JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 23, 475–483 (1994)

Adaptive Multiresolution Structures for Image Processing on Parallel Computers[1]

SOTIRIOS G. ZIAVRAS*,[2] AND PETER MEER†,[3]

*Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, New Jersey 07102; and †Department of Electrical and Computer Engineering, Rutgers University, Piscataway, New Jersey 08855

This paper presents a solution to the problem of creating region adjacency graph (RAG) pyramids on parallel computers comprising the hypercube topology. RAG pyramids represent hierarchies of irregular tesselations, with each tesselation generated in parallel by independent stochastic processes, and can be used for multiresolution image analysis. The outcome of the stochastic processes depends on the input data allowing the adaptation of RAG pyramids to the image content. For the extraction of connected components from labeled images, different connected components are reduced to different roots which are interconnected in a final region adjacency graph. An algorithm for implementing RAG pyramids on hypercube computers is discussed in detail and timing results are presented for the Connection Machine CM-2 supercomputer. The time complexity of the algorithm is found for the hypercube and the CRCW PRAM. The results show that the iterative process that creates a new level of the hierarchy from its preceding one does not heavily depend on the size of the graph, as its expected time is $O(\log N)$ for a random graph, where N is the total number of vertices in the input graph. The total number of levels is $O(\log(image\_size))$, as for the regular pyramid. © 1994 Academic Press, Inc.

1. INTRODUCTION

Salient features in an image are often large, that is, they have a global nature. A popular method for fast delineation of global features is the multiresolution approach [21]. At sufficiently low resolution a small neighborhood will contain all the salient information about a global feature. The feature can be detected by the application of a local operator to the low resolution representation. For successful multiresolution image analysis all the feature representations must remain discriminable during resolution reduction.

Let the input image be level 0 of the hierarchy of representations. The value of a pixel on the level $l + 1$ representation is computed from the values of a small set of pixels on level l. The computation is performed by a processing cell, and in this paper we will use the terms *cell*, *vertex*, and *pixel* interchangeably. The cell on level $l + 1$ is called the *parent*, while the aforementioned set of cells on level l are its *children*. Note that in general a child can have more than one parent. The child–parent links define the structure of the hierarchy. Tracing these links from a cell on level l down to the base delineates the *receptive field* of the cell in the input image. The value of the cell represents the receptive field at the resolution of level l.

The values of the parents are computed in parallel and the average number of children is always larger than one. Thus, the size of the receptive field of a cell increases strictly monotonically with the level in the hierarchy. The receptive field cannot exceed the size of the image and therefore the hierarchy building process will always converge after a few steps. If the average number of children on every level of the hierarchy is $\overline{K}$, then the complete stack of multiresolution representations contains $O(\log_{\overline{K}}(image\_size))$ levels.

Traditional multiresolution representations have regular structure with all parents having K children arranged in a square neighborhood. Adjacent neighborhoods may overlap. The value of the parent is computed as the weighted average of the values of its children. The weights are often chosen to correspond to a low-pass filtering of the children's level representation; therefore a smaller number of parents are required for the reduced resolution representation. In a traditional multiresolution hierarchy the number of pixels decreases fourfold between successive representations. These hierarchies are also known as *image pyramids*.

The regularly structured image pyramids are not optimal for *feature extraction* tasks where homogeneous regions have to be delineated. For example, consider an image with two large features separated by a small gap. During the resolution reduction, not only the extent of the features decreases but also their separation. Thus, the two features will be fused into one before they can be represented as local information, and it is not guaranteed that they will be discriminated at higher levels of the pyramid. Similarly, global connectivity of a connected component may become evident only at the top of the hierarchy, in which case the logarithmic processing time property of the pyramid will be violated [14]. Several feature extraction algorithms with adaptive child–parent links were proposed to avoid the artifacts of rigid pyramid structures. See [10, 21] for reviews. All these algorithms modify the links after the hierarchy is built and

[1] The authors were supported in part by the National Science Foundation under Grants CCR-9109084 and IRI-9210861, respectively.

[2] E-mail: ziavras@hertz.njit.edu.

[3] E-mail: meer@peacock.rutgers.edu.

0743-7315/94 $6.00

therefore they do not offer a satisfactory and efficient solution [4].

In most applications the features to be delineated are homogeneous regions in the input image. To keep their representations separated during the resolution reduction, the child–parent links must be defined based only on local homogeneity; i.e., all the children connected to a parent must belong to the same homogeneous patch. Such a child–parent relation yields homogeneous receptive fields for all the cells in the hierarchy. If the receptive fields do not overlap, feature delineation is reduced to identifying the cell whose receptive field coincides with the feature of interest. The condition for nonoverlapping receptive fields is a unique child–parent connection. That is, a cell on level l should be connected to only one cell on level $l + 1$. However, the resolution reduction requires that on the average several cells from level l converge to the same cell of level $l + 1$.

When the child–parent links are defined based on local homogeneity, every level of the multiresolution representation can be described as a *region adjacency graph (RAG)*. The cells on a given level are the vertices of the RAG for that level, while the edges of the graph describe the adjacency relations among the cells' receptive fields. Hierarchies built as stacks of RAG representations will be called *RAG pyramids*. Being molded after the homogeneous regions in the image, the structure of a RAG pyramid is no longer regular. RAG pyramids are optimal for feature extraction as they eliminate the artifacts of regular image pyramids while preserving the logarithmic processing time property [13]. A probabilistic technique was proposed recently for building RAG pyramids [17]. This technique is prone to parallel implementation, and in this paper we solve the problem of generating RAG pyramids on hypercube parallel computers and on the CRCW PRAM, and estimate the complexity of the solution. Performance results for the Connection Machine system CM-2 supercomputer follow.

The paper is organized as follows. Section 2 presents a sequential algorithm for building RAG pyramids. Section 3 introduces an algorithm that builds RAG pyramids on the hypercube. Its time complexity is compared to that of a similar algorithm for the CRCW PRAM. Section 4 presents results from the implementation of the algorithm on the Connection Machine system CM-2 supercomputer. Finally, Section 5 contains concluding remarks.

2. A PROBABILISTIC ALGORITHM FOR RAG PYRAMID CONSTRUCTION

An algorithm that applies a stochastic process for the construction of RAG pyramids [17] is succinctly described in this section. For the moment, let us assume that the entire image is homogeneous under the employed homogeneity criterion. For example, if homogeneity is measured by similar gray level, all the pixels in the image have the same value. Thus, the input adaptive behavior of RAG pyramids is eliminated to simplify the presentation of the algorithm. How the structure of RAG pyramids becomes molded by the input is discussed later in this section.

On level 0, which contains the input image, the RAG is the 8-connected mesh structure of the image. Every cell determines in parallel that its eight neighbors in the graph belong to the same homogeneity class. All the cells on level 0 thus acquire the same information. However, to build the hierarchy in $O(\log(image\_size))$ steps, only a subset of the level 0 cells can be selected for inclusion into the RAG of level 1. The selection process must satisfy the following two requirements for the creation of the level $l + 1$ RAG, $G_{l+1}(V_{l+1}, E_{l+1})$, where V_{l+1} and E_{l+1} are the sets of vertices (cells) and edges, respectively.

- Cells selected for level $l + 1$ should not be neighbors.
- Cells not selected for level $l + 1$ should have at least one selected neighbor.

These two conditions ensure that the apex of the hierarchy is reached in a logarithmic number of steps while using only local operations. In addition, they imply that the vertices of the RAG on the next level, $G_{l+1}(V_{l+1}, E_{l+1})$, represent a *maximum independent set (MIS)* [6] of the vertices in the graph $G_l(V_l, E_l)$. The most common technique for the generation of an MIS uses a greedy approach that generates an independent set vertex by vertex [11]. This technique selects a set of vertices for the MIS with probability reciprocal to the degree of the vertex. A parallel version of this technique first extracts an elimination tree from the given graph [20].

A parallel randomized algorithm to find an MIS of a graph was also presented [3]. This algorithm is iterative, with an expected number of iterations equal to $O(\log N)$, where N is the total number of vertices in the input graph. Another parallel algorithm of similar complexity was proposed in [12]. A probabilistic symmetry breaking iterative parallel technique [13] is used here. Every vertex in the graph is allocated with an i.i.d. (independent identically distributed) random variable. Uniform distribution [0, 1] is the simplest choice. If the outcome is a local maximum, i.e., all the neighbors drew smaller values, the vertex is selected into the MIS. The algorithms in [3, 13] are very similar. In contrast to the technique in [3] that applies a corrective step, the selection criterion cannot be satisfied by two adjacent vertices. However, both techniques require $O(\log N)$ iterations for random graphs. As the graphs used here are derived from an eight-connected mesh, in practice we have observed that a constant number, i.e., $O(1)$, of steps are required to generate an MIS. In fact, comparison of our technique with the one in [12] has shown a significant speed-up [16].

An example of vertices selected for the next level after one iteration of this MIS algorithm is shown in Fig. 1a for an 8×6 segment of an image. The density of selected vertices (filled in squares) may be low as the probability of being a local maximum is the reciprocal of the degree

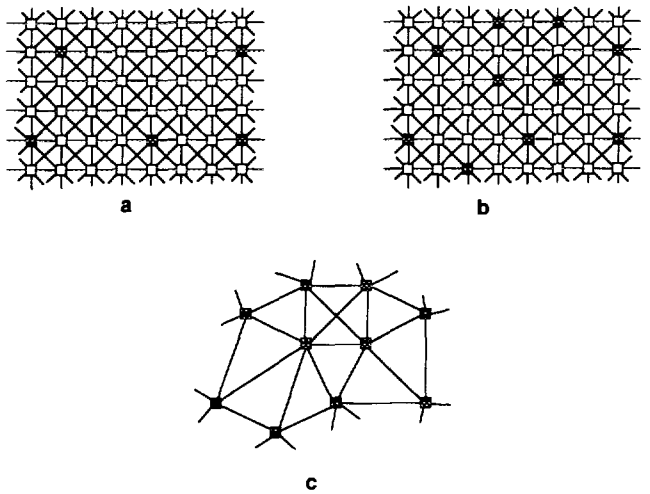

FIG. 1. Iterative process to select the vertices for level 1 in a uniform image: (a) First iteration. (b) Second iteration. (c) The RAG on level 1.

of the vertex. Subsequent iterations are applied only to vertices not adjacent to an already selected vertex. In all of our experiments, in at most five iterations the MIS was obtained. Figure 1b shows the selected vertices after the second iteration, which is the last one for this example.

After the selection of cells for inclusion in the next level graph, the child–parent links must be established. Note that in RAG pyramids each parent on level $l + 1$ is always a cell from level l. Thus, the child–parent links between levels l and $l + 1$ are the edges in the RAG of level l connecting the selected (parent) cell to some of its neighbors. When a child is neighbor to more than one parent, all but one of the links are broken by using the outcomes of the random variables in the last iteration (e.g., only the connection to the neighbor with the largest value drawn is kept). Since the receptive fields are nonoverlapping, the adjacency relations for cells on level $l + 1$ can be determined from the RAG of level l. This is a standard graph reduction procedure. Two parents are neighbors on level $l + 1$ if two or more of their children are neighbors on level l. This way, the edges for the RAG of level $l + 1$ are established. Figure 1c shows the RAG on level 1 for the input graph in Fig. 1a. The average decimation ratio for RAG pyramids is between 4 and 5 and the number of levels is $O(\log(image\_size))$ [13].

We can consider now the case of an arbitrary input image. Adapting the structure of the RAG pyramid to the content of the input image is done by introducing an additional processing step. Before the selection of the cells for level $l + 1$, every cell on level l classifies its neighbors into two classes, namely *similar* or *nonsimilar*, based on the homogeneity criterion defined by the task. Edges to nonsimilar neighbors are temporarily removed from the RAG and several separate *similarity partial subgraphs* are obtained. Each similarity graph corresponds to a homogeneous feature in the input. Since the decision threshold is computed locally and therefore may differ for adjacent cells, the similarity graphs are directed graphs. The resolution reduction is performed as above with the removed edges being taken into account when constructing the RAG of level $l + 1$.

When the local class allocation is independent of the cell selection process for previous levels, the RAG pyramid converges to a unique representation of the image. This is the case for labeled images. Each homogeneous region (i.e., connected component) is then represented by a single cell at the apex and the final RAG describes the topology in the input. Figure 2 shows the four highest levels of the hierarchy for an example image. By uniquely enumerating the nodes of the RAG at the apex and transmitting the labels in a top-down fashion following the links in the RAG pyramid, the connected components of the given image can be labeled. We restrict the discussion to examples using labeled images. In the case of gray level images, the local similarity threshold depends on the employed neighborhood which is determined by the previous selections. The probabilistic nature of the algorithm thus inroduces a spread in the possible outputs of the RAG pyramid which can be exploited to achieve reliable feature representations [16].

3. PARALLEL CONNECTED COMPONENT LABELING USING RAG PYRAMIDS

The introduction of an algorithm that creates RAG pyramids on hypercube parallel computers, as well as comparison of its performance with that of the theoretical CRCW PRAM, is the major objective in this section. Without loss of generality, the application algorithm cho-

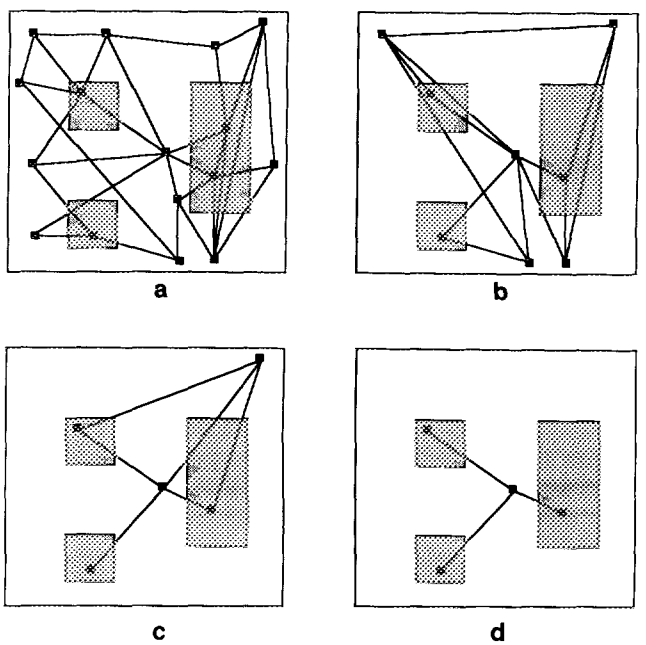

FIG. 2. The four highest levels of a RAG pyramid.

sen is connected component labeling. The connected component labeling problem is fundamental in image analysis and machine vision. The assignment of unique labels to connected regions reduces the complexity of high level image operations that distinguish among the various components. Several algorithms have been proposed for sequential and parallel computers. A survey of parallel algorithms was presented in [2] while representative work appeared, among others, in [8, 9, 15, 18].

The hypercube is one of the most frequently used topologies for the implementation of distributed-memory parallel computers. Its robust structure permits the efficient emulation of other frequently used topologies [23]. For example, algorithms that embed regular pyramids and a class of more robust multilevel structures [24] into the hypercube were investigated in [22, 25], respectively. Three types of parallel systems are considered in this paper for the theoretical evaluation of our algorithm's performance, namely the CRCW PRAM and the SIMD and MIMD hypercubes. It is assumed that hypercube systems have bidirectional channels, while processors can receive data on all incoming channels simultaneously and can send data only on one outgoing channel at a time.

An optimal one-to-one mapping of the $2^n \times 2^n$ image array onto the hypercube with 2^{2n} nodes is the first step of our algorithm. This mapping uses the binary reflected Gray code [22]. The creation of the RAG pyramid is accomplished as follows. Each cell with r neighbors in $G[l]$ creates an array λ_i of binary values, for $i = 0, 1, 2, \ldots, r$, with one value for each neighbor. At the base of the pyramid (i.e., $l = 0$), $r = 8$. More specifically, in Phase 1 of the algorithm the cell sets $\lambda_i = 1$ if the ith neighbor belongs to the same class; otherwise it sets $\lambda_i = 0$. The 0th neighbor is the cell itself; therefore $\lambda_0 = 1$. This way edges to nonsimilar neighbors can be temporarily removed to derive the similarity partial subgraphs discussed in Section 2 and subsequently adapt the structure of the RAG pyramid to the content of the image.

To determine the binary values of λ_i in Phase 1 for each cell on level l, every processor representing a cell on level l sends its pixel value to its r neighbors. This process does not take a significant amount of time for $l = 0$ due to the perfect mapping of the image array onto the hypercube. Phase 1 requires $O(r)$ time on the CRCW PRAM. As the structure of $G[l]$ is generally random, the transmission of a single value on the hypercube may take $O(n)$ time when message congestion at intermediate processors is ignored. For an SIMD hypercube, $O(n \times r)$ time is consumed in Phase 1 for the transmission of the r values, assuming that the transmission of a single message makes $O(n)$ hops. For an MIMD hypercube, however, this time is proportional to $(r - 1) + 2n$ or $O(n + r)$; this worst case appears when the last value sent by the processor with the largest number of neighbors goes to its diametrically opposite processor (i.e., at distance $2n$). Each processor then compares its own pixel value with the values it receives in order to determine the values for the elements of the array λ. As for higher levels the degree of a vertex may be larger than 8, for computer implementation the array λ should be created with many more than eight elements. Since the comparison of the r values takes $O(r)$ time, the time complexity for Phase 1 is similar to the time complexity for the transmission of the r values, as evaluated above.

The interative local process that selects the vertices from $G[l]$ to be included in $V[l + 1]$ is implemented as follows. Let cell c_0 have at most r neighbors in $G[l]$. Two binary state variables p and q and a random variable x uniformly distributed between 0 and 1 are associated with every cell. The cell c_0 is selected for inclusion in the MIS, and therefore in $G[l + 1]$, if after the last iteration k, where $k = 0, 1, \ldots$, its $p_0(k)$ state variable is 1. Initially, $p_i(0) = 0$, $\forall i = 0, 1, \ldots, r$, and every iteration of the local selection process is implemented in two phases, namely Phases 2 and 3.

In Phase 2, the value of q_0 in the kth iteration is determined by

$$q_0(k) = \begin{cases} 1 & \text{if } \lambda_i p_i(k-1) = 0 \quad \forall i = 0, 1, \ldots, r \\ 0 & \text{otherwise.} \end{cases}$$

Thus, $q_0(k)$ is set to 1 if none of its neighbors in the similarity subgraph (including c_0) was selected earlier. The implementation of Phase 2 requires the transmission of the value of the variable p to its r neighbors in time similar to that for Phase 1. The updated value of $q_0(k)$ is transmitted in Phase 3 to the neighbors and the new value of $p_0(k)$ is computed based on the received values for the array q:

$$p_0(k) = \begin{cases} 1 & \text{if } q_0(k)x_0(k) = \max(\lambda_i q_i(k) x_i(k)) > 0 \\ & \qquad \forall i = 0, 1, \ldots, r \\ p_0(k-1) & \text{otherwise} \end{cases}$$

The cell c_0 is selected for inclusion in the MIS if the outcome of the random variable for c_0 is a local maximum in the similarity graph. Phase 3 takes the same amount of time as Phase 1.

The iterative process that selects the cells for the next higher level is terminated when every cell either was selected earlier or has at least one neighbor which was selected. Otherwise, Phases 2 and 3 are implemented again. This condition is checked in Phase 2 by summing up the values of $q_0(k)$ for each unselected cell using the global sum operation and then comparing the result against zero. If the result is zero then the iterative process is terminated. The global sum operation is carried out by assuming a binary tree circuit, where bottom-up additions are performed. For the hypercube and a $2^n \times 2^n$ image, a many-to-one mapping of the binary tree with 2^{2n} nodes in the base onto the $(2n)$-dimensional hypercube is assumed. All the processors in the hypercube form the

base of the binary tree while parents on level s of the binary tree are emulated by hypercube processors whose $(2n)$-bit binary addresses have the lower $2s$ bits equal to 0. The global sum is then found in time proportional to $2n$ or $O(n)$ on any of the three systems (this time is normally $O(\log N_l)$ for the CRCW PRAM, where N_l is the number of nodes in the graph of level l and $N_l \leq 2^n \times 2^n$). In fact, the Connection Machine system CM-2 implements several global reduction operations in time proportional to the logarithm of the total number of processors in the system. To summarize, Phase 2 of the algorithm takes time $O(n + r)$, $O(n \times r)$, and $O(n + r)$ on the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively.

As the number of iterations is $O(n - l)$ for a random graph on level l (the number of iterations is $O(\log(\text{number\_of\_vertices}))$ while the graph $G[l]$ contains $O(2^{n-l})$ vertices), the iterative process in Phases 2 and 3 that selects the cells for the next level of the RAG pyramid requires total time $O((n - l) \times (n + r))$, $O((n - l) \times n \times r)$, and $O((n - l) \times (n + r))$ for the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively. As the expected number of iterations is $O(1)$ when the input graph is an image, the three time complexities are reduced to $O(n + r)$, $O(n \times r)$, and $O(n + r)$, respectively, for images.

In Phase 4 the links between parents on level $l + 1$ and their children on level l must be established. Every unselected cell chooses from its selected neighbors as parent the one with the largest random value drawn in the last iteration. Then it sends its address to its chosen parent, who stores it locally. Phase 4 takes the same amount of time as Phase 1 of the algorithm.

Phase 5 determines the set of edges $E[l + 1]$ for the RAG of level $l + 1$. By the condition imposed on the cell selection process, two cells in $G[l + 1]$ can be neighbors only if they were not more than three edges apart in $G[l]$. Since vertices in $V[l + 1]$ now also represent regions corresponding to their children on level l, region adjacencies denoted by their children should be combined to derive region adjacency information for the new graph $G[l + 1]$. This is the most time-consuming phase of the algorithm because it may involve a very significant amount of communication.

Phase 5 of the algorithm represents parallel *graph contraction*. Graph contraction is a sequence of edge contractions where pairs of vertices are merged and the edges between them are deleted [20]. The selective contraction in our algorithm is carried out only for edges that connect parents to their children. The creation of $E[l + 1]$ is carried out as follows. Every child gets the addresses of its neighbors' parents in time similar to that for Phase 1 of the algorithm. A one-bit flag that indicates the class of the neighbor as similar or nonsimilar is appended to each address. This class information is used later to decide about the neighbors' class on level $l + 1$. Cells selected for level $l + 1$ then get the latter addresses from their children. In our examples cells have 9–11 neighbors and 7–9 children (except for the topmost pair of levels which may have much less). Thus, each selected cell may get from 60 to 100 addresses.

The addresses that a parent gets from its children may be repeated several times, since several of its children may have links in $G[l]$ with cells that now have a common parent. Therefore, the receiving processor must compare the received addresses in order to discard any repetitions. Similarly, the children could first compare the parent addresses they receive from their neighbors in order to discard any repetitions. Assuming that a linked list structure is used to store the received addresses, the binary search method can discard repetitions and also sort the addresses is $O(r \times \log r)$ time [1]. The next step of the algorithm involves the children that send their sorted list of parent addresses for their neighbors to their parent. Since the number of addresses in a list is $O(r)$, this process takes time $O(r)$, $O(n \times r)$, and $O(n + r)$ for the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively. Each parent will then merge the $O(c)$ sorted lists of $O(r)$ addresses that it receives from its $O(c)$ children in order to discard more repetitions. Two sorted lists can be merged in linear time and the received sorted lists can be sorted in pairs using a binary tree of height $O(\log c)$, so this process takes $O(r \times c \times \log c)$ time as the total number of elements to be merged on each level of the binary tree is $O(r \times c)$. To conclude, Phase 5 of the algorithm takes time $O(r \times \log r + r \times c \times \log c)$, $O(n \times r + r \times \log r + r \times c \times \log c)$, and $O(n + r \times \log r + r \times c \times \log c)$ on the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively.

Alternately, the following potential improvement could be included in Phase 5 of the algorithm to reduce the computation time. The merging of sorted lists using the binary tree structure mentioned at the end of the last paragraph could be speeded up by using the children for the merging of intermediate lists. Therefore, a binary tree of height $O(\log c)$ is mapped onto the $O(c)$ processors that represent the children and the parent. This improvement increases the utilization of the processors in the system. This process starts with half of the children sending their sorted lists to the other half, then the receivers sorting the received sorted lists in order to discard repeated addresses, half of them sending their new sorted lists to the other half that merge their own list and the one just received, and so on. The final merging involves only the parent. The parent can send to the senders the addresses of the destinations for all the communication cycles in time $O(c/2 + c/4 + c/8 + \cdots + 1)$ or $O(c)$, $O(n \times c)$, and $O(n + c)$ for the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively. The number of levels in the binary tree is $O(\log c)$, a sender on level j of the binary tree transmits $O(2^j \times r)$ values, a receiver on level $j + 1$ spends time $O(2^{j+1} \times r)$ to merge a pair of sorted lists, $\sum_{j=0}^{a} 2^j = 2^{\alpha+1} - 1$, and $2^{\log c} = c$, therefore the total time for Phase 5 is $O((c \times r/c) \times \log(c \times r) + \log^2(c$

TABLE I
The Performance for a Random Graph of the Algorithm That Creates RAG Pyramids

System	Algorithm I	Algorithm II
CRCW PRAM	$O(n^3 + n^2 \times r + r \times \log r + r \times c \times \log c)$	$O(n^3 + n^2 \times r + r \times \log r)$
SIMD hypercube	$O(n^3 \times r + r \times \log r + r \times c \times \log c)$	$O(n^3 \times r + n \times r \times c)$
MIMD hypercube	$O(n^3 + n^2 \times r + r \times \log r + r \times c \times \log c)$	$O(n^3 + n^2 \times r + n \times \log^2 c + r \times c)$

$\times r)$) or $O(r \times \log r)$, $O(n \times r \times c)$, and $O(n \times \log^2 c + r \times c)$ for the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively.

The entire RAG pyramid building process is terminated when the graph obtained in the last step is the region adjacency graph of the original labeled image. For this condition to be satisfied, all the neighbors of each cell should belong to a different class from that of the referenced cell. Every cell tests this condition individually in Phase 6. It produces zero if none of its neighbors belong to its own class, otherwise it produces one. These results from all the selected cells are then added using the global sum operation. The entire process is terminated if the global sum is zero. This phase of the algorithm takes a time similar to that for Phase 1.

The total number of levels in the hierarchy is $O(n)$, as for the regularly structured pyramid; therefore Phases 1 and 4–6 are performed $O(n)$ times, while Phases 2–3 are performed $O(\sum_{l=0}^{n}(n - l))$ or $O(n^2)$ times. This process creates the RAG pyramid of an image. For the subsequent labeling of the connected components, the addresses of the processors that represent cells at the highest level of the hierarchy are propagated in a top-down fashion to the processors that represent the base. This final phase of connected component labeling takes time $O(n \times c)$, $O(n^2 \times c)$, and $O(n^2 + n \times c)$ on the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively.

Tables I and II summarize the performance of the algorithm that builds RAG pyramids for a random graph and for an image array in the input, respectively. These are also the times for labeling the connected components. Algorithm II incorporates the potential improvement suggested for Phase 5. In general, the constants associated with the time complexity for the MIMD hypercube are larger than the constants for the CRCW PRAM. Also, potential message collisions were ignored for the hypercube. For practical cases, the times are $O(n \times r)$, $O(n \times r \times c)$, and $O(r \times c)$ for the CRCW PRAM, SIMD hypercube, and MIMD hypercube, respectively.

The creation of a RAG pyramid on a hypercube may not be very efficient because of its irregularity. A close to optimal mapping of a RAG pyramid onto a hypercube, where neighboring nodes from the RAG pyramid are mapped to neighboring nodes of the hypercube, is very difficult to obtain and may require dynamic reassignment of cells. Neighboring nodes from the source graph may be mapped to distant nodes in the target graph, and vice versa. Therefore, communication times may be quite significant. These aspects were taken into consideration for the evaluation of the algorithm's time complexity. However, advanced systems with wormhole routing capabilities [19] can provide very good performance because, ignoring congestion at intermediate nodes, the communication time is very similar for any source–destination pair due to pipelining on the communication channels. The worst-case performance on the new Connection Machine system CM-5 for communication operations between distant processors is only a factor of 4 worse than best-case performance for neighboring processors. In fact, often the time complexity for communication operations is found by counting the number of routing steps in the parallel algorithm rather than the number of hops for their implementation. The worst-case time complexity of our algorithm for the SIMD and MIMD hypercubes then becomes equal to that for the CRCW PRAM.

4. CONNECTION MACHINE IMPLEMENTATION

A Connection Machine system CM-2 [5, 7] with 16K processors was used for the implementation of our algorithm. The image size considered was $2^7 \times 2^7$ or 128×128 (i.e., a total of 16K pixels) in most of the cases. The CM-2 contains the hypercube topology. To minimize the time for the comparison of addresses with the purpose of

TABLE II
The Performance for an Image Array of the Algorithm That Creates RAG Pyramids

System	Algorithm I	Algorithm II
CRCW PRAM	$O(n^2 + n \times r + r \times \log r + r \times c \times \log c)$	$O(n^2 + n \times r + r \times \log r)$
SIMD hypercube	$O(n^2 \times r + r \times \log r + r \times c \times \log c)$	$O(n^2 \times r + n \times r \times c)$
MIMD hypercube	$O(n^2 + n \times r + r \times \log r + r \times c \times \log c)$	$O(n^2 + n \times r + n \times \log^2 c + r \times c)$

removing duplicate ones during the creation of $E[l+1]$, a hashing of the addresses was first performed. Every processor representing a cell selected for the new level stores each address it receives from a child into one of four local arrays. The appropriate array is selected for each address based on a *modulo* 4 operation (i.e., the address is divided by 4 and the remainder of the operation determines the array to be used). Then, duplicate addresses are discarded in individual arrays. A procedure applying binary search may become very time-consuming, as multibit values must be compared and often be exchanged, while the CM-2 contains one-bit processors. Therefore, a one-bit variable is initialized to 0 for all elements in each array. The comparison is then done sequentially starting with the first element, and only elements for which the one-bit variable is equal to 0 are involved. Elements are discarded by setting this variable to 1. This way several unnecessary multibit comparisons are avoided. At the end, only elements that contain 0 for the one-bit variable are used to connect to neighbors and therefore establish the edges in $E[l+1]$.

Communication operations on the CM-2 are not carried out efficiently due to hardware limitations. Router nodes often create bottlenecks because they are shared by 16 processors. In addition, the bandwidth of the channels is low, as they are only one bit wide and data transfers must be synchronized to the low frequency clock due to the SIMD mode of computation. Finally, "send" operations with multiple simultaneous receipts, which appear very often in our algorithm, are not permitted; "get" operations must be used instead. As each "get" operation is implemented as a pair of "send" operations, it consumes significantly more time.

Before we present results for some synthetic images, it is important to show the almost logarithmic increase in the total execution time with the image size. For this reason, uniform images where all the pixel values are 0 are considered. The sizes of these images are $2^i \times 2^i$, for $i = 3, 4, 5, 6, 7$ and 8. In five cases each processor receives a single pixel. Only for $i = 8$ does each processor receive four pixels. Since the stochastic nature of the process that builds RAG pyramids does not necessarily produce the same number of levels with the regular pyramid, the algorithm sometimes was run several times until the number of levels produced was equal to $n + 1$ for the $2^n \times 2^n$ image. Results are presented in Table III. The almost logarithmic increase in the total execution time is illustrated by the fact that the ratio f of the total execution time (i.e., *Total* in Table III) divided by n, for the $2^n \times 2^n$ image, grows approximately linearly with the value of n. While the value of f is constant in the ideal case, these results show that the total execution time grows more slowly than $\alpha \times n^2$, where α is a constant. Therefore, increases in the total execution time do not keep up with increases in the image size (i.e., 2^{2n}).

Several other timings, expressed in seconds, are also shown in Table III. T represents the time it takes to find the addresses of neighbors on the new level $l + 1$ for the creation of $E[l+1]$. The following three timings are components of T. T_1 represents the time it takes children on level l to get the addresses of their neighbors' parents. T_2 represents the time it takes cells on level $l + 1$ to get the addresses of parents collected earlier from all of their children. This is the most time-consuming process in the algorithm as up to 100 addresses may be retrieved as discussed earlier. Finally, T_3 represents the time it takes cells on level $l + 1$ to discard duplicate addresses.

For the assignment of a single pixel from the image to each processor, synthetic images of size 128×128 were considered in the input. Seven images with rich features that can test all the intricacies of the process that builds irregular tesselations in a hierarchical fashion were considered. The results for these images are presented in Table IV. The T_i's, for $i = 1, 2$, and 3, are as for Table III. Image I contains four square objects with each object occupying approximately 50% of one of the four quadrants in the image. Images II and III contain four and six long objects, respectively. Images IV and V represent checkerboards with 8×8 pixels/square and 16×16 pixels/square, respectively. Finally, images VI and VII contain two white and three black bands, and one white and two black bands, respectively (i.e., the connectedness puzzle images of Minsky and Papert in [17]).

All of the results presented in this section show that the most time-consuming step is the one corresponding to parents getting addresses of neighbors' parents from all of their children (i.e., time T_2). The large execution times are attributed to the stochastic nature of the process that

TABLE III
Performance Results on the Connection Machine System CM-2 for Uniform Images, as a Function of the Image Size

Image Size	n	Total	T	T_1	T_2	T_3	f
8×8	3	8.7100	7.8169	0.1104	6.0137	1.6926	2.9033
16×16	4	12.3213	10.9785	0.2617	8.4222	2.2943	3.0803
32×32	5	15.6794	13.9171	0.2091	10.6325	3.0752	3.1358
64×64	6	20.3631	17.6813	0.3586	13.6242	3.6982	3.3938
128×128	7	24.1082	19.4058	0.3218	15.0322	4.0514	3.4440
256×256	8	29.3601	24.4006	0.4231	18.9321	5.0194	3.6700

TABLE IV
Performance Results for Several Images of Size 128 × 128 on the Connection Machine System CM-2

Image	Total	T	T_1	T_2	T_3	Levels
I	36.7411	29.5114	0.4067	23.4824	5.6219	10
II	34.5275	27.6435	0.5099	21.5504	5.5827	10
III	32.6256	26.5781	0.4811	20.8030	5.2935	9
IV	41.7637	33.6086	0.4690	26.1449	6.9942	11
V	37.9642	30.9815	0.5255	23.9695	6.4859	12
VI	32.5569	25.9896	0.4714	20.0891	5.4286	9
VII	31.3258	25.4032	0.3988	19.6916	5.3123	10

Note. The conventional pyramid has eight levels.

results in random graphs and to the weak capabilities of the CM-2 system, as justified earlier.

5. CONCLUSIONS

This paper introduced an algorithm for creating RAG (region adjacency graph) pyramids on parallel computers comprising the hypercube topology. In contrast to the regular pyramid, RAG pyramids adapt their structure to the content of the image. Nevertheless, the number of levels in both structures is $O(\log(image\_size))$. RAG pyramids are appropriate for feature extraction tasks where homogeneous regions must be delineated. The implementation of our algorithm on the Connection Machine system CM-2 was discussed and results were presented. The results show that the creation of RAG pyramids for practical cases may become a time-consuming process when compared to the creation of the regular pyramid, for a system that cannot implement efficiently global communications. However, the creation of RAG pyramids facilitates subsequently the efficient and highly accurate extraction of image features.

ACKNOWLEDGMENT

The permission to access the CM-2 system at the University of Maryland Institute for Advanced Computer Studies (UMIACS) in College Park is greatly appreciated. Also, the authors thank Mr. D. Shah for the programming work on the CM-2.

REFERENCES

1. Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
2. Alnuweiri, H. M., and Prasanna, V. K. Parallel architectures and algorithms for image component labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **14,** 10 (1992) 1014–1034.
3. Alon, N., Babai, L., and Itai, A. A fast and simple randomized parallel algorithm for the maximum independent set problem. *J. Algorithms* **7** (1986), 567–583.
4. Bister, M., Cornelius, J., and Rosenfeld, A. A critical view of pyramid segmentation algorithms. *Pattern Recognition Let.* **11,** (1990), 605–617.
5. Chen, L. T., Davis, L. S., and Kruskal, C. P. Efficient parallel processing of image contours. *IEEE Trans. Pattern Anal. Mach. Intell.* **15,** 1 (1993), 69–81.
6. Christofides, N. *Graph Theory: An Algorithmic Approach*, Academic Press, London, 1975.
7. Thinking Machines Corporation. *Connection Machine Model CM-2 Technical Summary*, Version 6.0., Cambridge, MA, 1990.
8. Cypher, R., Sanz, J. L. C., and Snyder, L. An EREW PRAM algorithm for image component labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **11,** 3 (1989), 258–262.
9. Cypher, R., Sanz, J. L. C., and Snyder, L. Algorithms for image component labeling on SIMD mesh connected computers. *IEEE Trans. Comput.* **39,** 2 (1990), 276–281.
10. Dyer, C. R. Multiscale image understanding. Uhr, L. (Ed.), *Parallel Computer Vision*. Academic Press, Boston, 1987, pp. 171–213.
11. Gavril, F. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM J. Comput.* **1** (1972) 180–187.
12. Luby, M. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.* **15** (1986), 1036–1053.
13. Meer, P. Stochastic image pyramids. *Comput. Vision Graphics Image Process.* **45,** 3 (1989), 269–294.
14. Meer, P., Sher, C. A., and Rosenfeld, A. The chain-pyramid. Hierarchical processing of contours. *IEEE Trans. Pattern Anal. Mach. Intell.* **12** (1990), 363–376.
15. Miller, R., and Stout, Q. Data movement techniques for the pyramid computer. *SIAM J. Comput.* **16,** 1 (1987), 38–60.
16. Montanvert, A., Meer, P., and Bertolino, P. Hierarchical shape analysis in gray level images. Toet, A., O, Y.-L., Foster, D. A., Heijmans, H. J. A. M., and Meer, P. (Eds.). *Shape in Picture*. Springer-Verlag, Berlin, 1993.
17. Montanvert, A., Meer, P., and Rosenfeld, A. Hierarchical image analysis using irregular tessellations. *IEEE Trans. Pattern Anal. Mach. Intell.* **13,** 4, (1991), 307–316.
18. Nassimi, D., and Sahni, S. Finding connected components and connected ones on a mesh-connected parallel computer. *SIAM J. Comput.* **9,** 4 (1980), 744–757.
19. Ni, L. M., and McKinley, P. K. A survey of wormhole routing techniques in direct networks. *Computer* (Feb. 1993), 62–76.
20. Reif, J. H. (Ed.). *Synthesis of Parallel Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.
21. Rosenfeld, A. (Ed.), *Multiresolution Image Processing and Analysis*. Springer-Verlag, Berlin, 1984.
22. Ziavras, S. G. Connection machine results for pyramid embedding

algorithms. Bouge, L., Cosnard, M., Robert, Y., and Trystram, D. (Eds.). *Lecture Notes in Computer Science*, Vol. 634. Springer-Verlag, Berlin/New York, 1992, 31–36.

23. Ziavras, S. G. On the problem of expanding hypercube-based systems. *J. Parallel Distrib. Comput.* **16,** 1 (1992), 41–53.
24. Ziavras, S. G. Efficient mapping algorithms for a class of hierarchical systems. *IEEE Trans. Parallel Distrib. Systems* **4,** 11 (1993), 1230–1245.
25. Ziavras, S. G., and Shah, D. P. High performance emulation of hierarchical structures on hypercube supercomputers. *Concurrency Practice Experience* **6,** 1 (1994).

SOTIRIOS G. ZIAVRAS received the Diploma degree in electrical engineering from the National Technical University, Athens, Greece, in 1984, the M.Sc. degree in electrical and computer engineering from Ohio University in 1985, and the D.Sc. degree in computer science from George Washington University in 1990. He was a distinguished graduate teaching assistant and a research assistant at George Washington University. From 1988 to 1989, he was also with the Center for Automation Research at the University of Maryland, College Park. He was a visiting assistant professor in the Department of Electrical and Computer Engineering at George Mason University in the Spring of 1990. He joined the Department of Electrical and Computer Engineering at New Jersey Institute of Technology in the Fall of 1990, as an assistant professor. He has authored or coauthored more than 45 refereed technical papers. He is on the Advisory Board of the Computer and Information Science Section of the New York Academy of Sciences, and is an associate editor of the *Pattern Recognition* journal.

PETER MEER received the Dipl. Engn. degree from the Bucharest Polytechnic Institute, Romania, in 1971, and the D.Sc. degree from the Technion, Israel Institute of Technology, Haifa, Israel, in 1986, both in electrical engineering. From 1971 to 1979, he was with the Computer Research Institute, Cluj, Romania, working on R&D of digital hardware. Between 1986 and 1990, he was an assistant research scientist at the Center for Automation Research, University of Maryland, College Park. In 1991, he joined the Department of Electrical and Computer Engineering, Rutgers University, as an assistant professor. He spent the Summer of 1992 at Auditory and Visual Perception Research Laboratories, Advanced Telecommunications Research (ATR) Institute, Kyoto, Japan, as an invited research scientist. He was cited by the International Pattern Recognition Society for an outstanding contribution to the journal *Pattern Recognition* in 1989, and is an associate editor of that journal.

Received June 18, 1993; revised August 25, 1993; accepted September 8, 1993.

Statement of ownership, management, and circulation required by the Act of October 23, 1962, Section 4369, Title 39, United States Code: of

JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING

Published monthly by Academic Press, Inc., 6277 Sea Harbor Drive, Orlando, FL 32887-4900. Number of issues published annually: 12. Editors: Dr. A. Gottlieb, Ultracomputer Research Laboratory, Courant Institute of Mathematical Sciences, New York University, 715 Broadway, 10th Floor, New York, NY 10003; Dr. K. Hwang, Department of EE-Systems, EEB 200, University Park, University of Southern California, Los Angeles, CA 90089-2562; and Dr. S. Sahni, Department of Computer and Information Science, University of Florida, 358 Computer Science Road, Gainesville, FL 32611.

Owned by Academic Press, Inc., 525 B Street, San Diego, CA 92101-4495. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, and other securities: None.

Paragraphs 2 and 3 include, in cases where the stockholder or security holder appears upon the books of the company as trustee or in any other fiduciary relation, the name of the person or corporation for whom such trustee is acting, also the statements in the two paragraphs show the affiant's full knowledge and belief as to the circumstances and conditions under which stockholders and security holders who do not appear upon the books of the company as trustees, hold stock and securities in a capacity other than that of a bona fide owner. Names and addresses of individuals who are stockholders of a corporation which itself is a stockholder or holder of bonds, mortgages, or other securities of the publishing corporation have been included in paragraphs 2 and 3 when the interests of such individuals are equivalent to 1 percent or more of the total amount of the stock or securities of the publishing corporation.

Total no. copies printed: average no. copies each issue during preceding 12 months: 1523; single issue nearest to filing date: 1500. Paid circulation (a) to term subscribers by mail, carrier delivery, or by other means: average no. copies each issue during preceding 12 months: 945; single issue nearest to filing date: 967. (b) Sales through agents, news dealers, or otherwise: average no. copies each issue during preceding 12 months: 0; single issue nearest to filing date: 0. Free distribution by mail, carrier delivery, or by other means: average no. copies each issue during preceding 12 months: 76; single issue nearest to filing date: 76. Total no. of copies distributed: average no. copies each issue during preceding 12 months: 1021; single issue nearest to filing date: 1043.

(Signed) Evelyn Sasmor, Senior Vice President