ON THE MAPPING PROBLEM FOR MULTI-LEVEL SYSTEMS

Sotirios G. Ziavras
Department of Electrical Engineering and Computer Science
George Washington University
Washington, D.C. 20052
and
Center For Automation Research
University of Maryland
College Park, MD 20742

Abstract

Hierarchically-structured arrays of processors have been widely used in the low-level and the intermediate-level phases of computer vision. This is because tasks in these phases require both local and global operations, when the two-dimensional array structure of the image is considered. This paper introduces mapping (process assignment) algorithms for systems in the above class. It is the first time in parallel computer vision that both the domain and the range of the mapping functions are in a general set of hierarchically-structured arrays of processors. More specifically, the systems being studied here are not necessarily homogeneous; the processing powers of processors at different levels and the reductions between different pairs of consecutive levels are allowed to vary. Efficient mapping is achieved by first proposing objective functions, so that each objective function measures the quality of a given mapping with respect to a particular optimization goal. Mapping algorithms, one for each objective function, that attempt to produce an optimal mapping by minimizing the corresponding objective function, are then proposed. It is proven theoretically that our mapping algorithms always yield an optimal solution for systems composed of processors with identical processing powers. In all other cases, some assignment choices in the algorithms allow to take advantage of the increased processing powers of processors.

1 INTRODUCTION

The main goal of the low-level and the intermediate-level phases of computer vision is to locate objects present in images and then produce a description of them; this description is then used by the high-level image understanding tasks to identify individual objects and their spatial relationships in the given scene. The first two phases of computer vision are characterized by both local and global operations, when the two-dimensional array structure of the image is considered, with the majority of the operations being local [Ro84]. Pyramid machines, which consist of successive layers of Mesh-Connected arrays of Computers (MCC), of size decreasing with the increase of the level number (if the lowest level number corresponds to the finest array), support efficiently both local and global operations [MLL88, Ro84, Uh87]; they are also very suitable for divide-and-conquer types of computations. Most of the times, the design is based on the (standard) pyramid, where each PE at a higher level has four sons at the level below.

As noted in [CaLe88], any of two major strategies may be followed to build a pyramid machine: (1) using fine granularity, where one PE per image pixel is conceived at each level, and (2) using coarse granularity, where one microprocessor is associated to an image block (since the VLSI technology may not support the utilization of fine granularity within a reasonable cost). There are two major categories of pyramid machines that differ in their control structure: the SIMD pyramid [Dy81, Ta83] and the MSIMD (Multiple SIMD) pyramid [ClMe87], which is of the SIMD type in each level. Of course, the first category of pyramidal systems can be considered a subset of the second category, because all the operations of a member in the former category can be implemented by the corresponding member in the latter category (by synchronization of its controllers) without loss of efficiency. Emphasis is given to MSIMD systems in this research.

Multiresolution (pyramid) structures have been simulated on arrays and hypercubes [ChSa86, Re86, St86]. We now present the most important reasons that may make pyramid machines more cost effective and/or more efficient than hypercube systems; we will make this comparison because commercial systems of the latter type have been widely used for scientific applications [Hi85], including computer vision. If more than one levels of the pyramid are assumed to be active at a time on different tasks (for example, pipelining [AhSw84] or con-

current multi-level processing [ChSa86, Te85]), then the (MSIMD) pyramid will be more cost-effective mainly because it has as many controllers as levels, while the hypercube will need to operate in the MIMD mode (which means that each processor in the system needs to have its own controller). Another reason that makes the pyramid machine more cost-effective is that it has a much smaller number of communication channels. Concurrent multi-level algorithms do not also seem as well-suited to the hypercube, if distinct nodes from the pyramid need to be mapped onto distinct nodes of the hypercube, with the distances between "lateral" neighbors being one and between "vertical" neighbors being at most two, since one has to use a hypercube of one higher dimension than that needed for just the finest mesh [ChSa86]; if the size of the finest mesh is $2^n x 2^n$ nodes, then the pyramid has $\lfloor 2^{2(n+1)}/3 \rfloor$ PE's and $2^{n+2}(2^n-1)$ bidirectional channels, while the hypercube has 2^{2n+1} PE's and $(2n+1)2^{2n}$ bidirectional channels. If a hypercube system having as many PE's as the base of the pyramid is considered, then for the above cases and for pipelining with the same operation at all levels (i.e., for algorithms that are also appropriate for SIMD machines), the performance of the hypercube will be lower compared with the performance of the pyramid; this is because there will be processors in the hypercube assigned to more than one levels of the multiresolution structure [St86], so the hypercube will basically be capable of simulating only one level of the pyramid at a time. If SIMD systems without pipelining or concurrent multi-level processing are considered, then the hypercube can offer performance comparable to that of the pyramid. Thus, in most of the cases a pyramid machine delivers more throughput and is more cost-effective than a hypercube machine for this kind of applications.

Of interest to this study is also the simulation of multiresolution structures on arrays of processors, because our mapping algorithms will also be applicable in that case. A mapping of a pyramid onto a flat array has been presented in [CaLe88] to obtain the maximum of efficiency, hardware economy, and modular simplicity. Each PE belongs to at most two different levels, when the corresponding mapping is considered: the base level and a higher level. Fig. 1 shows the mapping for a standard pyramid of four levels (the finest array is at level 0). The above mapping of PE's onto the two-dimensional grid produces a planar recursive distribution similar to the one of the balanced binary tree with a number of nodes equal to $2^n - 1$, where n is a positive integer.

In this paper, the mapping (process assignment) problem is considered in such a way that both the domain (containing *source architectures*) and the range (containing *target architectures*) of the mapping functions are in the set of multi-level systems.

Definition 1. *Multi-level systems* are pyramid-like systems (that is, systems composed of successive layers of mesh-connected arrays of PE's, where the number of PE's in the arrays decreases with the increase of the level number, only pairs of consecutive levels are allowed to communicate directly with each other, PE's are connected to each other by point-to-point bidirectional

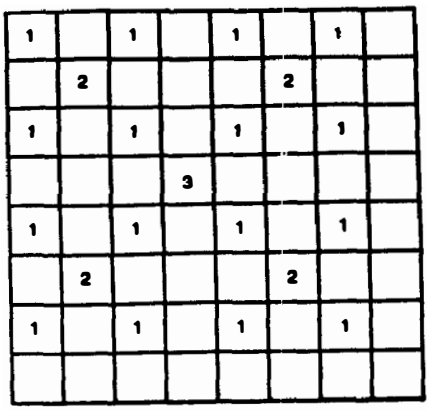

Fig. 1. Embedding a pyramid of four levels in a flat array [CaLe88].

communication channels, and the number of data transfer registers (DTR's) of any PE is equal to the number of its communication channels) that have the following characteristics: **(1)** they are homogeneous at each level (that is, all PE's of a single level are of the same type); **(2)** they may be heterogeneous between different levels; more specifically, the processing powers of PE's are allowed to stay the same between pairs of consecutive levels, or to increase with the level number; the formal definition of a PE's processing power is given in Definition 2 below; we assume that the bandwidths of channels also increase proportionally to the processing powers, to avoid bottlenecks caused by communications; systems where the "processing power" of PE's increases with the level number are sometimes very powerful for this kind of applications [Uh87]; **(3)** they have any number of levels (i.e., between 1 – flat array – and k, where $k > 1$, levels), which means that they are not necessarily single-rooted systems; **(4)** they have reductions of size $2^m x 2^m$, where m are natural numbers, between consecutive levels (the reductions are not forced to be the same for all pairs of consecutive levels); and finally, **(5)** there is a single controller per level (MSIMD mode of parallelism).

Definition 2. The *processing power* of a PE is defined to be the frequency it is driven at, divided by the average instruction cycles; the *average instruction cycles* is obtained by summing up the number of cycles for all the instructions in the processor's instruction set and then dividing the result by the total number of instructions.

The average instruction cycles has been used as the most important metric to distinguish between RISC (Reduced Instruction Set Computers) and CISC (Complex Instruction Set Computers) processors [Ka85]; it is a good measure of comparing the speed of two processors driven at the same frequency.

The kind of mappings being studied in this paper, for multi-level source and target architectures, may be necessary for one or more of the following reasons: (1) a target architecture with a few levels is considered, which

is more cost-effective than the source architecture corresponding to a particular application algorithm, because the former architecture has fewer PE's and communication channels; (2) the reductions in the source are different from the reductions in the target architecture; (3) the algorithms are designed for the most efficient multi-level architectures (i.e., the ones that correspond to the smallest amount of operations) and their portability to existing (target) architectures is required; and (4) the efficiencies of different target architectures in simulating a source architecture need to be compared. It is the first time in parallel computer vision that the mapping problem is considered in such a way that both the domain and the range of the mapping functions are in a general set of pyramid-like systems. Another innovation of our research is that it also allows for some degree of heterogeneity in those systems. It is worthy mentioning at this point that, if our mapping procedure is applied for the simulation of a standard pyramid on a flat array, then the obtained mapping will be identical to the one proposed in [CaLe88].

The paper is organized as follows. In order to produce mappings of the maximum of efficiency, objective functions are proposed in the next section, to measure the quality of a given mapping with respect to particular optimization goals. The set of inputs to our mapping algorithms are also presented in the same section. Then our mapping algorithms (one for each objective function), whose goals will be to minimize the previously mentioned objective functions, are proposed in Section 3. A theorem is also presented which states that, any of our mapping algorithms achieves its ultimate goal by always minimizing the corresponding objective function for systems composed of PE's having identical processing powers. Some illustrative examples of mapping are discussed in Section 4, while some tables corresponding to a large number of example systems and containing the parameters of the mappings (obtained through the application of our mapping algorithms), are presented in Section 5. Finally, the conclusions (Section 6) and an appendix containing some theorems, on which our mapping algorithms are based, are presented.

2 THE MAPPING PROBLEM

2.1 AN INTRODUCTION

The solution to the mapping problem for application algorithms is generally composed of two phases: *process assignment* and *scheduling* [Bo81, BoMi88]. The process assignment problem for systems composed of identical PE's concerns in general with the mapping of nodes (processes) from a given source architecture (having a graph representation corresponding to an application algorithm) onto the nodes (PE's) of a given target architecture, in a manner that minimizes the distances between communicating PE's, but maximizes the number of processes mapped onto distinct PE's. The scheduling problem concerns with the efficient implementation of the communication patterns and the scheduling of the operations on the target architecture, after the process assignment phase is complete.

In this paper, we deal only with the process assignment phase of the solution to the mapping problem, so the terms process assignment and mapping will be used interchangeably from now on. Our mapping algorithms formally yield assignments f of type

$$f = \{i \mapsto j,\ (i, x_i, y_i) \mapsto (j, x'_j, y'_j)$$

$$with\ x'_j = f_i(x_i)\ and\ y'_j = f_i(y_i)\}$$

In the above, "$i \mapsto j$" represents the mapping of level i from the source architecture onto level j of the target architecture; "$(i, x_i, y_i) \mapsto (j, x'_j, y'_j)$" represents the mapping of the PE with Cartesian coordinates (x_i, y_i) at level i of the source architecture onto the PE with Cartesian coordinates (x'_j, y'_j) at level j of the target architecture.

We propose two mapping algorithms in this research, based on the identification of two classes of vision algorithms for multi-level systems. The first class of vision algorithms, say *class A*, corresponds to single image processing without concurrency at different levels (i.e., one level of the source architecture is active at any time), while the second class of algorithms, say *class B*, corresponds to single-image concurrent multi-level processing (e.g., [Te85]) and pipelining. We distinguish between the two classes based on the performance requirements of vision algorithms with respect to the required utilization of levels on a multi-level system. More specifically, the minimization of the total image turnaround time, irrespective of the utilization of levels on the target architecture, is the optimization goal for algorithms in class A; however, the maximization of the throughput in the case of multiple image processing, and the minimization of the total image turnaround time in the case of single-image concurrent multi-level processing (i.e., for algorithms in class B), require the minimization of the workload, corresponding to the processing of a single image, for each active level of the target architecture; this is because the level with the highest utilization will be the bottleneck of the system for the particular algorithm. Objective functions, one for each class of vision algorithms, are proposed in the next section, so that each objective function will measure the quality of a given mapping with respect to the corresponding optimization goal. Two mapping algorithms –one for each objective function– whose goals will be the minimization of the corresponding objective functions, are presented in Section 3.

Before we propose the objective functions that our mapping algorithms will attempt to minimize, let us see the input sets required by these algorithms. There are two sets of inputs, one set corresponding to the source architecture and the other one corresponding to the target architecture. We will start with the set of input values that uniquely describe the structure of the source architecture.

The *set of input values for the source architecture* are:
- n: $2^n x 2^n$ is the number of nodes at the lowest level;
- l: the number of levels;

- an array of values $r_{i,i+1}$, for $0 \leq i \leq l-2$: $r_{i,i+1} x r_{i,i+1}$ is the reduction from level i towards level $i+1$ (this actually implies that the number of nodes at level i will be equal to $2^n x 2^n / (\prod_{j=0}^{i-1} r_{j,j+1} x \prod_{j=0}^{i-1} r_{j,j+1})$).

The *set of input values for the target architecture* are:
- N: $2^N x 2^N$ is the number of PE's at the lowest level;
- L: the number of levels;
- an array of values $R_{i,i+1}$, for $0 \leq i \leq L-2$: $R_{i,i+1} x R_{i,i+1}$ is the reduction from level i towards level $i+1$;
- an array of values P_i, for $0 \leq i \leq L-1$: P_i represents the *processing power* of a single PE at level i.

2.2 THE OBJECTIVE FUNCTIONS

We will first propose the objective function which is appropriate for vision algorithms in class A. As already emphasized in Section 2.1, our main objective for algorithms in class A is the minimization of the total image turnaround time. To achieve our objective, we need to maximize the number of pairs of neighboring nodes (processes) from the source architecture that fall on pairs of directly connected PE's in the target architecture, while the workload (expressed in number of assigned processes) of target PE's will be kept as low as possible.

The objective function that we have chosen to minimize for algorithms in class A, and for systems that contain PE's with identical processing powers, is

$$\sum_{i=0}^{l-1} T_i^2 D_i$$

where

- T_i^2: the number of nodes from level i of the source that are mapped onto a single PE of the target architecture; from now on, we will be calling T_i^2 the *shrinking factor of level i* obtained by the chosen mapping; T_i will represent the shrinking factor of level i for the corresponding one-dimensional system.

- D_i: the distance between any two PE's in the target onto which two (4-connected) neighboring nodes from level i of the source architecture are mapped; if $T_i > 1$ then we assign $D_i = 1$ (not zero), because the maximum distance between "neighbors" will be one (it is based on the fact that if all PE's of a single level are synchronized, then the connection with the largest communication cost determines the overall performance). From now on, we will be calling D_i the *dilation of level i* obtained by the chosen mapping.

The component of the mapping function of Section 2.1 will be then given by: $f_i(z_i) = \lceil z_i D_i / T_i \rceil$, where z_i is either x_i or y_i.

This objective function corresponds to the total image turnaround time, which is the sum of the total computation and the total communication times. The choice of the above objective function as an appropriate qualitative measure of the total image turnaround time on the target architecture is based on the following issues: (1) the total computation time for level i is proportional to T_i^2; (2) the total lateral communication time for level i is proportional to $T_i^2 D_i$; (3) the total vertical communication time between the pair of levels h and i (where h is either $i-1$ or $i+1$) is proportional to $T_i^2 T_h^2$. (largest vertical distance); this distance is a function of D_i and/or D_h; (4) the term $T_i^2 D_i$ dominates the term T_i^2 and is in general a component of the last term; and (5) the deletion of the term that corresponds to the total vertical communication time can be justified based on the following issues: (5.i) the minimization of T_h^2 is taken indirectly into consideration in the minimization of the term $T_h^2 D_h$; (5.ii) our corresponding mapping algorithm has an inherent property that helps to minimize the vertical distances between communicating nodes. More specifically, our algorithm always achieves the following: if there are appropriate reductions in the target architecture, then a particular node from level σ and its children from level $\sigma - 1$ of the source architecture are mapped onto PE ξ_τ of level τ and the set of PE's Ω_φ of level φ respectively in the target architecture, where $\varphi < \tau$, so that the set of PE's Ω_φ are *directly visible* from ξ_τ. The following definition is pertinent.

Definition 3. A set of PE's Ω_φ at a level φ are said to be *directly visible* from PE ξ_τ of a higher level τ, if they are located within a sub-pyramid whose root is ξ_τ. Otherwise (i.e., if there are not appropriate reductions in the target architecture) the node of level σ is mapped in the middle of a square defined by a collection of PE's; the set Ω_φ of PE's are then directly visible from the PE's of that collection.

For systems where the processing powers of PE's may differ between different levels, an assignment with $T_i = D_i = 1$ (which guarantees the smallest possible value for the contribution of level i to the objective function) is chosen, if it exists; otherwise, the inverses of the processing powers of PE's at different levels are used as weights to the corresponding terms $T_i^2 D_i$ in order to decide about the best assignment.

The objective function that we have chosen to minimize for vision algorithms in class B is

$$\max_{j \epsilon F} \{ \sum_{i \epsilon S_j} T_i^2 D_i \}$$

where the sets F and S_j are defined as follows
F={j / a level of the target onto which one or more levels from the source architecture are mapped}, and
S_j={i / a level from the source architecture mapped onto level $j \epsilon F$}. This objective function will be a qualitative measure of the maximum workload for a single level of the target architecture.

Of course, a mapping algorithm whose optimization goal would be the minimization of one of the above objective functions, would normally have much lower complexity than a mapping algorithm whose optimization goal would take the detailed computation and communication requirements of any particular application algorithm –designed for a source architecture– into consideration [Zi89].

3 THE MAPPING ALGORITHMS

We describe in this section our two mapping algorithms –one for each objective function–, whose optimization goals will be the minimization of the objective functions in the preceding section. A theorem is also presented to prove that our mapping algorithms always achieve their goal by yielding optimal solutions for systems composed of identical PE's. Some theorems related to the formulae on which our algorithms are based, along with the corresponding proofs, are presented in the appendix. Let us now present the definition of a term to be used throughout the description of the mapping algorithms.

Definition 4. The *optimal mapping* for level i of the source architecture (if the mapping of level $i-1$ is given) is the mapping that yields $T_i = D_i = 1$; that is, if level i is mapped onto level ζ of the target architecture, then in that case every PE of level ζ will be assigned a single process (i.e., a node) from level i of the source architecture.

The optimal mapping of the above definition does not necessarily exist on a given target architecture for all levels of a given source architecture; the term is used to indicate that our main objective here is to get optimal mapping for each level of the source architecture, because in that case the contribution of every level i to the objective function is minimal (i.e., $T_i^2 D_i = 1$).

We will first present the mapping algorithm for application algorithms in class A. We will then briefly describe the modifications required in that algorithm, in order to get a mapping algorithm to minimize the objective function for application algorithms in class B.

Our mapping algorithm makes use of the following parameters:

- k: the current level of the source architecture;
- K: the current level of the target architecture;

The mapping algorithm is as follows.

STEP 1: (INITIALIZATION)

- Level 0 from the source is mapped onto level 0 of the target architecture. We assume that $N \leq n$. If $N = n$, then we have optimal mapping for level 0 of the source architecture. If $N < n$, then every node of level 0 in the target will simulate an array of $2^{(n-N)}x2^{(n-N)}$ nodes from level 0 of the source architecture, so T_0 is initialized to 2^{n-N}. In both cases D_0 is initialized to 1.
- The parameters k and K are initialized to 1.

STEP 2:

The *optimal reduction factor* for the pair of levels $k-1$ and k of the source architecture, when the parameters of the mapping for level $k-1$ are given (i.e., the reduction in one dimension that yields optimal mapping for level k), will be $\frac{D_{k-1}}{T_{k-1}} r_{k-1,k}$, while the reduction factor that the target architecture supports for the pair of levels $K-1$ and K is $R_{K-1,K}$. We have to distinguish between two cases.

2.a. If $\frac{D_{k-1}}{T_{k-1}} r_{k-1,k} \leq R_{K-1,K}$ (that is, if the reduction factor in the target architecture is greater than or equal to the optimal reduction factor), then one of the following two solutions is chosen, as described below.

2.a.i. If the current level k from the source is mapped onto the current level K of the target architecture, we get

$$T_{k,1} = \frac{R_{K-1,K}}{\frac{D_{k-1}}{T_{k-1}} r_{k-1,k}}$$

and

$$D_{k,1} = 1$$

where $T_{k,1}$ and $D_{k,1}$ represent the values of T_k and D_k if this assignment is chosen. So, each node of level K in the target will be assigned $T_{k,1}^2$ processes from level k of the source architecture by the mapping algorithm in this case.

2.a.ii. However, if the current level k from the source is mapped onto level $K-1$ of the target architecture, we get

$$T_{k,2} = \lceil \frac{T_{k-1}}{r_{k-1,k}} \rceil$$

and

$$D_{k,2} = \lceil \frac{D_{k-1}}{T_{k-1}} r_{k-1,k} \rceil$$

Then, in order to choose from the above two solutions the one that delivers more efficiency, we define the *relative speedup* $RS_{1,2}$ of the first mapping of level k onto level K, versus the second mapping onto level $K-1$, as the ratio

$$RS_{1,2} = \frac{P_K}{P_{K-1}} \frac{T_{k,2}^2 D_{k,2}}{T_{k,1}^2 D_{k,1}}$$

This formula takes the processing powers of PE's at levels $K-1$ and K of the target architecture into consideration. If $RS_{1,2} > 1$, then the first solution can deliver more efficiency than the second one, so level k from the source is mapped onto level K of the target architecture, and we subsequently assign $T_k = T_{k,1}$, $D_k = D_{k,1}$, $k = k+1$, and $K = K+1$. Otherwise (that is, if $RS_{1,2} \leq 1$, level k from the source is mapped onto level $K-1$ of the target architecture and we assign $T_k = T_{k,2}$, $D_k = D_{k,2}$, and $k = k+1$; the PE's of level $K-1$ in the target that are assigned nodes from level k of the source architecture in this case have Cartesian coordinates $(iD_k + \lceil \frac{D_k}{2} \rceil - 1, jD_k + \lceil \frac{D_k}{2} \rceil - 1)$, where i and j are positive integers and k assumes its current value minus 1; the Cartesian coordinates of the PE in the upper-left corner of each level are (0,0). We choose these Cartesian coordinates for PE's to be assigned tasks, because the minimization of average distances between pairs of parent-child nodes is then guaranteed by allocating a parent in the middle of the square defined by its four outermost children.

2.b. If $\frac{D_{k-1}}{T_{k-1}} r_{k-1,k} > R_{K-1,K}$ (that is, if the reduction factor required for the optimal mapping of the current level is greater than the available reduction factor in the target architecture), then level k from the source is mapped onto level λ of the target architecture, where λ is found as shown below.

2.b.i. If there exists a value v (where $v < L$) such that $\frac{D_{k-1}}{T_{k-1}} r_{k-1,k} = \prod_{i=K}^{v} R_{i-1,i}$, then λ will get the value of v and we set $T_k = 1$, $D_k = 1$, $k = k+1$, and $K = \lambda+1$. ($\prod_{i=K}^{v} R_{i-1,i}$ represents the total reduction in one dimension between levels $K-1$ –where K assumes its previous value– and v.) This is the case when optimal mapping for the current level is achieved, because this mapping sets $T_k = D_k = 1$. Levels $k-1$ and k – where k assumes its previous value– from the source will be mapped $(\lambda - K_1 - 1)$ levels away in the target architecture, where K_1 is the level number of the target onto which level $k-1$ from the source architecture is mapped.

2.b.ii. ("Shrinking" in the target architecture.) If $\frac{D_{k-1}}{T_{k-1}} r_{k-1,k} > \prod_{i=K}^{L-1} R_{i-1,i}$ (that is, if the total reduction factor available, which is obtained by combining the reductions of the remaining levels in the target architecture, is smaller than the optimal reduction factor for level k), we then set $\lambda = L-1$ (that is, level k from the source is mapped onto the highest level of the target architecture), $T_k = 1$, $D_k = \frac{\frac{D_{k-1}}{T_{k-1}} r_{k-1,k}}{\prod_{i=K}^{L-1} R_{i-1,i}}$, $k = k+1$, and $K = L$.

2.b.iii. Otherwise, the largest value λ for which $\frac{D_{k-1}}{T_{k-1}} r_{k-1,k} > \prod_{i=K}^{\lambda} R_{i-1,i}$ is found, and we then go to Step 2.a with $r_{k-1,k} = \frac{\frac{D_{k-1}}{T_{k-1}} r_{k-1,k}}{\prod_{i=K}^{\lambda} R_{i-1,i}}$, $T_{k-1} = 1$, $D_{k-1} = 1$, and $K = \lambda + 1$, to choose from the two solutions that map the current level of the source onto levels λ and $\lambda+1$ respectively of the target architecture, the solution that yields the higher speedup.

STEP 3:

3.a. If $k = l$ (i.e., there are no more levels in the source architecture) we then go to Step 4;

3.b. if $K = L$ (i.e., there are no more levels in the target architecture), then the remaining levels from the source are mapped onto the highest level of the target architecture. The formulae of part 2.a.ii will then be applied to find the parameters of these mappings.

3.c. otherwise (if there are more levels in both the source and the target architectures) we go to Step 2 to decide about the assignment of the next level from the source architecture.

STEP 4:

Stop.

The modifications required in the previous algorithm, in order to achieve the minimization of the objective function for application algorithms in class B (which was proposed in Section 2) are as follows.

Change 1. The relative speedup $RS_{1,2}$ in Step 2.a will be given by

$$RS_{1,2} = \frac{P_K}{P_{K-1}} \frac{\sum_{i \epsilon S_{K-1}} T_i^2 D_i + T_{k,2}^2 D_{k,2}}{T_{k,1}^2 D_{k,1}}$$

Change 2. The mapping of Step 2.b.ii is performed "temporarily". The processing continues with the following steps.

(1) Create a set Q of levels from the source architecture, with elements obtained by picking out from every of those levels in the target (except for level 0) onto which one or more levels from the source architecture have already been mapped, the level of the source architecture with the smallest number that has been mapped onto it; **(2)** choose from set Q, level β of the source architecture that has the smallest value for the speedup ratio (with respect to the level it is currently mapped onto, and the immediately lower level in the target architecture); if one or more levels of the source architecture in set Q satisfy the above condition, then choose the one with the largest number in order to achieve low average computation complexity for the mapping algorithm; **(3)** if level β from the source is currently mapped onto level γ of the target architecture, then map level β onto level $\gamma-1$ (the corresponding parameters of this mapping are obtained by the formulae in part 2.a.ii); **(4)** map the remaining levels, as far as level $k-1$, according to the current assignment of level β from the source architecture; and finally, **(5)** choose from the solutions obtained by the two mappings (i.e., the current solution and the solution that was obtained in the beginning of this step of the algorithm) the one that assigns the smaller value to the quantity

$$\max_{\gamma-1 \le j \le \lambda} \{\sum_{i \epsilon S_j} T_i^2 D_i\}$$

(i.e., the part of the objective function that corresponds to different assignments of levels in the two mappings). If the second solution is proven to be better than the first one, then repeat the above steps (steps 1 through 5). The above procedure attempts to perform minimal changes to the previously existing assignment.

Change 3. Step 3.b is modified as follows. If $K = L$ then for the mapping of the next level from the source architecture, the following two solutions are compared: the solution that maps it onto the highest level in the target architecture and the one which is obtained by applying the same technique as in Step 2.b.ii.

The following theorem is relevant to our mapping algorithms.

Theorem 1.

Our mapping algorithms always yield an optimal assignment for systems composed of PE's with identical processing powers; that is, the assignment of nodes from a source architecture to the PE's of a target architecture minimizes the corresponding objective function.

Proof: The proof is omitted for the sake of brevity. □

4 EXAMPLES

Example 1.

Let us now consider a *source architecture* with characteristics

- $l = 7$;
- $r_{0,1} = 8$; $r_{1,2} = 4$; $r_{2,3} = 2$; $r_{3,4} = 4$; $r_{4,5} = 2$; $r_{5,6} = 2$;

while the characteristics of the *target architecture* are

- $L = 3$;
- $R_{0,1} = R_{1,2} = 8$;
- the same processing power for all PE's.

We also assume the same number of nodes in the finest arrays of both structures (i.e., $n = N$). We should remind at this point that the two systems are not necessarily single-rooted in this research; that is, each system is allowed to have more than one PE's in its highest level.

We will consider the minimization of the objective function that corresponds to application algorithms in class A; the result of applying the algorithm is shown in Table 1. The value of the objective function is 35; this means that if PE's with identical processing powers are considered in the source and the target architectures, and there is also uniform distribution of the workload among the levels of the source architecture, then the image turnaround time on the target will be approximately equal to 35/7 times the image turnaround time on the source architecture (because the value of the objective function is 7 for the source architecture). Fig. 2 shows the mapping for levels 4-6 of the source architecture.

Example 2.

Let us now consider a standard pyramid of ten levels (i.e., the size of its finest array is 512x512) as the source architecture, and a target multi-level system of L levels, where $1 \leq L \leq 10$, with reductions 2x2 between all pairs of consecutive levels, and with a size of 512x512 for its finest array. The values assigned to the objective functions by our mapping algorithms, for application algorithms in classes A and B, as well as the average values of the corresponding parameters of mappings, as functions of L, are shown in Fig. 3 and Fig. 4 respectively. The comparison of the values assigned to the two objective functions by the two mapping algorithms is shown in Fig. 5. We conclude from this comparison that maximization of the throughput in cases of pipelining does not necessarily imply minimization of the total image turnaround time; this should be expected, because uniform utilization of the used levels in the target architecture is required in order to achieve high throughput.

5 PERFORMANCE RESULTS

The mapping algorithms of Section 3 were implemented. Code was also developed to find the optimal solutions that minimize the objective functions of Section 2, by applying exhaustive search (i.e., by generating all the possible mappings of levels from the source architecture). A large variation of source and target architecture pairs were considered, and the parameters of the mappings (that were obtained by the application of our mapping algorithms proposed in Section 3) for some examples are shown in Table 2 and Table 3. The processing powers of the PE's at the different levels of the target architectures were chosen to have the same value. The reductions were randomly chosen (however, they were always powers of two). The column of the tables containing the average T_i^2 value represents the average workload, expressed in number of processes (i.e., nodes) from a single level of the source architecture, assigned to a single used PE in the target architecture. The average value of D_i represents the average dilation for a single level of the source architecture. The average value of $T_i^2 D_i$ represents the approximate average expansion of the execution time for a single level of the source architecture, if uniform distribution of the workload among the levels of the source architecture is considered. The last two columns show the overall dissimilarities between the two architectures. The results shown in the tables verify that the more different the target from the source architecture is, the larger the value to be assigned to the corresponding objective function. The execution of the algorithm that performs exhaustive search also verified that, our mapping algorithms each time find the optimal solution that corresponds to mappings at higher levels (i.e., the solution with the maximum value for the sum of j's, where $i \mapsto j$ and i is a level of the source architecture); generally, there may exist more than one solutions that minimize the objective function for class B algorithms. Detailed simulations of some multi-level systems, based on the ERCW (Exclusive Read Concurrent Write) model of parallelism, and executing some vision algorithms (e.g., finding the perimeter of objects, convolution, connected-component labeling, etc.) have shown that the performance obtained by our mapping algorithms is most of the times optimal, and very good in general, when compared with the execution times or throughputs of the optimal mappings, where the latter ones are derived by applying exhaustive search.

6 CONCLUSIONS

Mapping algorithms were presented in this paper for a class of heterogeneous hierarchical systems. It is the first time in parallel computer vision that both the domain and the range of the mapping functions are in a general set of pyramid-like systems. Our mapping algorithms map source architectures, corresponding to application algorithms, onto target architectures. The general optimization goal of our mapping algorithms is to meet the performance requirements of the application algorithms. This is achieved by identifying two classes of vision algorithms for multi-level systems; we distinguish between the two classes based on the required utilization of levels on a multi-level system, which is dictated by high performance issues. Two objective functions, one for each class, are proposed, in order to measure the quality of a given mapping with respect to the corresponding utilization goal. Each mapping algorithm, one for each objective function, yields an optimal mapping by minimizing the corresponding objective function.

It was proven theoretically that our mapping algorithms always yield an optimal solution for systems composed of PE's having identical processing powers; in all other cases, some assignment choices in the algorithms allow to take advantage of the increased processing powers of PE's.

Acknowledgement. The author gratefully acknowledges some very useful discussions with Professors N.A. Alexandridis and L.S. Davis.

References

[AhSw84] N. Ahuja and S. Swamy, "Multiprocessor Pyramid Architectures for Bottom-Up Image Analysis," *IEEE Trans. on Pattern Anal. and Mach. Intel.*, Vol. PAMI-6, No. 4, pp. 463-474, July 1984.

[Bo81] S.H. Bokhari, "On the Mapping Problem," *IEEE Trans. on Comp.*, Vol. C-30, pp. 207-214, Mar. 1981.

[BoMi88] S.W. Bollinger and S.F. Midkiff, "Processor and Link Assignment in Multicomputers Using Simulated Annealing," *in Proc. Intern. Conf. on Paral. Proc.*, pp. 1-7, Aug. 1988.

[CaLe88] V. Cantoni and S. Levialdi, "Multiprocessor Computing for Images," *Proc. IEEE, Special Issue on Comp. Vision*, Vol. 76, No. 8, pp. 959-969, Aug. 1988.

[ChSa86] T.F. Chan and Y. Saad, "Multigrid Algorithms on the Hypercube Multiprocessor," *IEEE Trans. on Comp.*, Vol. C-35, No. 11, pp. 969-977, Nov. 1986.

[ClMe87] P. Clermont and A. Merigot, "Real Time Synchronization in a Multi-SIMD Massively Parallel Machine," *in Proc. Conf. on Arch. for Pattern Anal. and Mach. Intel.*, pp. 131-136, 1987.

[Dy81] C.R. Dyer, "A VLSI Pyramid Machine for Hierarchical Parallel Image Processing," *in Proc. Pattern Recogn. and Image Proc.*, pp. 381-386, 1981.

[Hi85] W.D. Hillis, The Connection Machine, MIT Press, Cambridge, MA, 1985.

[Ka85] M. Katevenis, Reduced Instruction Set Computer Architectures for VLSI, MIT Press, 1985.

[MLL88] M. Mareska, M.A. Lavin and H. Li, "Parallel Architectures for Vision," *Proc. IEEE, Special Issue on Comp. Vision*, Vol. 76, No. 8, pp. 959-969, Aug. 1988.

[Re86] A.P. Reeves, "Pyramid Algorithms on Processor Arrays," in Pyramidal Systems for Computer Vision, V. Cantoni and S. Levialdi (Eds.), Springer-Verlag, New York, NATO ASI Series F, Vol. 25, pp. 195-214, 1986.

[Ro84] A. Rosenfeld (Ed.), Multiresolution Image Processing and Analysis, Springer-Verlag, New York, 1984.

[St86] Q.F. Stout, "Hypercubes and Pyramids," in Pyramidal Systems for Computer Vision, V. Cantoni and S. Levialdi (Eds.), Springer-Verlag, New York, pp. 75-89, 1986.

[Ta83] S.L. Tanimoto, "A Pyramidal Approach to Parallel Processing," *in Proc. 10th Intern. Conf. on Comp. Arch.*, pp. 372-378, 1983.

[Te85] D. Terzopoulos, "Concurrent Multilevel Relaxation," *in Proc. Image Underst. Workshop*, L.S. Baumann (Ed.), Miami Beach, FL, 1985.

[Uh87] L. Uhr (Ed.), Parallel Computer Vision, Academic Press, New York, 1987.

[Zi89] S.G. Ziavras, "Techniques for Efficient Mappings of Deterministic Algorithms onto Multi-Level Systems," *in preparation.*

TABLE 1.
Parameters of mapping for "Example 1."

Level i of source	Level j of target with i ↦ j	T_i	D_i
0	0	1	1
1	1	1	1
2	1	1	4
3	2	1	1
4	2	1	4
5	2	1	8
6	2	1	16

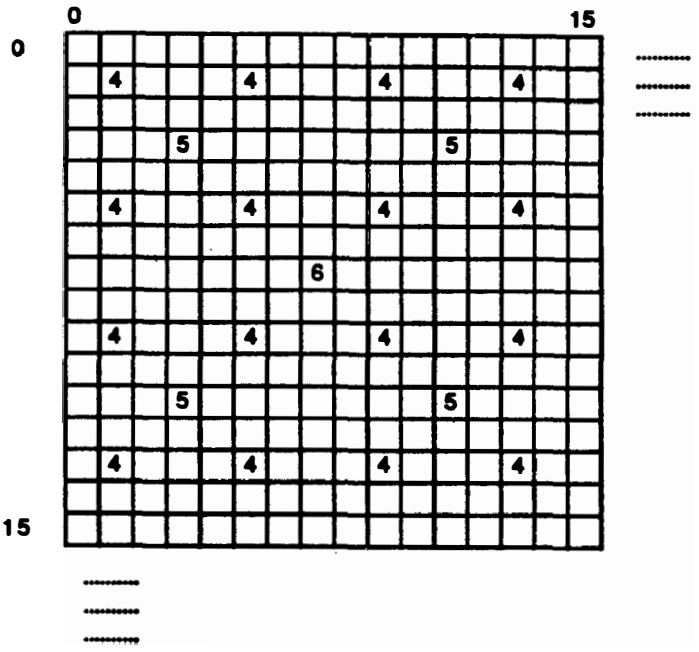

Fig. 2. The mapping of levels 4-6 from the source architecture, for "Example 1."

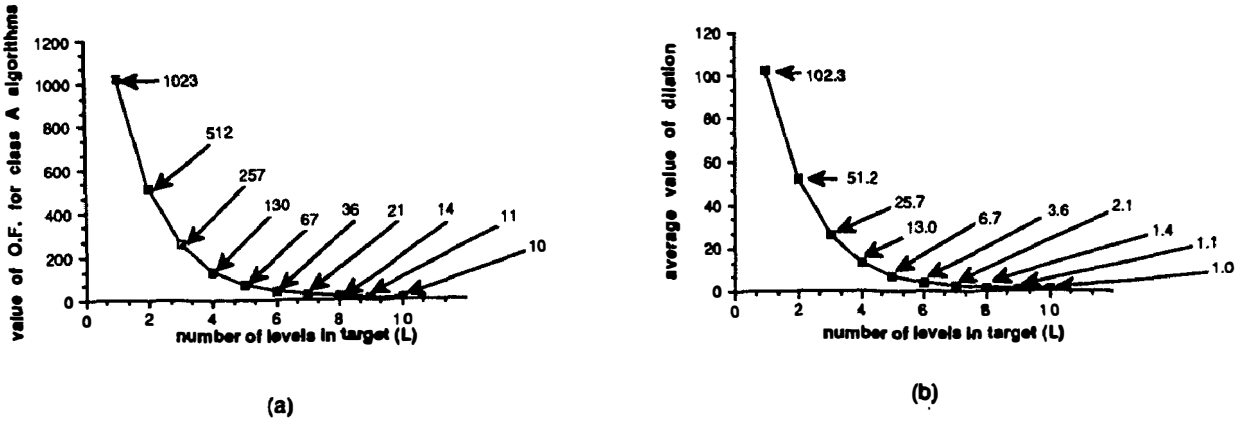

Fig. 3. Minimizing the O.F. for class A algorithms in "Example 2." **(a)** The value of the O.F.; **(b)** the average value of the dilation. T_i is 1 for all values of i. (O.F.: Objective Function.)

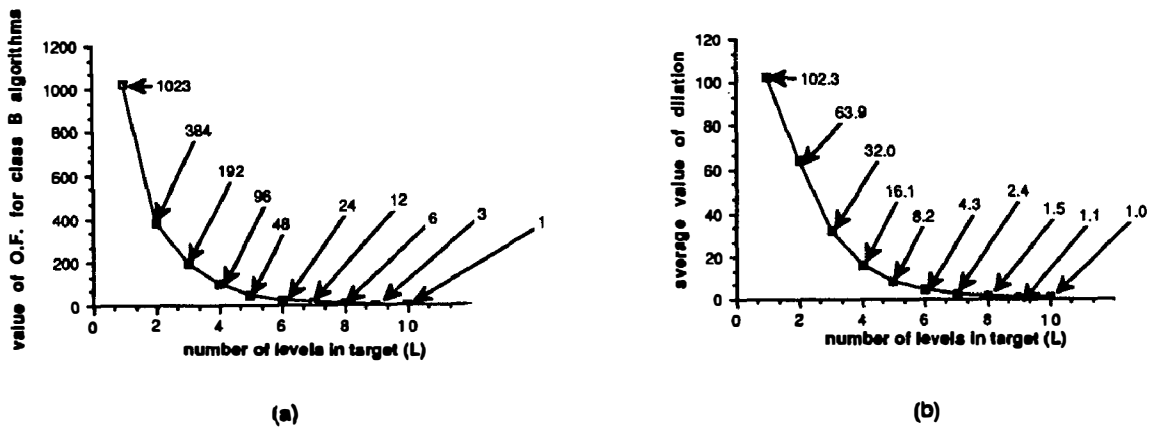

Fig. 4. Minimizing the O.F. for class B algorithms in "Example 2." **(a)** The value of the O.F.; **(b)** the average value of the dilation. T_i is 1 for all values of i.

TABLE 2.
Performance results for class A application algorithms.

Average D_i	Average T_i^2	Value of Normalized objective function. Average $T_i^2 D_i$	Number of levels in source - total reduction in one dimension	Number of used levels in target - total reduction in one dimension for used levels
1.66	1.33	2.00	9 - 256	4 - 64
2.16	1.50	2.66	6 - 1024	3 -1024
1.88	1.38	2.26	8 - 1024	4 - 512
2.14	1.42	2.57	7 - 512	3 - 128
2.22	1.33	2.55	9 - 2048	3 - 512
2.66	1.66	3.33	9 - 256	3 - 512
7.77	1.00	7.77	9 - 256	2 - 8
4.33	1.33	4.66	9 - 256	2 - 16
8.14	1.00	8.14	7 - 512	3 - 16
7.33	1.00	7.33	9 - 1024	4 - 32
5.00	1.00	5.00	7 - 1024	3 - 64
1.50	1.00	1.50	2 - 2	1 - 0
4.50	1.00	4.50	6 - 256	3 - 16
4.33	1.33	4.66	9 - 256	2 - 64
2.80	1.60	3.40	5 - 1024	3 - 2048
1.20	4.00	4.20	5 - 1024	4 - 4096
1.80	4.00	4.80	5 - 4096	4 - 16384
1.50	1.00	1.50	4 - 32	4 - 32
2.20	1.00	2.20	5 - 64	3 - 16
		Average values		
3.42	1.52	3.94	6.84 - 786.60	3.05 - 1344.40

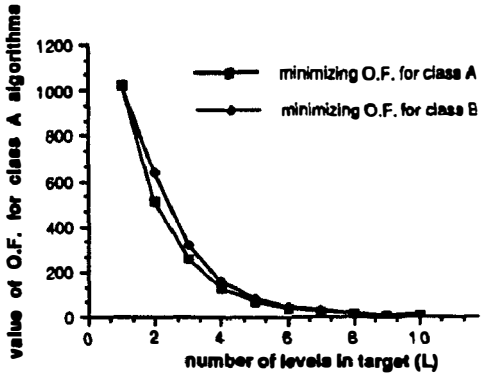

Fig. 5. Comparing the two mappings in "Example 2."

TABLE 3.
Performance results for class B application algorithms.

Average D_i	Average T_i^2	Value assigned to the objective function	Average $T_i^2 D_i$	Number of levels in source - total reduction in one dimension	Number of used levels in target - total reduction in one dimension
2.00	1.00	7	2.00	9 - 256	4 - 64
2.17	1.50	10	2.67	6 - 1024	3 - 1024
1.67	1.38	7	2.25	6 - 1024	4 - 512
2.14	1.43	10	2.57	7 - 512	3 - 128
2.22	1.33	11	2.55	9 - 2048	3 - 512
3.44	1.33	15	3.77	9 - 256	3 - 512
10.11	1.00	60	10.11	9 - 256	2 - 8
5.11	1.00	31	5.11	9 - 256	2 - 16
13.14	1.00	48	13.14	7 - 512	3 - 16
6.77	1.00	48	6.77	9 - 1024	4 - 32
6.00	1.00	28	6.00	7 - 1024	3 - 64
1.50	1.00	3	1.50	2 - 2	1 - 0
5.50	1.00	20	5.50	6 - 256	3 - 16
2.67	3.00	27	4.67	9 - 256	2 - 64
2.80	1.60	11	3.40	5 - 1024	3 - 2048
1.20	4.00	16	4.20	5 - 1024	4 - 4096
1.80	4.00	16	4.80	5 - 4096	4 - 16384
1.50	1.00	2	1.50	4 - 32	4 - 32
2.20	1.00	8	2.60	5 - 64	3 - 16
			Average values		
4.00	1.56	19.79	4.56	6.84 - 786.60	3.05 -1344.40

APPENDIX

Theorem 2.

If the mapping of level $k-1$ from the source architecture yields (T_{k-1}, D_{k-1}) and the reduction between the pair of levels $k-1$ and k in the source is $r_{k-1,k} x r_{k-1,k}$, then the required reduction in the target architecture, which would yield optimal mapping for level k, that is a mapping with parameters $(T_k, D_k) = (1, 1)$, will be

$$\frac{D_{k-1}}{T_{k-1}} r_{k-1,k} x \frac{D_{k-1}}{T_{k-1}} r_{k-1,k}$$

Theorem 3.

If the mapping of level $k-1$ from the source, onto level $K-1$ of the target architecture, yields (T_{k-1}, D_{k-1}), the reduction between the pair of levels $k-1$ and k in the source is $r_{k-1,k} x r_{k-1,k}$, and the available reduction $R_{K-1,K} x R_{K-1,K}$ in the target architecture is greater than or equal to the optimal reduction, then the mapping of level k from the source onto level K of the target architecture will yield

$$T_k = \frac{R_{K-1,K}}{\frac{D_{k-1}}{T_{k-1}} r_{k-1,k}}$$

and $D_k = 1$.

Theorem 4.

If, for the same data as in Theorem 3, level k from the source is mapped onto level $K-1$ of the target architecture, the parameters will be

$$T_k = \lceil \frac{T_{k-1}}{r_{k-1,k}} \rceil$$

and

$$D_k = \lceil \frac{D_{k-1}}{T_{k-1}} r_{k-1,k} \rceil$$

Theorem 5.

If the mapping of level $k-1$ from the source architecture yields (T_{k-1}, D_{k-1}), the total reduction in one dimension between the level onto which level $k-1$ is mapped and another higher level v in the target architecture is τ, and the reduction required for the optimal mapping of level k is greater than τ, then if level k from the source is mapped onto level v of the target architecture, we will get

$$T_k = 1$$

and

$$D_k = \frac{\frac{D_{k-1}}{T_{k-1}} r_{k-1,k}}{\tau}$$