USE OF A PYRAMID PROCESSOR IN INTERMEDIATE-LEVEL VISION[1]

Steven L. Tanimoto
Department of Computer Science
University of Washington

FR-35
Seattle, Washington 98195
U. S. A.

ABSTRACT

Whereas low-level vision consists of filtering and other image-to-image operations, and high-level vision involves matching and inference of symbolic, relational structures such as graphs and frames, the problem area known as "intermediate-level vision" requires the extraction of features and symbols from a two-dimensional array of pixels. Conventional serial architectures do not have enough parallelism to handle low- and intermediate-level tasks very rapidly; at the same time, standard mesh architectures do well with low-level operations but are inefficient for intermediate and high-level problems. A pyramidal architecture, consisting of a hierarchy of meshes, can be more effective at intermediate-level problems than these architectures. Two powerful algorithm design methodologies are explored: bottom-up (reduction) methods, and top-down (refinement) methods. Feature extraction and feature selection are complementary approaches to intermediate-level vision and they are easily implemented with these methodologies. This is illustrated with a hierarchical Hough transformation and top-down feature-point localization. The programming of pyramid machines poses an interesting challenge; a pyramid-oriented iconic dataflow language, promises to make certain algorithms easy to express.

INTRODUCTION

Architectures for computer vision have generally addressed the need to process pixels using operations such as convolutions, median filtering, and contrast enhancement. To a lesser extent special hardware has been developed for feature extraction; special boards for connected components analysis and computing statistics such as averages and areas of regions have been developed. What has been missing are architectures that support the efficient parallel computation of wide varieties of image descriptions and features.

Intermediate-level vision can be described as those image computations which transform a two-dimensional image (in the form of an array of pixels) into one or more symbolic descriptors (which may be scalar features or more complicated relational structures). Intermediate-level computations can be contrasted with low-level computations, which input images and produce images (not symbols), and with high-level vision computations which begin with symbolic descriptions of images and manipulate them to resolve ambiguities or to infer specific things about the images.

Mesh-style parallel computers [Preston and Duff 1984] such as the CLIP4, MPP, GAPP, and the Connection Machine (ignoring for the moment its hypercube routing subsystem) can perform low-level operations efficiently because they can use one processing element per pixel, and the neighborhood data can be gathered very efficiently over the interconnections of the grid. These systems are less effective at intermediate-level operations, since the mesh does not provide the efficient global interconnections needed to rapidly compute overall statistics of an image.

Parallel processors suitable for handling symbolic computations are multiprocessors that follow the MIMD discipline rather than the SIMD one. These systems are effective at handling computation with coarse-grained parallelism; the tight communication required for neighborhood-oriented image processing would be inefficient on an MIMD system.

One suggestion for intermediate-level vision is to build a hybrid architecture that includes both a mesh SIMD processor for the pixel pushing and an MIMD multiprocessor for the symbolic computations. Then the problem is building an appropriate interface that allows the iconic-to-symbolic computations to run without a bottleneck (see [Tanimoto 1986]).

An alternative is to enhance the mesh architecture with a hierarchy of smaller meshes, resulting in a pyramid of processing elements. The hierarchical structure can permit global statistics to be computed much more efficiently than on a straight mesh. The pyramid of processing elements, or pyramid machine, can then handle both low-level and intermediate-level vision computations.

In this paper, the question which is addressed is, "how can a pyramid machine be used effectively for intermediate-level vision?"

PYRAMID MACHINE STRUCTURE

Pyramid machines can be defined in terms of topological structures called "hierarchical domains." A hierarchical domain is a structured discrete space made up of cells. We define a *pyramidal cell* to be a triple (k, i, j) where k is a *level index*, and $0 \leq k \leq L$, and L is a constant which is one less than the number of levels in the hierarchical domain. The cell indices obey the limits: $0 \leq i \leq 2^k$, and $0 \leq j \leq 2^k$.

We call the set of all pyramidal cells for a given value of L the *hierarchical domain* of order L. We call the subset of a hierarchical domain having first coordinate k the k-th *level* of the hierarchical domain. The zeroth level is called the *apex* or "root," and the L-th level is called the *base* or "finest level." Figure 1 shows the hierarchical domain of order three. The topology of a hierarchical domain is expressed in terms of two kinds of relations on pyramid cells: lateral and hierarchical. Let us first consider the hierarchical relations. A pyramidal cell $C' = (k', i', j')$ is a *child* of pyramidal cell $C = (k, i, j)$ if $k' = k + 1$, either $i' = 2i$ or $i' = 2i + 1$, and either $j' = 2j$ or $j' = 2j + 1$. In that case, we also say that C is the *parent* of C'.

The lateral relations include neighborings in the eight compass directions, as well as the general relation which is the union of the individual directional relations. In general, if $k' = k$, and $|i - i'| \leq 1$, and $|j - j'| \leq 1$, then C' is in the *lateral neighborhood* of C. If this is the case and $C \neq C'$ then we say C' is a lateral neighbor of C. More specifically, if $k' = k$, and $i' = i$, and $j' = j - 1$, then C' is a *south neighbor* of C. The other seven

[1] Research supported in part under NSF Grant IRI-8605889.

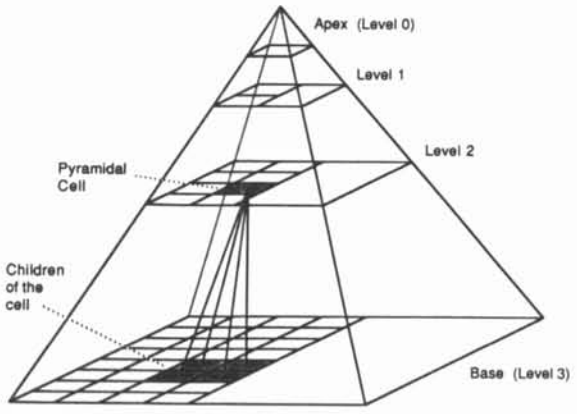

Figure 1: The hierarchical domain of order three.

compass-point relations are defined in similar fashion.

The entire neighborhood of a cell can now be defined. For a given pyramidal cell C, the lateral neighbors of C, the children of C, and the parent of C are all *pyramidal neighbors* of C. The *pyramidal neighborhood* of C is defined to contain only C and all the pyramidal neighbors of C. Except for pyramidal cells touching the borders of the hierarchical domain, a pyramidal neighborhood contains 14 pyramidal cells.

To define a pyramid machine of order L, one starts with the hierarchical domain of order L. For each cell of H, a processing element is provided. Secondly, a commmunication channel is established between each processing element and that for each of its neighboring cells (in general 13). The processing elements are centrally controlled by a control unit which consists of a Von-Neumann style serial computer with special extra instructions which control the operations of the pyramidal array of processing elements. Each processing element has its own local memory and it addresses it according to the global memory address broadcast by the controller. Some pyramid machines may permit this address to be modified locally, such as by adding the contents of a local index register. Each processing element has the capability to perform both logical and arithmetic operations.

Several prototypes of pyramid machines either have been constructed or are under construction. One system, developed at the University of Washington, employs a custom VLSI chip which implements sixteen processing elements per chip [Tanimoto et al 1987], and it is describe in more detail below. Another pyramid has been constructed from chips left over from construction of the MPP [Schaefer 1985]. The PAPIA pyramid machine is under development in Italy [Cantoni et al 1985].

In the University of Washington system, each processing element can, in one instruction, compare a pattern of bits (given as part of the instruction) with the contents of the neighborhood around it. It then resets its state according to whether there was a match or no match [Tanimoto 1984]. This pyramid machine implements the full 14-cell neighborhood whereas the other systems implement only a 10-cell neighborhood (diagonal lateral neighbors are not implemented).

THE UNIVERSITY OF WASHINGTON PYRAMID PROCESSOR HCLM-1

A modest prototype machine using custom VLSI chips has been constructed and used for simple image processing. This system uses nMOS technology with 16 processing elements per chip. The current implementation incorporates 341 active processing elements in a pyramid of base 16×16. We refer to this system here as the HCLM-1 (Hierarchical Cellular Logic Machine version 1). The control unit is a simple sequencer that supports conditional branching. The prototype can be and has been hosted by an IBM PC/AT system and by an Atari personal computer, interfaced through a parallel printer port. An assembler language and a graphics monitor program support program development in this hardware environment. Additional algorithm development support is provided by a pyramid simulation facility in ZetaLisp on a Symbolics 3600 with color display. Additional details on the HCL-1 prototype may be found in [Tanimoto et al 1987].

Currently work is proceeding on a new chip and prototype that will achieve higher circuit densities.

TAXONOMY OF INTERMEDIATE-LEVEL OPERATIONS

In order to better understand the realm of operations that may be considered as intermediate-level vision ones, let us consider possible ways to classify the operations.

One taxonomic scheme is to consider an operation as a function and examine its domain and its range. Usually, the domain of an intermediate-level vision operation is assumed to be two-dimensional image arrays. That leaves only the range of the operator as a basis for classification. The range should not be the same as the domain, for then the operation would be syntactically indistinguishable from a low-level (iconic) operation. The range is typically one of the following:

1. the real numbers; thus the output is a scalar, as might represent the area of the black regions of the image.
2. a list of vectors. The problem of recovering vectors from an image is the inverse of plotting vectors into a raster array which is common in graphics.
3. a relational structure; segmentation algorithms in the literature often produce a network whose nodes represent regions in the image.

Another classification scheme is based on the roles that the operations play in the larger process of vision. While an operator that selects a threshold for the image produces a scalar that is much like the area computed by another operator (mathematically they are both real numbers), these values play very different roles in the vision process are are semantically distinct. Then an operator which produces a set of threshold values would be classified with the operator which produces a single threshold, even though a set is syntactically different from a single element. This scheme can, however, classify operators together in the same class even though the processing implied by the operators can be very different (e.g., segmentation by thresholding followed by connected components labelling is done very differently from segmentation by split-and-merge region formation).

A third classification scheme is in terms of the computational complexity of the best known algorithm for the operator. This is particularly nice from an algorithm-design perspective, since it tends to group operators in such a way that algorithmic generalizations can easily be perceived.

Related to this scheme is a fourth classification method which is according to algorithmic strategy. We can classify an operator according to the most natural algorithmic strategy for it, on the architecture of interest. This is the classification used in organizing the remainder of this paper.

BOTTOM-UP METHODOLOGY

An obvious example of a bottom-up feature-extraction method is the computation of the average brightness of an image by having each processing element compute the average of the brightnesses of its four children. A related method begins with a thresholded edge image and counts the number of edge pixels by having each PE sum the numbers from its children; the overall sum can be used as a measure of the perimeter of an object in the image, provided there is only one object in the image, and provided that the edges are of proper quality.

A more interesting example of the bottom-up approach to intermediate-level vision is the extraction of line descriptors in a "pyramidal Hough transform" [Ku 1986], [Tanimoto 1988]. Let us now describe this algorithm.

Input: A gray-scale image of dimensions $2^L \times 2^L$. It is assumed that this image has been moved into the base of the pyramidal array before step 1 (below) begins.

Output: A list of (θ, ρ) pairs describing the m most prominent lines in the image.

Step 1. At level L apply a directional edge operator at each cell of the image, obtaining a descriptor (θ, α), where θ is the angular orientation of the edge and α is a measure of the contrast across the edge.

Step 2. Using the coordinates (i, j) of each base-level pyramidal cell, convert the (θ, α) pair for each cell to a set containing a triple $K_{(L,x,y)} = \{(\theta, \rho, \alpha)\}$. Here ρ is the distance from the origin of the coordinate system to the line passing through (i, j) making an angle θ with the horizontal.

Step 3. For $k \leftarrow L-1$ down to 0 do
For all cells (k, i, j) at level k do
$K_{(k,i,j)} \leftarrow \text{TopClusters}(K', K'', K''', K'''')$.

Here K' is $K_{(k+1,2i,2j)}$, K'' is $K_{(k+1,2i,2j+1)}$, K''' is $K_{(k+1,2i+1,2j)}$, and K'''' is $K_{(k+1,2i+1,2j+1)}$.

The function of TopClusters takes each of the four sets of clusters from the children and merges them into a single set; clusters that are within a short distance of one another are merged and their weights combined, and then the m winning clusters are retained and the others deleted. TopClusters uses a distance measure on parameter-space points:

$$\text{distance}((\theta_1, \rho_1), (\theta_2, \rho_2)) = w_\theta|\theta_1 - \theta_2| + w_\rho|\rho_1 - \rho_2|$$

where w_θ and w_ρ are weights to balance the contributions from each dimension of the parameter space.

The pyramidal Hough transform was implemented and tested [Ku 1986] on a Symbolics 3600 (with simulation software) The algorithm runs efficiently because it requires only a single bottom-up wave of activity through the pyramid. The amount of computation required at each level is $O(m(\log m)^2)$ time steps, which is the number of operations required to sort m items with a non-branching program.

For another example of a bottom-up line-clustering method, see [Hartley and Rosenfeld 1985].

TOP-DOWN METHODOLOGY

Provided a hierarchical representation of the image has been established (e.g., by averaging each cell's children to establish a value for the cell), various operations can proceed in a top-down manner through the pyramid. By performing an evaluation and a confinement of attention at each level in turn, the precision of the desired solution can be gradually improved. This general algorithm paradigm, termed "assessment and focus," is the subject of a recent Ph.D. dissertation [Blanford 1988].

As an example of a top-down operator for intermediate-level vision, we take the problem of finding the brightest point within the brightest zone of an image. If images were perfect digitizations of scenes, we might accept the pixel of maximum value in the image as representing the brightest point in the scene. However, because of sensor noise, the maximal pixel is not likely to be the one desired. A pixel which is locally maximal within a set of hierarchically-related contexts can be efficiently found by the following procedure.

Beginning with a pyramid data structure built by having each cell average the values of its children, a top-down search is performed. The search begins by marking the apex. Then a loop is begun: in each iteration the four children of the marked node compare their values and the mark is passed to the child with the maximal value. (The marked parent becomes unmarked.) The loop proceeds until a base-level cell becomes marked. The coordinates of this base-level cell may then be output as the solution.

Actually the most time-consuming part of the loop can be taken out of the loop and performed in parallel before the loop begins: for every cell in the hierarchical domain, the four children can compare their values and set a bit at the child with the highest one. Then in the loop it is only necessary to pass the mark to the child having the bit set. More interesting kinds of points can be found in an image by building the pyramid in ways other than straight averaging. Combinations of averaging and maximizing are particularly useful [Blanford and Tanimoto 1988].

While this simple search technique is indeed very simple, it is powerful because of the speed with which it can be computed and because it can be applied to a wide variety of images. It can be used as a component of more complex algorithms such as those for segmentation; region-based segmentation algorithms sometimes require the judicious selection of seed pixels before they become useful.

Another top-down algorithm is an interesting one for the generalized Hough transformation [Dyer 1987].

PROGRAMMING CONSIDERATIONS

While programming computers is challenging under ordinary circumstances, programming parallel machines can be particularly difficult because of the added complications of managing processor workload balance, interprocessor communication, and synchronization.

Image processing is very amenable to parallel processing. While massive amounts of pixel processing may be going on, it can all be expressed concisely using symbolic representations of whole images or pyramids.

Good software tools are important for programming parallel vision machines such as a pyramid machine. In addition to the essential language tools (assemblers, operating systems), special image-algorithm visualization tools are appropriate. For example, a spreadsheet-style interface for pyramidal algorithms has been explored [Blanford and Tanimoto 1986]. An image-flow language incorporating pyramids has been proposed for describing smart-sensor processing. This language employs pyramid-shaped icons along with other icons and symbols [Burt 1988]. We are currently exploring the possibility of using an iconic pyramid language for intermediate-level vision algorithm development. The elementary objects for this language are integer pyramids and functions of integer pyramids. A related language for images is HI-VISUAL [Hirakawa et al 1987]

Ultimately it may be possible to have a kind of automatic programming for these vision machines. Through algorithm generators, automatic parameter tuning, and good libraries of routines, efficient new algorithms for applications of machine vision could be produced with little human pain.

It should be noted that many studies by many researchers have been made on pyramid algorithms. It is beyond the scope here to survey them. One compendium of the earlier studies is [Tanimoto and Klinger 1980], and a slightly later collection is [Rosenfeld 1984].

REFERENCES

1. Blanford, R. P. 1988. Assessment and Focus: An Approach to Parallel Computer Vision. Ph.D. dissertation, Dept. of Computer Science, University of Washington, Seattle WA.

2. Blanford, R. P, and Tanimoto, S. L. 1986. The Pyramid-Calc system for research in pyramid machine algorithms. *Proceedings of the 1986 IEEE Computer Society Workshop on Visual Languages*, Dallas, TX, June 25-27, 1986. pp138-142.

3. Blanford, R. P. and Tanimoto, S. L. 1988. Bright-spot detection in pyramids. *Computer Vision, Graphics and Image Processing*, Vol. 43, pp.133-149.

4. Burt, P. J. 1988. Algorithms and architectures for smart sensing. *Proceedings of the DARPA Image Understanding Workshop*, April, pp.139-153.

5. Cantoni, V., Ferretti, M., Levialdi, S., and Maloberti, F. 1985. A pyramid project using integrated technology. In [Levialdi 1985], pp.121-132.

6. Duff, M. J. B. 1986. *Intermediate-Level Computer Vision.* London: Academic Press.

7. Dyer, C. R. 1987. Multiscale image understanding. In [Uhr 1987], pp171-213.

8. Hartley, R., and Rosenfeld, A. 1985. Hierarchical line linking for corner detection. In [Levialdi 1985], pp.101-119.

9. Hirakawa, M., Iwata, S., Yoshimoto, I., Tanaka, M., and Ichikawa, T. 1987. HI-VISUAL iconic programming. *Proceedings of the 1987 Workshop on Visual Languages*, sponsored by Linköping University and the University of Pittsburgh, held in Linköping, Sweden, Aug. 19-21, 1987, pp305-314.

10. Ku, K. L. 1987. Algorithms and Architectures for the Hough Transform. Ph.D. dissertation, Department of Electrical Engineering, University of Washington, Seattle, WA.

11. Levialdi, S. (ed). 1985. *Parallel Integrated Technology for Image Processing*, London: Academic Press.

12. Preston, K., Jr., and Duff, M. J. B. 1984. *Modern Cellular Automata: Theory and Applications.* New York: Plenum Press.

13. Rosenfeld, A. (ed.) 1984. *Multiresolution Image Processing and Analysis.* New York: Springer-Verlag.

14. Schaefer, D. H., Wilcox, G. C., Harris, V. J. 1985. A pyramid of MPP processing elements – experiences and plans. *Proc. 18th Annual Hawaii International Conference on System Sciences*, Vol. 1, pp.178-184.

15. Tanimoto, S. L. 1984. A hierarchical cellular logic for pyramid computers. *Journal of Parallel and Distributed Computing*, Vol. 1, No. 2, pp.105-132.

16. Tanimoto, S. L. 1986. Architectural issues for intermediate-level vision. In [Duff 1986], pp3-17.

17. Tanimoto, S. L. 1988. From pixels to predicates in pyramid machines. Proceedings of the COST Workshop, *From the Pixels to the Features*, held at Bonas, Gers, France, August 22-27.

18. Tanimoto, S. L., and Klinger, A. (eds.) 1980. *Structured Computer Vision: Machine Perception Through Hierarchical Computation Structures.* New York: Academic Press.

19. Tanimoto, S. L., Ligocki, T. J., and Ling, R. 1987. A prototype pyramid machine for hierarchical cellular logic. In [Uhr 1987], pp.43-83.

20. Uhr, L. (ed). 1987. *Parallel Computer Vision.* Orlando, FL: Academic Press.