose and robust, looking for and being able to respond to, unanticipated objects. They can combine bottom-up and top-down processing, moving upward, downward, and laterally through the pyramid. Therefore they can be looking for the most expected objects, while at the same time monitoring the environment for the unexpected. And they can keep gathering information and testing tentative hypotheses until enough information has been accumulated and assessed to come to a decision.

They can combine and apply many models, that are all embedded in the pyramid, in a relatively efficient way. If each model is turned into a tree-like graph, with the proper number of levels, these graphs can be embedded into a pyramid so that each hardware level executes about the same number of instructions. That is, the model can be decomposed into a hierarchy of procedures that are embedded into hardware to load-balance across levels. Now all the different models of different objects can be squashed together. Wherever they have nodes (that is, processes) in common these can all be combined, so that the process need be executed only once. This will often be the case, especially at the lower levels, where the same gradient, edge, texture, and region detectors will be common to and appropriate for recognizing many different objects. They therefore appear to be capable of handling many objects, potentially with speeds great enough to handle the real-time perception of moving objects.

Developing Programs and Architectures for Fast, Efficient Execution

A large variety of different network topologies have been developed for multi-computers. An enormously larger variety is possible, since the network's underlying topology can be any possible graph.

The program graph must flow smoothly through the computer structure. That is, the flow of processes that makes up the program must be decomposable into the graph or graphs that make up the multi-computer that will execute those processes.

This means that for perception, the program must examine large 2-D images, on the order of 512 by 512, 1,024 by 1,024, or even larger. For scenes of moving objects, each image must be processed in only 20 to 50 milliseconds, or even less.

In the general case the perceptual system can't anticipate what might be there, or where. It

must look everywhere, and therefore must have a massively parallel bottom-up component to its processing.

The following begins to describe the several major types of multi-computer architectures that have been developed. It emphasizes structures that are most appropriate for perception, presenting them in a natural progression that leads up to an examination of pyramids. The section below on augmenting pyramids will briefly explore Then we will examine additional architectures with which pyramids can fruitfully be combined.

The Conventional 1-CPU Serial Computer (SISD)

The classic single-CPU serial computer is, essentially, a 1-node graph.

Any program graph can be flowed through it, but serially. Except for very small and simple problems, or toy demonstrations, this is much too slow for perception.

1-Dimensional Strings: Pipes (MISD), Rings, Arrays (Synchronized SIMD)

Several individual computers can be linked together to form a string. That is, each computer links to two other computers (except for the first and the last, which each link to only one). Systems of this sort have been built with as many as 1,024 processors.

Possibly the simplest way to use such a system is as a linear pipeline through which information is streamed. Each computer in the pipe repeatedly executes the same process, but on different sets of information - much as each worker in an assemblyline repeatedly executes his assigned task. Two major types of specialized pipelines have been designed, built, and successfully commercialized.

The first type is the number cruncher "super-computers" like the Crays and Cybers (see Riganati and Schneck, 1984). They are given pipelines of 10 or so very powerful 64-bit floating point processors to operate on vectors of numbers.

The second type has been specialized for image processing. These include PICAP (Kruse, 1976) and the Cytocomputers (Sternberg, 1978; Lougheed and McCubbrey, 1985), and also a number of commercial pipelined scanning arrays like the deAnza and Vicom. Each has 1 to 1,024 8-bit processors, specialized to do 8-bit fixed-point arithmetic on grey-scale and color images. Additional hardware allows for very fast scanning through the 2-dimensional array that contains an image - whether the raw image, or a transformed representation.

When a P-node pipe is kept full and busy, the system can get up to P-fold speed-ups, minus the time needed to fill and to empty the pipe.

The last node of a pipeline of this sort can be linked back to the first, forming a ring. The system can now cycle and pipe the last processor's results back around into the first, and this cycling can continue potentially without limit.

Alternately, the 1-dimensional line can be used to process a 2-dimensional array of data, by piping each row (or column) of the 2-dimensional array through the appropriate different processor in the line. For example, the PIXIE-5000's 1 by 1,024 array of 1-bit processors can process a 1,024 by 1,024 array in 1,024 instructions (taking 80 microseconds in toto).

2-Dimensional Arrays (Most are SIMD, But They Needn't Be)

The multi-computers that have actually been built with the largest number of processors (by far) are the 2-dimensional SIMD arrays. These are especially appropriate for perception, where the basic input is the very large 2-dimensional image.

The 2-dimensional array can directly process 2-dimensional data. It can also process larger arrays, and N-dimensional arrays.

It is important to mention a much smaller 2-dimensional array of each far more powerful 64-bit floating point processors - the number cruncher ILLIAC-IV (Barnes et al., 1968). This was the largest and most powerful computer built during the early 1970s.

A number of 2-dimensional arrays with a much larger number of processors have been built more recently. These include the 96 by 96 CLIP4 (Duff, 1976), the 64 by 64 DAP (Reddaway, 1978), and the 128 by 128 MPP (Batcher, 1980). Each computer links directly to its 4 or 8 nearest neighbors. All execute the instruction broadcast by a single controller, and therefore run in SIMD mode. All use relatively simple (but general-purpose) 1-bit processors, with 4 to 8 processors on each chip.

This kind of array can be given more and more computers, potentially without limit. For example, a 1,024 by 1,024 TV image can be input to a 1,024 by 1,024 array of computers, so that each contains one cell of the image. Now each processor can compute functions of local regions of information surrounding the spot stored in its memory. Successive functions will therefore compute successively more abstract results about this local region. Several processors can be used, if desired, to process each region. For example, each of 8 different oriented edges could be looked for at the same time, if 8 processors were assigned to each spot.

An array of processors smaller than the image array can iterate serially through the larger array. For example, a 1,024 by 1,024 array of computers can process a 4,096 by 4,096 image array by storing a 4 by 4 subarray in each computer, and serially iterating through the 16 spots. Alternately, each of the 16 1,024 subarrays can be processed in turn.

Arrays have very good local connectivity. When they are given a single controller this is nicely combined with very good global coordination, so that a massively large number of local operations can all be executed at the same time. For example, the program can in one step look for a local feature like a gradient, edge, color, or texture everywhere, in many thousands or millions of places. When several controllers are used, one for each subarray or reassignable under program control, the array can execute several different processes, as appropriate,

in different regions of the scene.

Arrays can become quite slow when they must make global assessments, or pass information over great distances. This is so because the only way that an array can compute a function of information stored at several different locations is to shift all the pieces of information to a single processor. To do this a serial sequence of shifts must be effected. In the worst case this can mean, in an N by N array with direct links from each processor to its 4 square neighbors (to North, South, East and West), up to 2N shifts for each piece of information.

Pyramid of Arrays (SIMD-MIMD)

A pyramid can most simply be described as a stack of successively smaller arrays that are linked together by a tree. For example, a 16 by 16 array might have each 2 by 2 subarray linked to the corresponding cell of an 8 by 8 array of parents. These similarly link to a 4 by 4 array, and this linking scheme continues to a 2 by 2 and a 1 by 1 (apex) array.

Pyramids can be designed in a large number of different ways (see Uhr, 1984, 1986c). For example, pyramids with square bases can converge 2 by 2, 3 by 3, or N by N, or N by M then M by N. They can have rectangular or hexagonal bases, and converge appropriately. They can converge different amounts at different levels. They can have overlap, with each node linked to several parents, or no overlap, with each node linking to only one parent. Since information can be passed laterally through the array links at each layer, overlap can be simulated quite easily by a non-overlapped pyramid. Similarly, an overlapped pyramid does not need links to siblings, which can be reached by moving down to children, and then back up.

Each array in the pyramid can execute all array operations efficiently. In addition, passing information up and down between arrays needs greatly reduced logarithmic distances, rather than the linear distances needed within an array. For example, whereas a 1,024 by 1,024 array needs up to 2,046 shifts to send information, when augmented with a pyramid (which adds less than 1/3d more computers) it needs at most 20. This is a crucial difference in terms of the extremely small amount of time available for real-time perception of real-world objects.

The major purpose of a pyramid is not to communicate, but to compute. Rather than simply pass information, or average and compress information, moving up the pyramid, the processors can be programmed to process and transform that information in any way desired.

Using Pyramid Software/Hardware Systems

Pyramid-like structures can be built and used in several different ways.

Possibly the simplest is the quad-tree, which is a tree superimposed over an array of infor-

mation. (Note that this is not a pyramid, since it has neither lateral links to siblings or overlap that allows for communication via children.) Wherever a parent node's children all contain the same value that value can be passed up to the parent and the children eliminated. The resulting quad-tree will (to the extent that images do not have many high frequency variations) be much reduced in size, yet it can be used to reconstruct the image exactly.

Possibly the simplest way to use a true pyramid is to successively reduce the image by averaging. This can serve for data compression. It also makes possible a quick global assessment at a higher level, so that the program can then zoom down and look in more detail at particular regions that it judges may be of interest. Or the program can apply feature-detectors of the appropriate frequency to each of the different levels.

Rather than average the raw image, the program can apply a feature detector, e.g., for gradients, edges, or textures, and then successively average their output.

A number of relatively efficient pyramid algorithms have been developed for such operations as region growing, blob counting, median filtering, and contour and region linking (Cibulski and Dyer, 1984; Miller, 1984; Stout, 1983, 1985; Tanimoto, 1983). These make use of the array links when appropriate, and of the up and down tree links to span larger distances. It is important to emphasize that the converging tree structure moving toward the pyramid's apex, which on the one hand gives the pyramid its good logarithmic information-passing distances, can cause severe bottlenecks when too much information must be passed. For example, pyramids, because of their tree structure, are poor for sorting and permuting information.

Using Pyramid Multi-Computers to Execute "Recognition Cone" Processes

The approach that I and my associates have been taking to using pyramids (Uhr, 1971, 1973, 1975; Uhr and Douglass, 1979; Schmitt, 1981; Li and Uhr, 1985a, 1985b, 1986) attempts to model what living perceptual systems appear to do as they successively process and converge information, leading toward recognition. The overall pyramid structure is an extraction from and simplification of the much larger system, which also has parallel arrays, diverges information, and fans out into the semantic memory and associative areas.

The massively parallel and shallowly serial pyramid structure offers many possibilities for parallelizing and speeding up perceptual processes, and for decomposing the extremely complex procedures needed to perceive real-world objects into a hierarchical structure of simple processes. The basic process is that of a probabilistic threshold element similar to a neuron that fires when enough neurons fire into it.

The model of each object to be recognized is decomposed and converted into a hierarchical tree of suitably simple processes, and then embedded into the pyramid. Each of these processes is made as simple and as local as possible. The many nodes that different object-

models have in common (particularly at the lower levels) are combined into one, so that all objects can be looked for in parallel.

The system can be primed in advance to look for particular objects, or (when processing a sequence of images of a scene of moving objects) it can use what information it has uncovered to help direct its search. Thus a top-down component can be combined with the bottom-up processing that is driven by the image input to the pyramid's base.

For example, structures of transforms of the following sort have been embedded into a 512 by 512-based 2x2 converging 9-layered pyramid.

1) Smooth; Eliminate Specks of Noise.
2) Get Gradients.
3) Get Short Edges, Colors, and Simple Textures.
4) Get Longer Edges, Simple Curves, More Complex Textures, Small Regions.
5) Get Angles, More Complex Curves, Larger Regions.
6) Get Simple Enclosures, and Simple Objects.
7) Get More Complex Objects.
8) Get Still More Complex Objects, and Groups of Objects.
9) Get Still More Complex Objects, and Groups of Objects.

Such systems have successfully recognized real-world objects like neurons, trees, windows, doors, roofs and houses.

A complete program may entail several phases of processing that move up, down, and around the pyramid. The system can be programmed to look downward for objects that are already stored in a "Lookfor" list, as well as do bottom-up processing of a new image. (No matter how strong is the top-down component, a general vision system must always be alert for new, unanticipated but possibly important, objects.) This bottom-up processing will begin to imply new objects that might be present. This in turn starts up new top-down processes to gather the additional information needed to recognize those objects. The system can keep cycling in this way until enough objects have been recognized, or no time is left.

When the system must process a continuing stream of images of objects in motion, there may well be little time for anything but bottom-up processing (remember, to model living visual systems there is time for only a few dozen, or a few hundred, instructions). Therefore it seems best to augment the basic pyramid with additional hardware, to further parallelize and to simultaneously execute bottom-up and top-down processes.

The Need to Augment the Pyramid

There are several reasons why it seems desirable to augment a pyramid, either by adding internal hardware or by embedding it in a larger network. SIMD pyramids (this is also true

for arrays) become inefficient when all processors are not working fruitfully. The tree bottlenecks cannot always be avoided by commensurate reductions of information. Different types of processors may be indicated for different purposes (e.g., to match masks, combine weights, choose among alternatives). It might be desirable to combine several pyramids, for example to handle different sensory modalities. Finally, the perceptual system must be combined with the larger cognitive system.

There are a number of interesting possible augmentations (see Pfeiffer, 1985; Tanimoto, 1985; Uhr, 1983, 1985a, 1985b, 1986a, 1986b). For example, processors can be made reconfigurable to, e.g., 1 64-bit (possibly organized to handle in parallel an 8 by 8 window operation), 4 16-bit, 8 8-bit, or 64 1-bit processors. Several processors can be placed at each cell. Several pyramids can be linked together. The pyramid can, wherever appropriate, be given more communication bandwidth, processor power and/or memory size. Processors can be reassignable under program control to top-down and/or bottom-up flow.

An MIMD network can be linked to the pyramid, at its base, apex, higher layers near the apex, or both high and low layers (now the MIMD network can serve for communication and feedback at the same time that it transforms and processes information). There are a large number of potentially attractive MIMD networks that might be used (see Uhr, 1984, 1986c).

Several networks of this sort can be built to give the programmer the impression that all processors share a common memory. The simplest link all computers via a common bus, ring or crossbar switch. But more than a few dozen computers linked in this way will overload the bus or ring, or need an excessively expensive crossbar. An NlogN switching network can be used to link larger numbers of computers - up to a few hundred, or possibly (but with rapidly increasing costs for the switching network) even a few thousand.

Point-to-point linked networks can be built in a great variety of topologies, since any connected graph can be used. Among the most popular today are N-dimensional hypercubes, trees, and several bus-based or crossbar-based clusters linked together. Such systems have the major advantage that they can, potentially, be built to any size. But there can be difficulties in passing information through a path of intervening processors.

There are a number of other potentially more attractive topologies that have not yet been used. There are a variety of compounding operations that appear to be better than N-cubes (which are compounds of N-1-cubes). A tree that is carefully augmented, usually at each leaf (which in a tree has only one link, to its parent), can be given substantially more computers within a given diameter (the shortest longest distance between nodes) and degree (the maximum number of links to each node). Good constructions of this sort include De Bruijn graphs and Hypertrees. There are also a variety of small graphs with good properties, including the optimally dense Moore graphs, that can be used individually or compounded into larger structures. There are almost certainly many other good constructions that have not even been discovered.

A Short History of Massively Parallel Hierarchical Structures (Pyramids)

The following, as requested by the editors of this volume, briefly sketches a bit of the history of pyramids, including data-structures, structures of software processes (algorithms and programs), and hardware architectures.

The first attempts to develop pyramid-like systems were in the form of software programs and algorithms. Much more recently, people have begun to design and build appropriate pyramid hardware.

Programs that Simulate Pyramid Structures

Kelly (1971) described a system that reduced (by averaging) an image up to a much smaller array, and then used this to determine which parts of the image to process in detail. It thus took a top-down approach to processing.

Klinger (1971, 1974) developed techniques for using quad-tree structures to represent images. Klinger and Dyer (1976) extended systems of this sort and examined how well they worked in actual experimental situations.

Uhr (1971, 1972) described hierarchically converging layered "recognition cone" systems that looked everywhere in a large array for successively more global structures of information. The transforming procedures used were motivated by the threshold element-like neuron, and had much the form of IF...THEN... production rules augmented with weights, thresholds, and 2-dimensional structural relations. An example of such a system was programmed in SNOBOL and given enough transforms to recognize a few simple objects. Later SNOBOL programs (Uhr, 1973, 1976) demonstrated how cone/pyramid structures could be used to describe as well as recognize objects in static scenes, and to begin to handle objects as they move about in real time.

Hanson and Riseman (1974, 1976, 1978) developed "preprocessing cones" that are used primarily for lower levels of semantically directed vision systems.

Tanimoto (1976, 1978) developed pyramids as data structures and, potentially, hardware structures. He was probably the first person to design a hardware pyramid (see below), and a language for describing and programming pyramid structures (1983a, 1984). He has also worked extensively on pyramid algorithms (e.g., 1983b).

Bajcsy and Rosenthal (1975, 1980) examined how to focus attention using hierarchical systems. Sloan (1977), working with Bajcsy, developed a multi-level hierarchical system that applied radically different types of processes at each level.

Levine and Leemet (1976) and Levine (1978, 1980) developed a pyramid program to recognize real-world objects using several different pyramids, each containing appropriate simple features (e.g., color).

Kruse (1976, 1980) used multi-resolution techniques within a pyramid structure to program his PICAP pipeline image processor to zoom in to regions of special interest. This approach was taken in programs to recognize finger prints and to examine circuit boards.

At about the time that Dyer went to Maryland, the group of vision researchers headed by Rosenfeld began to study pyramids (see, e.g., Rosenfeld and Vanderbrug, 1977; Davis and Rosenfeld, 1978, Rosenfeld, 1983). A large number of research papers from Maryland have explored pyramids, chiefly examining multi-resolution (Burt, 1984), quad-trees (Dyer et al., 1980; Samet, 1980), and linking of regions or feature elements (Burt et al., 1981; Pietikainen and Rosenfeld, 1981; Cibulski and Dyer, 1984).

Dyer (1979) also began to investigate pyramids from a theoretical point of view, and more recently to develop pyramid algorithms and programs (e.g., Neveu, Dyer and Chin, 1985). More recently, Stout (1983, 1985) and Miller (1984) have greatly extended this kind of theoretical analysis.

Uhr (1978), Uhr and Douglass (1979), Schmitt (1981), and Li and Uhr (1985a, 1985b, 1986) developed programs to recognize real-world images of houses, neurons, and other objects. These programs begin to extend cone/pyramid systems to use successively more global structures at higher semantic levels, and to combine bottom-up and top-down processing.

Several recent books (Tanimoto and Klinger, 1980; Rosenfeld, 1984; Uhr, 1987) have collected research on pyramids.

The essence of pyramid perception systems lies in the massively parallel (hence extremely fast) application of a converging hierarchy of micro-modular (hence relatively local) operations. Several of the most interesting and the most powerful computer vision systems that are not ordinarily thought of as using pyramids have many of these properties, for example, Tenenbaum and Barrow (1976), Marr (1982), Ballard (1985).

The Design and Construction of Pyramid Multi-Computers

To achieve the great speed that massively parallel hierarchical converging structures potentially offer, specialized pyramid hardware architectures are highly desirable, if not necessary.

Hardware designs for pyramids were first developed by Dyer (1981, 1982), Tanimoto (1981) and Uhr (1981, 1984). Several hardware pyramids are presently being built.

Tanimoto (1983a) is starting to build a large pyramid with each processor linked to 8 siblings, 4 children, and 1 parent. He has also simulated this system, and has designed a language that allows the programmer to work closely with the pyramid structure.

Cantoni et al. (1985) are working with a group of seven Italian Universities and several industrial firms. They have designed and are fabricating a chip that contains 1 parent and 4 children, and are investigating how to make the system fault tolerant.

Schaefer (1985), using MPP chips (each with 8 processors and a fast shift register), has completed a pyramid with a 16 by 16 base.

Merigot et al. (1985) have designed and are beginning to build a pyramid with the topology of a binary tree. This allows convergence at each level to one half (rather than one quarter) the previous level's processors.

All these systems use custom VLSI designs with several processors on each chip. All use 1-bit processors of the sort used by the CLIP, DAP and MPP arrays. All use a single controller for each layer (some may use a single controller for all layers).

Handler and his associates (Handler, 1975; Bode et al., 1985) have been building much smaller pyramids of much more powerful independent computers. Each computer has its own controller, so that the system runs in MIMD mode. These systems have been designed to handle numerical problems. But they should be quite useful for image processing and pattern recognition. Potentially, an MIMD pyramid can offer greater flexibility in applying different processes at different regions of the image, and allocating more resources where appropriate.

Closely Related Hardware Architectures of Interest

Shaw's (1982, 1986) original Non-Von computer, which links 8-bit processors via a tree, has recently been augmented so that the bottom ply (the leaves of the tree) also link laterally to siblings. This gives some of the pyramid properties but, because there is neither overlap nor lateral linkage at higher levels, it is probably closer to a quad-tree, and therefore in danger of having crucial information lost in or separated by cracks.

An interesting and ambitious new 4-level system has been started at the University of Massachussetts (Levitan, personal communication). It will have a very large array of 1-bit processors at its base, 16-bit processors at the next level each linked to an 8 by 8 array of children, and 32-bit Lisp processors at the third level, each linked to a number of children. At the top level, all the Lisp processors will link to a single host.

Several other types of parallel computer appear to be appropriate for executing pyramid programs with appreciable increases in speed over the serial computer. They will not be as fast as true pyramid hardware. But for many purposes they may be more cost effective, or more flexible.

The PIPE pipelined image processor (Kent et al., 1985) can be programmed to remap data into successively smaller arrays, and therefore to execute converging pyramid procedures (see Kent and Tanimoto, 1985). Its 8 stages allow the programmer to build and process a pyramid with a 256 by 256 base. The PIPE can also be programmed to execute up to 256 different instructions at different locations in the array. This is accomplished by storing instructions in one of the array's two 8-bit memories.

The new Cyto-HSS (Lougheed and McCubbrey, 1985) similarly allows the programmer to

address and remap data and therefore build and process pyramids. It has the added advantage that this can be done at the rate of 10 million 3-by-3 window operations per second, rather than the 20 or 30 millisecond TV scan rate for the entire image. As the arrays grow smaller moving up the pyramid this speedup becomes extremely significant.

The PIXIE-5000 (Wilson, 1985) can also remap data in the direction of its scan in one instruction, and in the other direction using a short sequence of instructions. Since it takes only 80 microseconds to process a 1,024 by 1,024 array, in contrast to the 30 milliseconds or so needed by a pipelined scanning array like the Cyto and PIPE, it will still be extremely fast. It will be possible to use CLIP7, which is now being built and will scan a 1-dimensional array of 8-bit processors over the 2-dimensional image array, in a similar manner (Fountain, 1983, 1985).

An array like CLIP4 (Duff, 1976), DAP (Reddaway, 1978), or the MPP (Batcher, 1980) can be programmed to handle pyramids by either squeezing the higher layers into one corner, tiling the higher layers over the array, or scattering the elements of the higher layer over the array. If the bottom several layers of the pyramid being simulated are larger than the hardware array, and subarrays are stored in each processor's memory, the distances information must be shifted laterally through the array in order to simulate a pyramid will be reduced, and processing speeded up significantly (see Reeves, this volume).

An N-dimensional hypercube of the sort being sold by INTEL and NCUBE can be used in the same way (see Stout, this volume), since an array can conveniently be mapped into it. Such a system has the additional advantage of the N-cube linkage, which will cut down the distances that information must be passed. It will also have the MIMD system's added flexibility. On the other hand, MIMD systems are expensive, and the time taken to send information in a message-passing system of this sort is typically thousands of times longer than the time needed for a simple lateral shift in an array or a pyramid. Additional hardware (as planned for the NCUBE company's new systems) should reduce this to a few hundred times longer - a much shorter but still costly amount of time.

Uhr et al. (1983) developed a detailed design for, but never built, a 2-layer system where processors could be reconfigured to either an array of 32-bit MIMD computers or an MIMD-SIMD array of 8-bit computers. Since memory was shared, these could conveniently be used for pyramid processes. Sandon (1985) designed an array that can be reconfigured to have successively more powerful processors moving up through what can be used as a pyramid structure.

Other attractive MIMD systems link all processors via an NlogN reconfiguring network - e.g., PASM (Siegel, 1981) and the Butterfly (Crowther et al., 1985). These can be used even more conveniently than the hypercubes, since any computer can pass information to any other over the switching network that links them together. However, the costs of the switches grow rapidly as the number of computers in the network grows.

Since these MIMD systems use far more expensive processors and communication channels than would a massively parallel pyramid they would inevitably be smaller, hence slower. For example, given the packing densities that will almost certainly be achieved in VLSI technologies during the next 5 to 10 years - of 10 million or more transistors per chip - 256 or 1,024 1-bit processors can be put on each chip. But at least 8 chips would be needed for a conventional serial computer (almost all for memory). Therefore whereas it may be feasible to fabricate a 1,024 by 1,024 array for the base of a pyramid using only 1,024 chips, only a few hundred or a few thousand conventional computers could be fabricated with a few thousand chips and linked together into an N-cube, reconfigurable network, or some other topology. This suggests that the fastest and most cost effective architecture will be one that combines large pyramids of synchronized computers with a suitably structured network of more powerful and more independent processors.

Summary Comments

This paper examines the development of appropriately process-structured software/hardware systems for the perceptual recognition of real-world objects in real time.

The perception program can be represented as a data-flow graph, and the problem of developing an appropriate multi-computer topology treated as one of finding a structure of computers that can efficiently execute that graph as it scans over it. A 1-node graph (the 1-CPU computer) can scan over any graph; but it is far too slow. A 1-D pipeline (e.g., PICAP, Cyto, PIPE) can effectively execute local operations on arrays of iterated information of any dimension, reducing the time needed by the pipe's length. A 2-D array (e.g., CLIP, DAP, MPP) can handle 2, 3, and N-D arrays with speed-ups proportional to the number of processors - but they can be very slow at moving information together for global operations. A pyramid of arrays has all the capabilities of an array for massively parallel local processes, and also the good global properties of tree-based logarithmic structures.

Pyramids can be used with great potential power and efficiency, by treating them as 2-D pipelines through which information is flowed and successively transformed. Several different types of complex real-world objects (e.g., neurons, trees, windows, houses) have been successfully recognized in this way.

A pyramid of arrays becomes inefficient when different processes must be applied in different regions, and when different sub-sets of information must be flowed in different directions. There are a number of possible augmentations - both internal to the pyramid and in the form of additional networks of computers into which the pyramid can be embedded - that offer promise of further substantial increases in power and in generality.

References

[1] R.K. Bajcsy and D.A. Rosenthal, "Visual focussing and defocussing - an essential part of the pattern recognition process," *Proc. IEEE Conf. on Computer Graphics, Pattern Recognition and Data Structures*, 1975.

[2] ----, "Visual and conceptual focus of attention" in *Structured Computer Vision: Machine Perception Through Hierarchical Computation Structures*, S. Tanimoto and A. Klinger, Eds. New York: Academic Press, 1980, 133-149.

[3] D.H. Ballard, "Task frames in visuo-motor coordination," *Proc. Third Workshop on Computer Vision*, IEEE Computer Society Press, 1985, 3-10.

[4] G.H. Barnes, R.M. Brown, M. Kato, D.J. Kuck, D.L. Slotnick and R.A. Stokes, "The ILLIAC IV Computer," *IEEE Trans. Computers*, 1968, *17*, 746-757.

[5] K.E. Batcher, "Design of a massively parallel processor," *IEEE Trans. Computers*, 1980, *29*, 836-840.

[6] A. Bode, G. Fritsch, W. Handler, W. Henning, F. Hofmann and J. Volkert, "Multi-grid oriented computer architecture," *Proc. Int. Conf. Parallel Proc.*, 1985, 89-95.

[7] P.J. Burt, "The pyramid as a structure of efficient computation" in *Multiresolution Image Processing and Analysis*, A. Rosenfeld, Ed. New York: Springer-Verlag, 1984, 6-35.

[8] P.J. Burt, T.H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation," *IEEE Trans. System, Man, Cybernetics*, 1981, *SMC-11*, 802-809.

[9] M. Pietikainen and A. Rosenfeld, "Image segmentation by texture using pyramid node linking," *IEEE Trans. System, Man, Cybernetics*, 1981, *SMC-11*, 822-825.

[10] V. Cantoni, S. Ferretti, S. Levialdi and F. Maloberti, "A pyramid project using integrated technology" in *Integrated Technology for Image Processing*, S. Levialdi, Ed. London: Academic Press, 1985, 121-133.

[11] J.M. Cibulski and C.R. Dyer, "An analysis of node linking in overlapped pyramids," *IEEE. Trans. Syst., Man, Cybernetics*, 1984, *14*, 424-436.

[12] W. Crowther, J. Goodhue, E. Starr, R. Thomas, W. Milliken and T. Blackadar, "Performance measurements on a 128-node Butterfly parallel processor," *Proc. Int. Conf. Parallel Proc.*, 1985, 531-540.

[13] L.S. Davis and A. Rosenfeld, "Hierarchical relaxation" in *Computer Vision Systems*, A.R. Hanson and E.M. Riseman, Eds. New York: Academic Press, 1978, 101-109.

[14] M.J.B. Duff, "CLIP4: a large scale integrated circuit array parallel processor," *Proc. 4th Int. Joint Conf. on Pattern Recognition*, 1976, *4*, 728-733.

[15] C.R. Dyer, "Augmented cellular automata for image analysis," *Unpubl. Ph.D. Diss.*, Dept. of Computer Science, Univ. of Maryland, 1979.

[16] -----, "A quadtree machine for parallel image processing," *Information Engin. Dept. Tech. Rept. KSL 51*, Univ. of Illinois at Chicago Circle, 1981.

[17] -----, "Pyramid algorithms and machines" in *Multicomputers and Image Processing*, K. Preston and L. Uhr, Eds. New York: Academic Press, 1982, 409-420.

[18] C.R. Dyer, A. Rosenfeld and H. Samet, "Region representation: boundary codes from quadtrees," *Comm. ACM*, 1980, *23*, 171-179.

[19] T.J. Fountain, "The development of the CLIP7 image processing system," *Pattern Recognition Letters*, 1983, *1*, 331-339.

[20] -----, "Plans for the CLIP7 chip" in *Integrated Technology for Image Processing*, S. Levialdi, Ed. London: Academic Press, 1985, 199-214.

[21] W. Handler, "A Unified Associative and Von-Neumann Processor - EGPP and EGPP Array," *Lectures Notes in Computer Sci., vol. 24 - Parallel Processing, 97-99*, Springer-Verlag, 1975.

[22] M.D. Kelly, "Edge detection in pictures by computers using planning" in *Machine Intelligence 6*, R. Meltzer and D. Michie, Eds. New York: Elsevier, 1971, 379-409.

[23] E.W. Kent, M. Shneier and R. Lumia, "PIPE - Pipelined image processing engine," *J. Parallel and Distributed Computing*, 1985, *2*, 50-78.

[24] E.W. Kent and S.L. Tanimoto, "Hierarchical cellular logic and the PIPE processor: structural and functional correspondence," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, IEEE Computer Society Press, 1985, 311-319.

[25] B. Kruse. "The PICAP picture processing laboratory," *Proc. 4th Int. Joint Conf. on Pattern Recognition*, 1976, *4*, 875-881.

[26] -----, "System architecture for image analysis" in *Structured Computer Vision: Machine Perception Through Hierarchical Computation Structures*, S. Tanimoto and A. Klinger, Eds. New York: Academic Press, 1980, 169-212.

[27] M.D. Levine, "A knowledge-based computer vision system" in *Computer Vision Systems*, A. Hanson and E.

Riseman, Eds. New York: Academic Press, 1978, 335-352.
[28] -----, "Region analysis with a pyramid data structure" in *Structured Computer Vision*, S. L. Tanimoto and A. Klinger, Eds. New York: Academic, 1980, pp. 57-100.
[29] M.D. Levine and J. Leemet, "A method for nonpurposive picture segmentation," *Proc. 4th Int. Joint Conf. on Pattern Recognition*, 1976, *4*, 494-498.
[30] Z.N. Li and L. Uhr, "Comparative Timings for a Neuron Recognition Program on Serial and Pyramid Computers," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Data Base Management*, IEEE Computer Society Press, 1985, 99-106. (a)
[31] -----, "Pyramidal Algorithms for Analysis of House Images," *Proc. Intersoc. Conference on Artificial Intelligence Applications*, 1985. (b)
[32] -----, "A pyramidal approach for the recognition of neurons using key features," *Pattern Recognition*, 1986, *19*, 55-62.
[33] R.M. Lougheed and D.L. McCubbrey, "Multi-processor architectures for machine vision and image analysis," *Proc. Int. Conf. Parallel Proc.*, 1985, 493-497.
[34] D. Marr, *Vision*, San Francisco: Freeman, 1982.
[35] A. Merigot, B. Zavidovique, and F. Devos, "SPHINX, a pyramidal approach to parallel image processing," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, IEEE Computer Society Press, 1985, 107-111.
[36] R. Miller, Pyramid Computer Algorithms, *Unpubl. Ph.D. Diss.*, Dept. of Math., SUNY Binghamton, 1984.
[37] C.F. Neveu, C.R. Dyer and R.T. Chin, "Object recognition using Hough pyramids," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1985, 328-333.
[38] J.J. Pfeiffer, Jr., "Integrating low level and high level computer vision," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, IEEE Computer Society Press, 1985, 119-125.
[39] S.F. Reddaway, "DAP - a flexible number cruncher," *Proc. 1978 LASL Workshop on Vector and Parallel Processors*, Los Alamos, 1978, 233-234.
[40] J.P. Riganati and P.B. Schneck, "Supercomputing," *Computer*, 1984, *17*, 97-113.
[41] A. Rosenfeld, "Pyramids: multiresolution image analysis," *Proc. Third Scandinavian Conference on Image Analysis*, July, 1983, 23-28.
[42] A. Rosenfeld, Ed., *Multiresolution Image Processing and Analysis*, New York: Springer-Verlag, 1984.
[43] A. Rosenfeld and G.J. Vanderbrug, "Coarse-fine template matching," *IEEE Trans. Systems, Man, and Cybernetics*, 1977, *7*, 104-107.
[44] H. Samet, "A tutorial on quadtree research" in *Multiresolution Image Processing and Analysis*, A. Rosenfeld, Ed. New York: Springer-Verlag, 1984, 212-223.
[45] P.A. Sandon, "A pyramid implementation using a reconfigurable array of processors," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Data Base Management*, IEEE Computer Society Press, 1985, 112-118.
[46] D.H. Schaefer, "A pyramid of MPP processing elements - experiences and plans," *Proc. 18th Int. Conf. on System Sciences*, Honolulu, 1985.
[47] L. Schmitt, "The use of a network representation of visual knowledge in a hierarchically structured vision system," *Unpubl. Ph.D. Diss.*, Dept. of Computer Sciences, University of Wisconsin, 1981.
[48] D.W. Shaw, "The NON-VON Supercomputer," *Comp. Sci. Dept. Tech. Rept.*, Columbia Univ., August, 1982.
[49] -----, "Organization and operation of a massively parallel machine" in *Computers and Technology*, G. Rabat, Ed. Amsterdam: North-Holland, 1986.
[50] H.J. Siegel, "PASM: a reconfigurable multimicrocomputer system for image processing" in *Languages and Architectures for Image Processing*, M. J. B. Duff and S. Levialdi, Eds. London: Academic Press, 1981.
[51] K. Sloan, "A system for world model driven recognition of natural scenes," *Unpubl. Ph.D. Diss.*, Dept. of Computer Science, Univ. of Pennsylvania, 1977.
[52] S.R. Sternberg, "Cytocomputer real-time pattern recognition." Paper presented at *Eighth Pattern Recognition Symp.*, National Bureau of Standards, 1978.
[53] Q.F. Stout, "Sorting, merging, selecting, and filtering on tree and pyramid machines," *Proc. Int. Conf. on Parallel Processing*, 1983, 214-221.
[54] -----, "An algorithmic comparison of arrays and pyramids," in *Evaluation of Multicomputers for Image Processing*, L. Uhr, K. Preston, S. Levialdi, M.J.B. Duff, Eds. London: Academic Press, 1985.
[55] S.L. Tanimoto, "Pictorial feature distortion in a pyramid," *Comp. Graphics Image Proc.*, 1976, *5*, 333-352.
[56] -----, "Regular Hierarchical Image and Processing Structures in Machine Vision," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New York: Academic Press, 1978, 165-174.
[57] -----, "Towards hierarchical cellular logic: design considerations for pyramid machines," *Computer Science*

Dept. Tech. Rept. 81-02-01, Univ. of Washington, 1981.
[58] -----, "A pyramidal approach to parallel processing," *Proc. 10th Annual Int. Symposium on Computer Architecture*, Stockholm, 1983, 372-378. (a)
[59] -----, "Algorithms for median filtering of images on a pyramid machine," in *Computing Structures for Image Processing*, M.J.B. Duff, Ed. London: Academic Press, 1983, 123-141. (b)
[60] -----, "A hierarchical cellular logic for pyramid computers," *J. Parallel and Distributed Computing*, 1984, *1*, 105-132.
[61] -----, "An approach to the iconic/symbolic interface," in *Integrated Technology for Image Processing*, S. Levialdi, Ed. London: Academic Press, 1985, 31-38.
[62] S.L. Tanimoto and A. Klinger, Eds., *Structured Computer Vision: Machine Perception Through Hierarchical Computation Structures*, New York: Academic Press, 1980.
[63] J.M. Tenenbaum and H.G. Barrow, "IGS: a paradigm for integrating image segmentation and interpretation," *Proc. 4th Int. Joint Conf. on Pattern Recognition*, 1976, *4*, 504-513.
[64] L. Uhr, "Layered "recognition cone" networks that preprocess, classify, and describe." *Preprints of Conf. on Two Dimensional Digital Signal Processing*, University of Missouri, 1971.
[65] -----, "Layered "recognition cone" networks that preprocess, classify, and describe." *IEEE Trans. on Computers*, 1972, *21*, 758-768.
[66] -----, "Describing, using "recognition cones." " *Proc. 1st Int. Conf. on Pattern Recognition*, Washington, 1973.
[67] -----, ""Recognition Cones" that perceive and describe scenes that move and change over time." *Proc. 4th Int. Joint Conf. on Pattern Recognition*, San Diego, *4*, 1976.
[68] -----, ""Recognition cones" and some test results; the imminent arrival of well-structured parallel-serial computers; positions, and positions on positions," in *Computer Vision Systems*, A. Hanson and E. Riseman, Eds. New York: Academic Press, 1978, pp. 363-372.
[69] -----, "Chapter 1: Parallel-serial variable resolution perceptual systems," in *Structured Computer Vision*, S. Tanimoto and A. Klinger, Eds. New York: Academic Press, 1980.
[70] -----, "Converging pyramids of arrays," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Data Base Management*, IEEE Computer Society Press, 1981, 31-34.
[71] -----, "Pyramid Multi-Computer Structures, and Augmented Pyramids," in *Computing Structures for Image Processing*, M.J.B. Duff, Ed. London: Academic Press, 1983, pp. 95-112.
[72] -----, *Algorithm-Structured Computer Arrays and Networks: Architectures and Processes for Images, Percepts, Models, Information*. New York: Academic Press, 1984.
[73] -----, "Augmenting pyramids and arrays by embossing them into optimal graphs to build multicomputer networks," in *Parallel Integrated Technology for Image Processing*, S. Levialdi, Ed. London: Academic Press, 1985, 19-31. (a)
[74] -----, "Pyramid Multi-Computers, and Extensions and Augmentations," in *Algorithmically Specialized Parallel Computers*, L. Snyder, L.H. Jamieson, D.B. Gannon, H.J. Siegel, Eds. New York: Academic Press, 1985, 177-186. (b)
[75] -----, "Multiple image and multi-modal augmented pyramid networks," in *Intermediate Level Image Processing*, M.J.B. Duff, Ed. London: Academic Press, 1986, 127-145. (a)
[76] -----, "Constructing multi-level multi-computer networks," in *Evaluating Multi-Computers for Image Processing*, L. Uhr, K. Preston, S. Levialdi and M.J.B. Duff, Eds. New York: Academic Press, 1986. (b)
[77] -----, *Multi-Computer Architectures for Artificial Intelligence*, New York: Wiley, 1986. (c)
[78] -----, *Massively Parallel Hierarchical Pyramid Multi-Computers for Perception*, New York: Academic Press, 1987. (in press)
[79] L. Uhr and R. Douglass, "A parallel-serial recognition cone system for perception," *Pattern Recognition*, 1979, 11, 29-40.
[80] L. Uhr, J. Lackey and M. Thompson, M., "A 2-layered SIMD/MIMD Parallel Pyramidal "Array/Net"," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Data Base Management*, IEEE Computer Society Press, 1981, 209-216.
[81] S.S. Wilson, "The PIXIE-5000 - a systolic array processor," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, IEEE Computer Society Press, 1985, 477-483.