Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Multilevel Embeddings
for
Massively - Parallel Hypercubes

by
Devenkumar P. Shah

Hierarchically structured arrays of processors have widely been used in the low and intermediate phases of image processing and computer vision. Since the pyramid structure efficiently supports local and global operations extensively required by these phases, it has been widely used for relevant algorithms. Multilevel systems keep all the advantages of the pyramid structure while providing a general hierarchical structure that is easier to be used for the development of several algorithms and may also provide higher performance.

Although the cost of pyramid machines may be tremendously high, they have limited applications. In contrast, the hypercube network is widely used in the field of parallel processing because of its small diameter and its rich interconnection structure. Several algorithms have been developed that embed pyramids into the hypercube. This thesis extends and also implements three pyramid embedding algorithms in order to embed multilevel structures into the hypercube. These embedding algorithms are evaluated for the Connection Machine system CM-2 that comprises a 10-dimensional hypercube. The results for multilevel structures are compared with those for the pyramid; three image processing algorithms are used for this purpose. The results prove that the embedding of multilevel structures, other than the pyramid, yields better performance in most of the cases.

This thesis also studies the implementation of stochastic pyramids on the hypercube. The structure of stochastic pyramids adapt to the contents of the image. Therefore artifacts present in the regular pyramid due to its rigid structure are alleviated. Connection Machine results are produced for the problem of connected component extraction using stochastic pyramids.

MULTILEVEL EMBEDDINGS FOR MASSIVELY-PARALLEL HYPERCUBES

by
Devenkumar P. Shah

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

May 1993

APPROVAL PAGE

Multilevel Embeddings
For
Massively-Parallel Hypercubes

Devenkumar P. Shah

Dr. Sotirios G. Ziavras, Thesis Adviser. date
Assistant Professor, ECE Department,
New Jersey Institute of Technology, Newark, NJ.

Dr. John Carpinelli, Committee Member. date
Assistant Professor, ECE Department,
New Jersey Institute of Technology, Newark, NJ.

Dr. Bruce Parker, Committee Member. date
Assistant Professor, CIS Department,
New Jersey Institute of Technology, Newark, NJ.

BIOGRAPHICAL SKETCH

Author: Devenkumar P. Shah

Degree: Master of Science in Electrical Engineering

Date: May 1993

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ, 1993
- Bachelor of Engineering (Electrical)
  Sardar Patel University, Vallabh Vidyanagar, India, 1991
- Diploma in Electrical Engineering
  B. & B. Polytechnic College, Vallabh Vidyanagar, India, 1987

Major: Computer Engineering

Presentations and Publications:

Ziavras, S. G., and D. P. Shah, "High Performance Emulation of Hierarchical Structures on Hypercube Supercomputers." *Concurrency:Practice and Experience*, Accepted for publication.

This thesis is dedicated to
my family.

ACKNOWLEDGMENT

I express my sincere appreciation to my thesis advisor, Dr. Sotirios G. Ziavras, for his guidance, suggestions, and support.

Special thanks to Dr. John Carpinelli and Dr. Bruce Parker for serving as members of the committee.

The permission to access the Connection Machine CM-2 System at UMIACS is greatly appreciated.

I thank Dr. B. T. Desai for the inspiration.

Raju Patel, Krishna Patel, and Kaushika Narola provided great moral support and help.

TABLE OF CONTENTS

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Parallel Techniques for Image Processing

Image processing and computer vision algorithms employ techniques from several other areas, such as signal processing, graph theory, advanced mathematics, and artificial intelligence. These algorithms require tremendous computational power. The cost of performing these algorithms sequentially can be very high.

Parallel processing is widely used to providing the necessary computational power because the inherent parallelism in most of the image processing algorithms can be exploited in order to process the image data in real time. The data domain of image processing algorithms can often be partitioned into blocks that can be operated upon in parallel. Time spent in data exchanges between related computations should be minimized through the communication facilities of the target parallel architecture.

1.2 Multilevel Structures

The pyramid is a special case of a multilevel structure. A multilevel system is composed of successive layers of mesh-connected two-dimensional arrays where the size of the arrays decreases with the increase in the level number (assume that the base is level 0). The reductions between pairs of neighboring layers are 2^m x 2^m, where m is a natural number. Each node at any level, except for nodes in the base, is directly connected to 2^m x 2^m children located at the immediately lower level. Multilevel structures are not necessarily single rooted [1]. The pyramids are special case of multilevel structures with a single root and m equal to 1.

Like the pyramid, multilevel structures also efficiently implement local and global

operations. This makes them equally capable for the low and intermediate phases of image processing and computer vision. Two reasons make multilevel structures very attractive [1]. The development of algorithms is very often easier for multilevel structures other than the pyramid. In addition, higher performance is often obtained by embedding a general multilevel structure into the hupercube [12].

1.3 Motivations and Objectives

The hypercube topology has a small diameter and a rich interconnection structure that provides efficient communication and a high degree of fault tolerance. It can very efficiently emulate a wide variety of other frequently used topologies such as the mesh, the binary tree and the pyramid. As a consequence, hypercube machines have become commercially available (e.g., Intel iSPC, nCUBE, Connection Machine system CM-2).

Although multilevel structures are appropriate for the low and the intermediate phases of computer vision, this topology may not be a good choice for the implementation of parallel computers because of its high cost and limited functionality (i.e. special purpose topology). In contrast, the hypercube is considered to be a "general purpose" topology. Therefore, it becomes imperative to develop techniques for mapping multilevel structures onto the hypercube. All the algorithms developed so far that map the pyramid structure onto the hypercube can be easily extended to map the multilevel structures. It is important to show that higher performance can often be obtained by mapping multilevel structures, other than the pyramid, onto the hypercube.

The major objective of this research is to expand three important pyramid mapping algorithms to make them appropriate for general multilevel structures. The evaluation of the performance for specific image processing application algorithms designed for multilevel architechtures, such as finding the perimeter of objects, segmentation of images, and convolution, can show that the pyramid is not necessarily the most efficient source architecture when compared to general multilevel structures. We present results for a Connection Machine system CM-2 with 16,384 processors that comprises a *10*-dimensional

hypercube .

A major limitation of pyramids is the rigidity of their structure that gives rise to artifacts when used for tasks such as analysis of line drawings, object-background discrimination or compact object extraction. An approach to compensating for the artifacts of the pyramid structure is to adapt this structure to the contents of the input image. Stochastic pyramids are built with this in mind using a local stochastic decimation process that builds the multiple resolution representations [5]. Also, evolution of the local connections within the hierarchy driven by the image data and the stochastic process might serve as a model for early visual perception. This thesis also studies implementation of these stochastic pyramids on the hypercube. The connected component analysis of binary labeled images is done by developing stochastic pyramids on the Connection Machine.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 describes the hypercube topology, techniques for embedding various topologies into the hypercube and the important features of the Connection Machine system CM-2. Chapter 3 presents algorithms for mapping multilevel structures onto the hypercube and introduces measures used for performance analysis. Chapter 4 presents our results and also includes a comparative analysis of the mapping algorithms for the Connection Machine. Chapter 5 discusses the implementation of stochastic pyramids on the Connection Machine. Finally Chapter 6 presents our conclusions.

CHAPTER 2

THE HYPERCUBE

The hypercube interconnection topology is general enough to be useful on a rich variety of problems, yet practical enough that a real concurrent processor system can be built with the topology. Most of the other topologies like the rings, linear arrays, tori, binary trees, and pyramids can easily be mapped onto the hypercube. So, applications designed for the latter topologies can easily be run by simulating the respective topology on the hypercube. In the hypercube network, the number of communication channels per node is in logarithmic order of the number of processors and so remains within practical limits. The maximum distance between nodes (i.e., the diameter) also varies logarithmically with the number of nodes. Thus, the hypercube provides a good balance between the number of channels per node and the maximum distance between nodes.

Hypercube computers are loosely coupled parallel processing systems based on the binary *d*-cube network. The Connection Machine system, used to derive results in this thesis, has been one of the most popular hypercube systems and is manufactured by Thinking Machines Corporation. It operates in the SIMD (Single-Instruction Stream Multiple- Data Stream) mode of computation.

2.1 Topology

The *d*-dimensional (binary) hypercube (H_d) or *d*-cube has 2^d nodes, each of which is connected to one other node in each dimension. Thus, the required number of channels per node is *d*. Each node has a unique *d*-bit binary address. The i^{th} bit of the address represents the coordinate of the node in the i^{th} dimension. All neighboring nodes connected by a communication channel differ by a single bit in their addresses. The number of bits that differ between the addresses of two nodes gives the distance between the two nodes. The hypercube can be partitioned into smaller dimensional hypercubes and the *d*-dimensional

hypercube can be constructed recursively from lower dimensional subcubes. For example two (d-1)-dimensional hypercubes can be combined to form a d-dimensional hypercube by joining the corresponding vertices of two subcubes by 2^{d-1} links.

In a hypercube computer, processing elements (PEs) are placed at each vertex of the hypercube and the edges of the hypercube represent communication links between the PEs. Each PE has its own local memory. In SIMD mode, this memory contains only data whereas in MIMD (Multiple-Instruction Stream Multiple-Data Stream) mode, it contains data as well as instructions.

2.2 Mappings for Hypercube Systems

Most of the popular parallel processing architectures such as rings, linear arrays, meshes, tori, binary trees and pyramids can easily be mapped onto the hypercube. Such mappings are important because algorithms developed for other special purpose topologies can easily be run on hypercube systems by mapping such topologies onto the hypercube.

2.2.1 Mapping Rings onto the Hypercube

Consider a ring with 2^d PEs. It can be mapped onto the d-dimensional hypercube by mapping adjacent nodes of the ring onto adjacent nodes of the hypercube. Since in the hypercube network, any two adjacent nodes have their binary addresses differing by only one bit, this means that the d-bit hypercube node addresses must be listed in sequence in such a way that any two successive addresses differ by a single bit. A binary sequence with this property is the binary Reflected Gray Code (RGC).

The mapping of the 3-dimensional hypercube into the 8-node ring is shown in Figure 2.1.

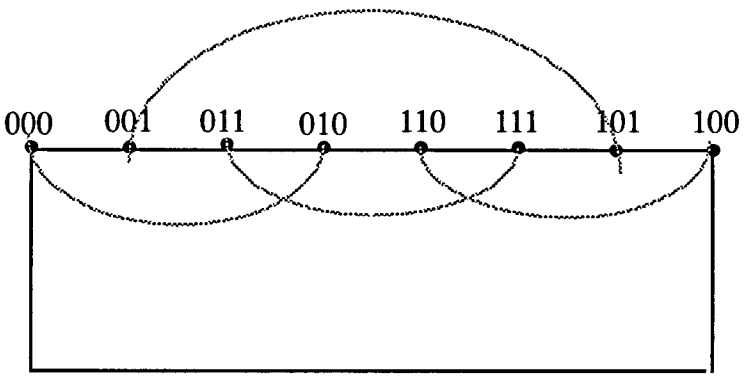


Figure 2.1 : Mapping the H_3 Hypercube into the 8-node Ring.

2.2.2 Mapping the Mesh onto the Hypercube

Consider an *n*-dimensional mesh that has size m_i in dimension *i*, where m_i is a power of 2 such that $p_i = \log_2 m_i$. This mesh can be mapped onto the *d*-dimensional hypercube where $d = p_1 + p_2 + p_3 + ... + p_n$. The algorithm to map the mesh onto the hypercube is a direct extension of the previously presented algorithm for the ring. The nodes in each dimension are numbered sequentially using the RGC. A node of the mesh is mapped onto the node of the hypercube whose address is obtained by concatenating the RGCs of the particular node for all dimensions.

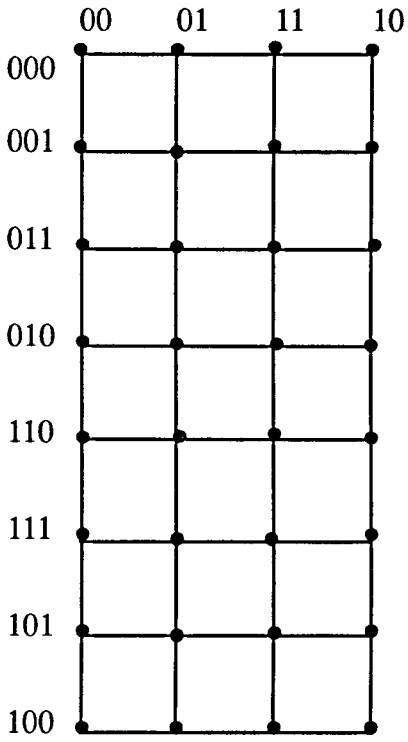
Figure 2.2 shows the mapping of the 5-dimensional hypercube into the 8 x 4 mesh and the 4 x 4 x 2 mesh.

2.3 A Case Study: The Connection Machine System CM-2

The Connection Machine model CM-2 is a distributed memory, SIMD, Concurrent-Read Exclusive-Write (CREW) parallel computer [10]. The system includes front-end computers that provide development and execution environments for the system software, a parallel processing unit of up to 64K processors that execute data parallel operations, and a high performance data parallel I/O system. The system software is based upon the environment of the front-end computer.

2.3.1 Front-end Interface

The front-end computer is the gateway to the Connection Machine system. It provides software development tools, software debugging tools, and a program execution environment familiar to the user. The front-end should contain specialized hardware, called a Front-End Bus Interface (FEBI), which allows communication with the Connection Machine processors. At the present, FEBI is available for most DEC Vax 8000 series minicomputers, Sun workstations, and Symbolics 3600 series Lisp machines.



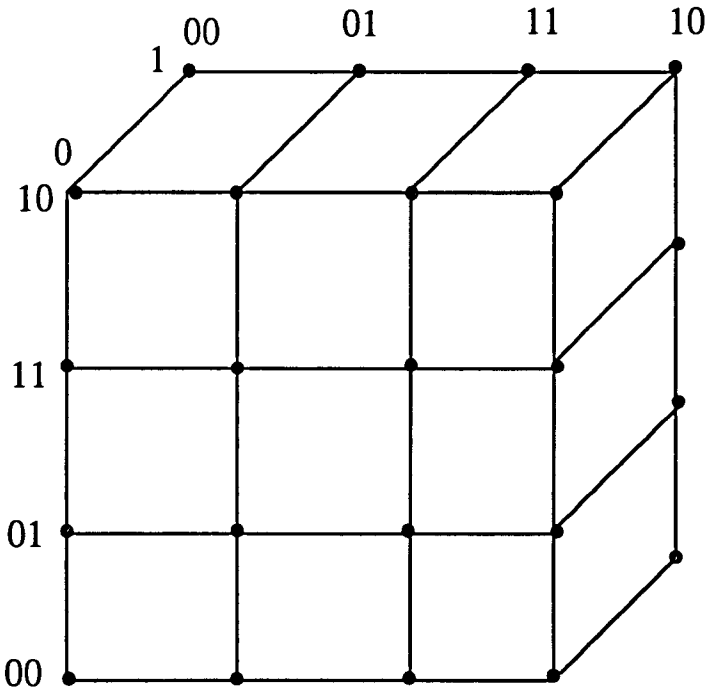


Figure 2.2 : Mapping the H_5 Hypercube into the 8 x 4 and the 4 x 4 x Meshes.

The data set, for the most part, is stored in the Connection Machine memory. In this way, the entire dataset can be operated upon in parallel through commands sent to the Connection Machine processors by the front-end. The part of the program code for any application that pertains to the interfaces between the program, the user, and the operating system is handled by the front-end. For program steps that involve parallel data, the front-end interacts with the Connection Machine parallel processing unit using an integrated command set, where commands are broadcast for execution by all processors all at once.

Thus, the front-end performs three primary functions in the Connection Machine system.

- It provides an application development and debugging environment.
- It runs applications and transmits the instructions and associated data to the Connection Machine parallel processing unit.
- It provides maintenance and operation utilities for controlling the Connection Machine and diagnosing problems.

2.3.2 System Organization

The central element in the system is the CM-2 *parallel processing unit*, which contains:

- thousands of *data processors;*
- an interprocess communication network;
- one or more *sequencers;*
- an interface to one or more front ends;
- and zero or more I/O controllers and/or framebuffers.

The parallel processing unit may contain 64 K, 32 K, or 16 K data processors. Each data processor has 64 K bits of bit-addressable local memory, an arithmetic-logic unit with variable length operands, and an optional parallel floating point accelerator. There are two forms of communications available within the unit. The more general mechanism is known as router, which allows any processor to communicate with any other processor in the system. The other, a more structured, somewhat faster communication

mechanism is the NEWS grid. It allows processors to pass data according to a regular rectangular pattern. These grids are programmable to have variable numbers of dimensions. Possible grid configurations for 64 K processors include 256 x 256, 8 x 8192, 64 x 32 x32, 16 x 16 X 16 x 16, and 8 x 8 x 4 x 8 x 8 x 4.

The set of instructions that the front-end may issue to the parallel processing unit is called Paris. Paris instructions from the front-end are processed by a sequencer in the parallel processing unit. The task of the sequencer is to break down each Paris instruction into a sequence of low level data processor and memory operations. The sequencer broadcasts these low-level operations to the data processors, which execute them at a rate of several millions per second. There is a sequencer for each 8 K of processors in the system.

The CM-2 system used for this thesis has 16 K processors and two sequencers that can be ganged together.

2.3.3 Software Support

Since the development environment is provided by the front-end and the editors, the file-system used for the software development and maintenance remain that of the front-end. The Connection Machine system software is designed to utilize the existing programming languages with minimal extensions required to support data parallel constructs. The program code for this thesis was developed using the C* language, which is a data parallel extension of the C language. The target language of the high level compilers is Paris, the assembly language of CM-2.

2.3.4 The Router

The router allows any processor of the parallel processing unit to communicate with any other processor; with all processors acting at the same time, the local memories of data processors are treated as a single large shared memory. Each CM-2 processor chip con-

tains one router node which serves the 16 processors on the chip. The router nodes on all processor chips are wired together to form the complete hypercube router network. With 16 K processors in the system, the router network is a 10-cube connecting, 1,024 processor chips, that is there are 16 processors per router node.

Each message travels from one router node to another until it reaches the chip containing the destination processor. The algorithm used by the router can be divided into stages called petit cycles. The delivery of all the messages for a Paris operation might require only one petit cycle if only a few processors are active, but if every processor is active, then typically many petit cycles are required. If the message can not be delivered to the destination processor in a single petit cycle, it is buffered in whichever intermediate chip it has reached at the end of the petit cycle. For this work, each application code used all 64 K processors (i.e. 16, PEs per router) and 1 K processors (i.e., 1 PE per router) by selectively disabling some of the processors. The latter case represents a perfect *10*-dimensional hypercube.

CHAPTER 3

MAPPING MULTILEVEL STRUCTURES ONTO THE HYPERCUBE

The robust interconnection structure of the hypercube can efficiently emulate multilevel structures. Several algorithms have been published for the emulation of pyramids on hypercubes. Stout has proposed an algorithm to map pyramids onto the hypercube [2]. Patel and Ziavras have proposed an algorithm to map pyramid and multilevel structures onto the hypercube [3]. Ho and Johnsson have proposed an algorithm to map hyper-pyramids onto the hypercube [4]. This thesis extends pyramid mapping algorithms for multilevel structures and implements all these algorithms on the Connection Machine for the purpose of performance comparison.

3.1 Problem Definition and Performance Measures

A multilevel structure is embedded into the hypercube topology by mapping the vertices of the former to the vertices of the latter in a one-to-one or many-to-one fashion.

Let the function $h: G \rightarrow G'$ represent the mapping of the vertices of the source graph G to the vertices of G' in one-to-one fashion. Three performance measures namely expansion, dilation and congestion [11] which are defined in the following three subsections can be used for a first level comparative analysis of the mapping algorithms.

3.1.1 Expansion

The expansion of h is the ratio $(|V(G')|/|V(G)|)$, where $V(G)$ and $V(G')$ are the vertex sets of G and G' respectively, and $|V(G)|$ and $|V(G')|$ are the numbers of vertices in those sets. This measure gives the relative cost of the target system with respect to the source system,

assuming that the cost of a system that implements a particular graph is proportional to the total number of nodes in the graph. Ideally, the value of this measure should be one. The closer the value of this measure to one, the smaller is the portion of unused resources in G'.

3.1.2 Dilation

When two neighboring nodes from G are mapped onto two distinct nodes in G', the dilation of the edge connecting the two nodes in G is the length of the corresponding path in G'. The maximum dilation is the maximum length of any such path in G'. The dilation measures the relative communication overhead on the target architecture. The smaller the value of the dilation, the smaller is the communication overhead.

Multilevel structures can not be mapped onto the hypercube with maximum dilation one. The reason is that a cycle in a multilevel structure consisting of a parent and two of its children contains an odd number of nodes, whereas in a hypercube all cycles contain an even number of nodes [2].

3.1.3 Congestion

The congestion is the number of edges in G that are mapped onto the same edge in G'. The maximum number of edges in G with the same image in G' is the maximum value of the congestion for the chosen mapping h. The smaller the value of the congestion, the less amount of time that messages will have to wait in queues at intermediate PEs for communication channels to become available.

3.2 Stout Algorithm (Algorithm I)

3.2.1 Algorithm for Pyramid Mapping

This algorithm embeds the pyramid with base of size 2^n x 2^n (P_n) into the H_{2n} hypercube [2]. The number of nodes in the hypercube is equal to the number of nodes in the base of the pyramid. Since a pyramid with a base of size 2^n x 2^n contains a total of $\lfloor \frac{2^{2(n+1)}}{3} \rfloor$ nodes, the expansion is less than one. For one-to-one mapping of base (say level 0) nodes onto PEs of the hypercube, the *n*-bit RGC is used to encode separately the rows and columns in the base. The binary addresses of the respective PEs in the hypercube are found by concatenating the bits of the encoded row and column numbers. This process produces a perfect mapping for the base of the pyramid; i.e., all three measures associated with the cost of the base mapping are optimal (i. e., they are equal to one).

Every node at the immediately higher level of the pyramid (i.e., level 1) has four children at the leaf level (i.e., level 0). The PE, from each square of four PEs, that has the least significant bit of the respective row and column numbers equal to 0 is chosen to represent a parent at level number 1. In general, PEs having the lower *k* bits of their encoded row and column numbers equal to 0 are chosen to simulate nodes from level *k* of the pyramid. Figure 3.1 shows the mapping of the H_6 hypercube into the P_3 pyramid. Thus, one of the children will use two communication links for data exchanges with its parent, and the maximum dilation of the mapping is two. The two significant advantages of this mapping are the smallest possible resultant dilation and the small expansion (i.e., less than one).

Since some hypercube PEs emulate nodes from several levels of the pyramid, this mapping algorithm does not allow the simultaneous emulation of multiple pyramid levels. So, it cannot support concurrent multilevel processing (e. g., pipelining).

3.2.2 Extension for Mapping Multilevel Structures

The following rule is applied for extending the mapping for the multilevel structure with

RGC	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0 0	0,1,2,3	0	0	0,1	0,1	0	0	0,1,2
0 0 1	0	0	0	0	0	0	0	0
0 1 1	0	0	0	0	0	0	0	0
0 1 0	0,1	0	0	0,1	0,1	0	0	0,1
1 1 0	0,1	0	0	0,1	0,1	0	0	0,1
1 1 1	0	0	0	0	0	0	0	0
1 0 1	0	0	0	0	0	0	0	0
1 0 0	0,1,2	0	0	0,1	0,1	0	0	0,1,2

Figure 3.1: Mapping the H_6 Hypercube into the P_3 Pyramid using Algorithm I.

RGC	000	001	011	010	110	111	101	100
000	0,1,2	0	0	0	0	0	0	0,1
001	0	0	0	0	0	0	0	0
011	0	0	0	0	0	0	0	0
010	0	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0
100	0,1	0	0	0	0	0	0	0,1

Figure 3.2 : Mapping the H_6 Hypercube into M_3 Multilevel Structure with Reductions of 4 x 4 and 2 x 2 using Algorithm I.

the base of size 2^n X 2^n (M_n). Level *j* of the multilevel structure is mapped to level *k* of the pyramid, where *k* is obtained by the following equation.

$$k = \sum_{i=1}^{j} m_i$$

and 2^{m_i} is the reduction between level *i-1* and *i* .

The resulting version of the algorithm results in maximum dilation $2m_i$ for the pair of levels *i-1* and *i*. Therefore, the maximum dilation of this extended mapping algorithm is $\max\{2m_i\}$, for i = 1,2,3,... . The extended algorithm does not support the efficient implementation of algorithms that apply concurrent multilevel processing. Figure 3.2 shows an example.

3.3 Patel and Ziavras Algorithm (Algorithm II)

3.3.1 Algorithm for Pyramid Mapping

Similarly to Algorithm I, this algorithm maps the P_n pyramid onto the H_{2n} hypercube [3]. Although the expansion is less than one, in contrast to the previous algorithm, it allows any subset of levels, excluding the base, to be active simultaneously. The emulation of the base excludes the simultaneous emulation of any other level in the pyramid because the total number of base nodes is the same as the total number of PEs in the hypercube.

The algorithm operates as follows. Similarly to Algorithm I, the RGC is used to independently encode rows and columns in the base. A perfect mapping is then produced for the base by concatenating the bits of the encoded row and column numbers in order to find the address of target PEs in the hypercube. The mapping of level 1 nodes is also similar to the mapping produced by the previous algorithm. The PEs chosen to emulate parents of leaf nodes have the least significant bit of their encoded row and column equal to 0. For each set of four PEs at level 1 with common parent, the PE chosen to serve as the parent is neighbor to one of the PEs representing the children and all PEs at level 2 form mirror images in squares outlined by their children. This procedure is repeated until the apex of

the pyramid is reached.

For example, Figure 3.3 shows the one-to-one assignment of leaf nodes from the P_3 pyramid to the H_6 hypercube. There are sixteen groups (squares) of 2 x 2 PEs in the base with common parent. The parent at level 1 for the children in such a square is emulated by the PEs marked with 1 in the square. The PEs marked with 1 are again grouped into quadruples with common parent. Parents at the next higher level are emulated by the PEs marked with 3. Thus PEs marked with 0, 1, 2, and 3 emulate nodes from levels 0, 1, 2 and 3 of the P_3 pyramid, respectively. Any subset of pyramid levels that does not include the base can be emulated simultaneously.

The dilation of the mapping for an edge connecting a parent at level 1 and one of its children is two. However, the maximum dilation for higher levels is equal to three. To conclude, this mapping algorithm yields maximum dilation three.

3.3.2 Extension for Mapping Multilevel Structures

The mapping for the base is identical to that for pyramid mapping. For mapping level *j* nodes, a subset of the PEs that emulate level *j* nodes of the pyramid are used. The chosen PEs should be as far apart as possible in the 2^n x 2^n base array. The maximum dilation is equal to $2\max\{m_i\} + 1$, where all possible values of *i* are considered.

Figure 3.4 shows the mapping of the H_6 into a case of M_3 where the reductions are 4 x 4 and 2 x 2 between levels 0 to 1, and 1 to 2, respectively. The algorithm still allows any subset of levels, excluding the base, to be active simultaneously.

3.4 Ho and Johnsson Algorithm (Algorithm III)

3.4.1 Algorithm for Mapping Hyper-Pyramids

A $\hat{P}$(k , d) hyper-pyramid is a level structure of *k* boolean cubes where the cube at level *i* is of dimension *id*, and (assuming that the apex is level 0) a node at level *i - 1* connects to every node in a *d*-dimensional Boolean subcube at level *i*, except for the leaf level *k*.

RGC	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0 0	0,1	0,2	0,3	0,1	0,1	0	0,2	0,1
0 0 1	0	0	0	0	0	0	0	0
0 1 1	0	0	0	0	0	0	0	0
0 1 0	0,1	0	0	0,1	0,1	0	0	0,1
1 1 0	0,1	0	0	0,1	0,1	0	0	0,1
1 1 1	0	0	0	0	0	0	0	0
1 0 1	0	0	0	0	0	0	0	0
1 0 0	0,1	0,2	0	0,1	0,1	0	0,2	0,1

Figure 3.3 : Mapping the H_6 Hypercube into the P_3 Pyramid using Algorithm II.

RGC	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0 0	0,1	0,2	0	0	0	0	0	0,1
0 0 1	0	0	0	0	0	0	0	0
0 1 1	0	0	0	0	0	0	0	0
0 1 0	0	0	0	0	0	0	0	0
1 1 0	0	0	0	0	0	0	0	0
1 1 1	0	0	0	0	0	0	0	0
1 0 1	0	0	0	0	0	0	0	0
1 0 0	0,1	0	0	0	0	0	0	0,1

Figure 3.4 : Mapping the H_6 Hypercube into the M_3 Multilevel Structure with Reductions of 4 x 4 and 2 x 2 using Algorithm II.

Hyper-pyramids contain pyramids as proper subgraphs [4].

This algorithm embeds the $\hat{P}(k , d)$ hyper-pyramid into the H_{2d+1} hypercube with maximum dilation *d*. Since a pyramid is a subgraph of a hyper-pyramid with $d = 2$, this mapping algorithm can also be used for mapping the P_n pyramid (i.e., $\hat{P}(n , 2)$) hyper-pyramid) onto the H_{2n+1} hypercube with dilation 2.

The algorithm operates as follows. It first assigns unique addresses (i , j) to the nodes of the hyper-pyramid, so that *i* is the level number and *j* is a number from 0 to $2^{id} - 1$ (the number of nodes at level *i* is equal to 2^{id}); the binary representation of *j* contains *id* bits. The parent of node $\alpha(i , j)$ is $\alpha(i+1 , j_{[d,id-1]})$, where $i > 0$, and $j_{[d,id-1]}$ represents the (d(i - 1) - 1)-bit number which is obtained by preserving the upper bits of *j* with subscripts d through id - 1. Its children are $\alpha(i+1, j \bullet X_{[0,d-1]})$, where "•" represents concatenation and X[0,d-1] represents *d* "don't care" bits.

Let ϕ_k be the mapping function for the two-rooted hyper-pyramid with k+1 levels and for the H_{kd+1} hypercube. When compared to the conventional hyper-pyramid, the two-rooted hyper-pyramid has an additional root node and additional edges that connect directly the additional root to all nodes at level 1. Let $\alpha(0 , \varepsilon)$ and $\alpha(0', \varepsilon)$ represent the two roots. The mapping function is defined in a recursive fashion. The roots for $k = 0$ are mapped onto the adjacent nodes $\phi_0(\alpha (0, \varepsilon)) = 0$ and $\phi_0(\alpha (0', \varepsilon)) = 1$.

The mapping function ϕ_{k+1} is then defined in terms of ϕ_k as follows. ϕ_k is first applied to the 2^d distinct H_{kd+1} subcubes of $H_{(K+1)d+1}$ which are distinguished by the d most significant bits of their addresses. If the superscript for node addresses distinguishes among these subcubes, then the recursive definition of the mapping function is as follows:

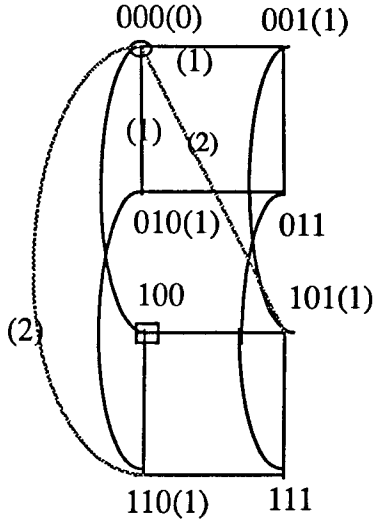
$$\phi_{k+1} (\alpha (0 , \varepsilon)) = \phi_k(\alpha (0 , \varepsilon))$$

$$\phi_{k+1}(\alpha (0' , \varepsilon)) = \phi_k(\alpha^{2^{d-1}}(0 , \varepsilon))$$

$$\phi_{k+1}(\alpha (1 , l)) = \phi_k(\alpha^l (0', \varepsilon)) \text{ if } l = 0 \text{ or } l = 2^d-1$$

$\hat{P}(0)$

$\hat{P}(1, 2)$



$\hat{P}(2, 2)$

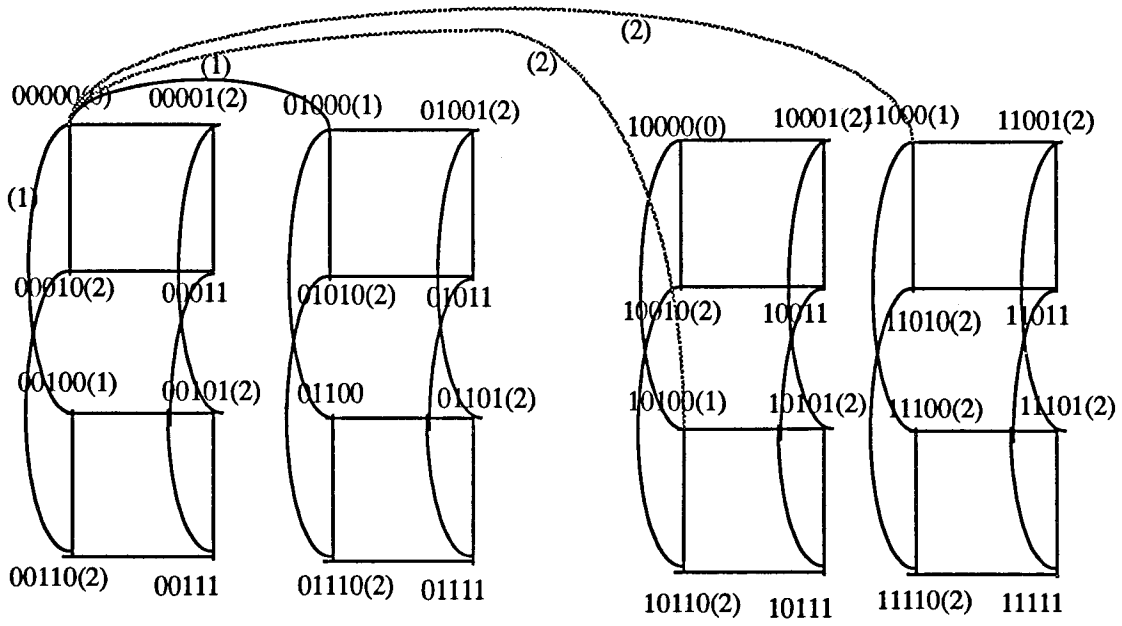


Figure 3.5 The Recursive Generation of Hypercube Node Addresses for the Mapping of the Hyper-pyramid onto the Hypercube.

otherwise,

$$\phi_{k+1}(\alpha(1,1)) = \phi_k(\alpha^1(0,\varepsilon))$$

$$\phi_{k+1}(\alpha(i,1 \bullet j)) = \phi_k(\alpha^1(i,j)), \text{ for } i > 1.$$

This mapping algorithm yields dilation d [4]. For the case of $d = 2$, which also corresponds to pyramid mapping, the dilation is 2.

Figure 3.5 shows the mapping of $\hat{P}(2,2)$ in a recursive fashion. The faint lines in the figure represent parent-child links and the weight of an edge gives its dilation. The next section discusses how the algorithm can be extended for the mapping of multilevel structures.

3.4.2 Extension for Mapping Multilevel Structures

The following rule is applied for extending the mapping for the multilevel structure with the base of size $2^n \times 2^n$ (M_n). Level j of the multilevel structure is mapped to level k of the pyramid, where k is obtained by the following equation.

$$k = \sum_{i=1}^{j} m_i$$

and 2^{m_i} is the reduction between level i-1 and i.

The resulting version of the algorithm results in maximum dilation $2m_i$ for the pair of levels i-1 and i. Therefore, the maximum dilation for this extended mapping algorithm is equal to $2\max\{m_i\}$, where all possible values of i are considered.

CHAPTER 5

COMPARATIVE ANALYSIS: CONNECTION MACHINE RESULTS

4.1 Features of the Connection Machine System CM-2 Affecting the Performance

Results for the Connection Machine system CM-2 with 16K PEs (or 8K PEs in some of the cases) are presented in this chapter. A 10-dimensional (9-dimensional) hypercube is the backbone (router) of the communications network. Each vertex of this hypercube contains a router node (communications processor) to which 16 PEs are attached.

The results on the system CM-2 are not always indicative of the performance for the pure hypercube for two reasons. Firstly, since there are 16 PEs per router node, the router nodes become bottlenecks for communication intensive operations. This may cause message routing to take many petit cycles. In the absence of this congestion one petit cycle is enough to process all of the bits of the destination address and a message, even if the message has to travel all of the dimensions in the hypercube. This makes the actual performance of the programs poorer. To alleviate this problem, results are provided for at most one active PE for each router node. Secondly, the communications hardware of the CM-2 is capable of combining multiple messages to the same destination by applying some arithmetic or logical combining (i.e., reduction) operation. The destination PE then receives the result. This reduction "on the fly" reduces the amount of traffic going to distant PEs, whenever in the algorithms many-to-one communications operations are followed by the application of combining (reduction) operations on the received data (e.g., sending data to the parent, and then finding the sum of the data received from the children at the parent level in the pyramid). In the presentation of the results below, the influence of both these issues on the performance is discussed.

For each algorithm, results were obtained for two cases. In the first case, all sixteen

PEs attached to any single router node may be used, with up to 16K PEs being initially "active". Therefore, the base for the multilevel structures assumed by Algorithms I and II has size 2^7x 2^7 while the base assumed by Algorithm III has size 2^6 x 2^6. For the purpose of getting results for the pure hypercube network, only one PE per router node may be used in the second case, with up to 1K PEs being initially "active". Therefore, the base assumed by Algorithms I and II is now 2^5x 2^5 while the base assumed by Algorithm III is 2^4x 2^4. For the purpose of comparison, results for multilevel structures with bases 2^6x 2^6 or 2^4x 2^4, and 16 PEs or 1 PE per router node, respectively, are presented for Algorithms I and II also. Average times calculated over several runs are listed. While all possible multilevel structures are considered, results are presented each time for the four most efficient structures.

4.2 First Application Algorithm: Perimeter of Objects

Each application algorithm assumes the assignment of a single pixel to each node in the base of the multilevel structure. For the sake of simplicity, this algorithm assumes the existence of a single object in the image. Assuming that the boundary pixels are known, a bottom-up process is applied to count the total number of boundary pixels. More specifically, nodes in the base that contain a boundary pixel send 1 to their parent, while nodes that do not contain a boundary pixel send 0 to their parent at level 1. To reduce the overall communication overhead, the latter addition is performed on the fly before values reach their destination. This process is repeated with higher levels until the apex is reached. The addition of values received by the apex produces the perimeter of the object.

Table I shows results for the algorithm that finds the perimeter of an object when all 16K PEs can be used. Timing results in the tables are expressed in msec. The time it takes to find the addresses of parents or children is not included in the results, these addresses are generated and stored in the local memories of the PEs during initialization. 'Reductions' represents the reductions in one dimension between pairs of neighboring lev-

els in the multilevel structures; the first reduction is for the pair of levels 0 and 1, the second reduction is for pair of levels 1 and 2 and so on. 'Base' represents the amount of time it takes to transmit data from level 0 to level 1. This process takes relatively significant time because almost all router nodes are involved in the data transfers with all the 16 PEs attached to them. Also, all data transfers in the implementation involve integer values; this was chosen for uniformity reasons because several algorithms in computer vision have a lot of similarities with the perimeter counting algorithm (e.g., they employ bottom-up processes) but they deal with integer variables. 'Top' represents the amount of time it takes the level located immediately below the highest level to send data to the topmost level and for the highest level to process the received data. 'Total' represents the amount of time taken by the algorithm. Table1 shows that although the three mapping algorithms are characterized by almost identical performance, Algorithms I and II very often achieve better performance. The reason for the relatively poor performance of algorithm III is that children have to go to higher cubes to find the parent. Also, the first two algorithms use only half of the hypercube used by the third algorithm, so communication operations are more local. The "on the fly" combining capabilities of the CM-2 are taken advantage of for multilevel structures with large reductions between the levels.

Table 2 shows results for the case where just one PE per router node may be active. Therefore, a pure hypercube network is used in this case. The relative performances are similar to the previous case. However, since there is only one PE per router node, the communication between any two PEs means communication between two distinct router nodes. With 16 PEs per router node the communication between two PEs may be within the same chip (i.e., a routing operation is not required) because of locality of communications in the immediate neighborhood in the pyramid algorithm. Therefore, the program for 16 PEs per router node may perform better than in the case of 1 PE per router node. For example, Algorithms I and II perform worse in the case of the pyramid with base 32 x 32 (Table 3) when compared with pyramid results for base 64 x 64 (Table 1).

Also, the thesis shows that the conventional pyramid performs worse than the

majority of multilevel structures. This is basically because of a smaller number of data transfers and the added advantage of "on the fly" application of reduction operations because of larger reductions between levels (i.e., more data transfers with the same destination). For this image processing algorithm, speed-ups as large as 4-fold are achieved with the incorporation of multilevel structures as compared to pyramids. Also, multilevel structures with reductions of $2^2 \times 2^2$, $2^3 \times 2^3$ or $2^4 \times 2^4$ perform better than those with extreme reductions. This is because with extremely large reductions many router nodes may have to transmit data to the same router node causing large communication overheads due to congestion at intermediate and destination nodes.

Pipelining can easily be applied to this application algorithm because all of the operations carried out at different levels, or a subset of operations for the base and the apex, are identical. This is permitted by the SIMD mode of computation provided by the CM-2. Since pipelining requires activating multiple levels simultaneously, only Algorithms II and III can be applied. In this thesis, multiple images with single object are considered for pipelining. Tables 3 and 4 show steady state results for pipelining with the same image processing algorithm.

'Throughput^{-1}' is the amount of time it takes to produce a single output in the steady state. It is the inverse of the pipelined algorithm's throughput (i.e., the number of outputs per unit time). 'Time' represents the time required to find the perimeter of the single object. The Throughput^{-1} for Algorithm II is high because it does not permit simultaneous multilevel processing including the base and so the step of transmissions from base to level 1 cannot be pipelined. When compared to the pyramid, multilevel structures can yield an upto 8.5-fold speedup for this algorithm.

4.3 Second Application Algorithm: Convolution

Two-dimensional convolution is the second image processing algorithm implemented on

the Connection Machine. The convolution algorithm convolves a $k \times k$ window of weighting coefficients with the $2^n \times 2^n$ image matrix. Let $X = \{x_{i,j}\}$ and $W=\{w_{i,j}\}$ be the image matrix and the window, respectively. The goal is to compute $Y=\{y_{r,s}\}$, where

$$y_{r,s} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} w_{i,j} \times x_{r+i,s+j}$$

with $o \leq r, s \leq 2^n - k$. This algorithm is very frequently applied in image processing.

The algorithm is described here for the pyramid topology. Its extension for multilevel structures is straightforward. Assigning a single pixel per processor in the base of the pyramid, the smallest integer Υ is found for which $2^{\Upsilon} \geq k$. Then, the base of the pyramid is partitioned into square blocks of size $2^{\Upsilon} \times 2^{\Upsilon}$. Each such partition contains the leaves of a subpyramid whose apex is at level Υ. The weighting coefficients are loaded into the upper left corner of each partition. On the Connection Machine these coefficients are loaded with k^2 broadcasting operations where appropriate sets of PEs are selected each time. This part is not included in the total execution time of the algorithm. The rest of the PEs in each partition receive zero as the weighting coefficient. The PEs then multiply the weighting coefficient with their pixel value and send the result to their parent. Parents at level 1 add the values they receive from their children and send the result to their parent. This process continues until the apex of each subpyramid is reached. Each apex at level Υ adds the values it receices from its children and sends the result, through the necessary intermediate PEs at lower levels, to the leaf PE in the upper leftmost corner of its partition. Each window in the base that contains the weighting coefficients is shifted to the right once, multiplications are performed as above, the results are shifted to left once, and the values are sent to parents at level 1. The bottom-up and the top-down processes described previously are applied with the results stored in the PE with offset (0 , 1) in each partition. To conclude, the convolution algorithm involves lateral shifts and multiplications in the base, bottom-up addition of values, and finally top-down transmission of final results. These steps are repeated $2^{2\Upsilon}$ times, which is equal to the total number of PEs in each par-

tition.

Results are presented in Table 5 for the pyramid and for the multilevel structure that yields the highest performance, assuming windows with $k = 2^s$, where s = 1, 2, 3, 4 and 5. These results indicate that Algorithm I yields the highest performance for the majority of the cases. This is because this implementation of convolution requires that only one level be active at a time, and reduced times for data transfers between parent and children result from several assignments with Algorithm I of parents and children to the same PE; the smaller dilation is also an important factor. In addition, Algorithm II has very often the largest execution time due to its larger dilation (i.e., 3) when compared to the dilation of Algorithms I and III (i.e., 2). When compared to the conventional pyramid, multilevel structures can achieve an up to 1.5-fold speedup for this algorithm.

Table 6 presents results for the convolution problem when pipelining is implemented. Only communication operations have been pipelined here due the CM-2's SIMD mode of computation. The bottom-up and top-down phases are implemented in two separate pipeline phases. The processing proceeds in two phases. In the first phase, the algorithm finds all required products between weighting coefficients and pixel values. The second phase of the algorithm implements pipelining with bottom-up and and top-down communication operations. Pipelining cannot be implemented with Algorithm II when the window size is smaller than 5 x 5 because the base can not be emulated simultaneously with other levels, leaving at most only one stage from level 1 to the apex of the subpyramid. Therefore, Table 5 does not present results for these cases. The results indicate that the implementation of pipelining with the convolution algorithm can significantly reduce the total execution time. Speedups of up to about 48% were achieved with this implementation. The implementation of pipelining reduces execution time for bottom-up and top-down processing significantly much more, but the implementation involves storing the values at the end points of the pipeline consuming considerable portions of the total execution time. When compared to the conventional pyramid, multilevel structures can achieve speedups of up to a 1.26-fold. This relatively low speedup is because the pyramid

has more levels and, therefore larger number of stages for the pipeline, which improves the performance of pipelining. Algorithms II and III yield similar performance on the average because the set of operations performed at the base are most often different from the set of operations performed at other levels.

4.4 Third Application Algorithm: Image Segmentation

Image segmentation is the third algorithm that was implemented on the Connection Machine. The segmentation algorithm partitions the image into more or less homogeneous regions with the application of an iterative technique that uses an overlapped linked pyramid structure [15].

Unlike conventional pyramids, the parent-child relationships are not fixed and may be redefined in each iteration. Each node, except for the nodes in the base, has a 4 x 4 subarray of candidate children at the immediately lower level (i.e., the standard pyramid structure is augmented by links that implement 50% ovelapping in each of the four directions). The node itself is a member of four such subarrays for the next higher level nodes. Each iteration links the node to one of four higher level candidate parents. The chosen parent is the one with the closest property value. Therefore, at the end of every iteration each parent will have from 1 to 16 children. The child-parent links define segments in the image. This iterative algorithm terminates when no further changes occur in the segments stored in the base. Tree structures that partition the image into segments are produced.

The implementation on the Connection Machine (CM-2) uses 8-bit gray level values for the pixels. The initialization of the property values at all levels is accomplished with a bottom-up processing phase. Each iteration consumes three processing phases. The first phase establishes child-parent links based on the smallest differences, as described in the previous paragraph. This phase can be carried out simultaneously for all of the levels. The second phase applies a bottom-up technique that calculates new property values for all the nodes based on the links that were established in the first phase. The third phase of

the algorithm applies a top-down technique for the assignment of the segment values to nodes in the base; following the available links, the property values of the root nodes in the existing trees serve as segment numbers and are copied into their leaves. Only the first phase of the algorithm can activate multiple levels simultaneously. However, this set of concurrent multilevel operations constitute only a small portion of all the operations carried out by this algorithm. The other two phases can activate only one level at a time. So pipelining is not appropriate for this application algorithm.

Images with a single object are considered. Tables 7 and 8 show results for this image segmentation algorithm. 'Iterations' and 'Total' represent the total number of iterations and the total execution time, respectively. For reasons similar to those for the preceding algorithms, Algorithm I yields highest performance for 16 PEs per router node. However, Algorithm III performs slightly better than Algorithm I for 1 PE per router node because of the concurrent multilevel phase and the non-locality of data transfers with at most 1 PE per router node. Algorithm II achieves performance comparable to Algorithm I for 16 PEs per router node because of the same number of required petit cycles. Although the same algorithm also performs very well for 1 PE per router node, it yields increased execution times when compared to Algorithms I and III because of the non-locality of data transfers and increased dilation. The results also show that an up to ten-fold speedup can be achieved by incorporating a multilevel structure other than the conventional pyramid.

Table 1 : Finding the Perimeter of an Object. 16 PEs per Router Node.

	Base: 128 x 128				Base : 64 x 64			
Alg.	Reductions	Base	Top	Total	Reductions	Base	Top	Total
I	128	1.49	1.49	1.52	64	1.30	1.30	1.33
	64,2	1.66	0.24	1.95	16,4	1.20	0.24	1.49
	4,32	1.74	0.33	2.13	4,4,4	0.64	0.27	1.30
	32,4	2.02	0.24	2.31	32,2	1.29	0.24	1.58
	pyramid	1.75	0.46	5.12	pyramid	0.64	0.45	2.85
II	128	1.30	1.30	1.33	64	1.20	1.20	1.23
	64,2	1.44	0.24	1.73	16,4	1.00	0.24	1.30
	8,16	1.53	0.24	1.81	4,4,4	0.54	0.27	1.30
	32,4	1.62	0.24	1.90	32,2	1.10	0.25	1.41
	pyramid	1.47	0.27	4.65	pyramid	0.54	0.45	2.73
III					64	1.11	1.11	1.11
					16,4	0.92	0.53	1.49
					4,4,4	0.92	0.53	1.58
					32,2	1.19	0.42	1.67
					pyramid	1.01	0.45	3.55

Table 2 : Finding the Perimeter of an Object. 1 PE per Router Node.

	Base: 32 x 32				Base: 16 x 16			
Alg.	Reductions	Base	Top	Total	Reductions	Base	Top	Total
I	32	0.93	0.93	0.95	16	0.83	0.83	0.85
	16,2	0.93	0.25	1.23	8,2	0.73	0.26	1.04
	8,4	0.93	0.27	1.24	4,4	0.73	0.35	1.13
	4,8	1.02	0.34	1.42	2,8	0.73	0.63	1.42
	pyramid	1.21	0.45	3.58	pyramid	0.74	0.46	2.24
II	32	0.93	0.93	0.95	16	0.73	0.73	0.75
	16,2	0.92	0.25	1.22	8,2	0.73	0.26	1.04
	8,4	0.92	0.25	1.22	4,4	0.73	0.44	1.22
	4,8	0.92	0.53	1.50	2,8	0.73	0.73	1.50
	pyramid	0.92	0.44	3.38	pyramid	0.73	0.44	2.39
III					16	0.83	0.83	0.85
					8,2	0.64	0.44	1.13
					4,4	1.11	0.52	1.67
					2,8	1.01	0.63	1.69
					pyramid	1.02	0.45	2.66

Table 3: Finding the Perimeter of Objects. Pipelining. 16 PEs per Router Node.

	Base: 128 x 128			Base: 64 x 64		
Alg.	Reductions	Throughput^{-1}	Time	Reductions	Throughput^{-1}	Time
II	4,32	2.99	3.47	32,2	2.87	3.32
	8,16	3.65	4.02	2,32	2.13	2.54
	16,8	4.93	5.31	16,2,2	3.97	4.61
	64,2	4.95	5.32	16,4	2.68	3.08
	pyramid	12.80	29.03	pyramid	5.22	12.82
III				32,2	2.49	4.98
				2,32	2.62	5.24
				16,2,2	2.75	8.26
				16,4	2.77	5.54
				pyramid	3.34	20.08

Table 4: Finding the Perimeter of Objects. Pipelining. 1 PE per Router Node.

	Base: 128 x 128			Base: 64 x 64		
Alg.	Reductions	Throughput^{-1}	Time	Reductions	Throughput^{-1}	Time
II	8,4	2.24	2.66	4,2,2	2.79	3.66
	4,8	2.85	2.85	2,2,4	2.93	3.94
	16,2	2.43	2.85	2,4,2	2.93	3.94
	2,16	3.27	3.78	pyramid	3.68	5.92
	pyramid	8.23	13.74			
III				8,2	1.39	2.79
				2,8	1.68	3.35
				2,2,4	1.84	5.53
				2,4,2	1.84	5.53
				pyramid	1.93	7.74

Table 5: Convolution

		16 PEs/router node			1 PE/router node		
Alg.	Window	Reductions	Base	Total	Reductions	Base	Total
I	2 x 2	pyramid	2.11	22.52	pyramid	6.10	28.20
	4 x 4	4	11.39	92.52	4	25.86	113.79
		pyramid	11.43	107.20	pyramid	25.92	136.45
	8 x 8	4,2	63.83	463.95	8	124.05	486.06
		pyramid	63.83	522.63	pyramid	124.05	651.11
	16 x 16	2,2,4	378.25	2234.18			
		pyramid	378.34	2452.34			
	32 x 32	4,2,4	1947.29	9113.64			
		pyramid	1948.87	14682.16			
II	2 x 2	pyramid	1.96	34.17	pyramid	6.17	28.30
	4 x 4	4	11.39	135.79	4	26.13	114.31
		pyramid	11.39	166.03	pyramid	26.05	241.07
	8 x 8	8	63.59	573.57	8	124.26	486.19
		pyramid	63.59	715.44	pyramid	124.02	650.66
	16 x 16	4,2,2	377.71	2862.62	4,2,2	596.22	2640.77
		pyramid	377.63	3285.52	pyramid	596.72	3003.72
	32 x 32	4,4,1	1962.25	9818.82	4,4,2	2582.63	10862.83
		pyramid	1951.83	14782.40	pyramid	2583.97	11840.71
III	2 x 2	pyramid	2.17	26.88	pyramid	2.35	28.51
	4 x 4	4	11.86	125.74	4	12.92	132.56
		pyramid	11.86	136.27	pyramid	12.96	158.78
	8 x 8	8	66.06	453.70	8	72.09	479.67
		pyramid	66.16	655.26	pyramid	72.31	752.60
	16 x 16	2,4,2	391.11	2623.78	8	364.76	2957.89
		pyramid	391.11	3230.60	pyramid	364.86	3377.83
	32 x 32	2,4,4	2009.76	11292.03			
		pyramid	2016.86	13878.42			

Table 6: Convolution. Pipelining.

		16 PEs/router node		1 PE/router	
Alg.	Window	Reductions	Total	Reductions	Total
II	8 x 8	pyramid	463.93	pyramid	488.80
	16 x 16	4,2,2	1714.90	4,2,2	1885.07
		pyramid	1946.37	pyramid	2050.13
	32 x 32	4,2,4	6478.40	4,2,4	7740.57
		pyramid	8218.48	pyramid	8388.96
III		pyramid	117.12	pyramid	111.38
	8 x 8	2,4	460.37	2,4	432.31
		pyramid	460.73	pyramid	433.22
	16 x 16	2,4,2	1976.33	2,4,2	1775.08
		pyramid	1941.60	pyramid	1786.18
	32 x 32	2,2,2,4	8236.37		
		pyramid	8237.66	pyramid	

Table 7: Image Segmentation. 16 PEs per Router Node.

	Base: 128 x 128			Base: 64 x 64		
Alg.	Reductions	Iterations	Total	Reductions	Iterations	Total
I	2,32,2	6	306.16	32,2	5	195.88
	64,2	7	320.01	2,16,2	5	224.87
	2,2,4,2,2,2	5	385.72	16,2,2	5	228.83
	2,2,2,8,2	6	400.75	2,2,8,2	5	260.27
	pyramid	7	599.15	pyramid	7	501.16
II	2,32,2	6	310.31	32,2	5	197.08
	64,2	7	322.22	2,16,2	5	226.59
	2,2,4,2,2,2	5	413.29	16,2,2	5	231.67
	2,2,2,8,2	6	422.76	2,2,8,2	5	267.33
	pyramid	7	666.91	pyramid	7	519.09
III				32,2	5	201.48
				2,16,2	5	232.34
				16,2,2	5	237.71
				2,2,8,2	5	266.78
				pyramid	7	512.91

Table 8: Image Segmentation. 1 PE per Router Node.

	Base : 32 x 32			*Base : 16 x 16*		
Alg.	Reductions	Iterations	Total	Reductions	Iterations	Total
I	16,2	2	47.14	8,2	2	40.43
	8,2,2	3	115.76	4,2,2	5	170.82
	4,4,2	3	116.93	2,4,2	8	268.68
	2,8,2	5	186.93	pyramid	3	141.38
	pyramid	7	468.42			
II	16,2	2	48.94	8,2	2	42.15
	8,2,2	3	121.38	4,2,2	5	178.05
	4,4,2	3	121.50	2,4,2	8	288.89
	2,8,2	5	195.29	pyramid	3	151.64
	pyramid	7	481.32			
III				8,2	2	36.19
				4,2,2	5	168.18
				2,4,2	8	266.85
				pyramid	3	140.10

CHAPTER 5

ALTERNATE MULTILEVEL STUCTURES: STOCHASTIC PYRAMIDS

The dependence of low resolution representations on the position of the sampling grid and the input image is important in image pyramid applications. The rigidity of the pyramid structure may give rise to artifacts when pyramids are used for tasks such as analysis of line-drawings, object-background discrimination, or compact object extraction. Stochastic pyramids adapt their structure to the content of the input image to compensate for these artifacts [5]. Stochastic pyramids are built using irregular tessellations that generate an adaptive multiresolution representation of the input image. The hierarchy of representations is built bottom-up and is molded to the structure of the input image. Stochastic pyramids preserve the most of the properties of "classical" image pyramids. However in such a pyramid the metrical relations among cells are no longer carried implicitly by the sampling structure. A cell at level $l + 1$ cannot know a priori where its neighbors on level $l + 1$ or its children on level l are located relative to the original sampling grid [5].

5.1 Graph Representation of Irregular Sampling Hierarchies

The cells on level l of the pyramid are taken as the vertices of an undirected graph $G[l]$. The edges of the graph describe the adjacency relations between cells on level l. $G[l]$ = (V[l],E[l]), where V[l] is the set of vertices and E[l] is the set of edges. The pyramid is constructed by a sampling or decimation process. Each level is constructed from the level below it by selecting a subset of the vertices. Thus, a vertex at any level can be regarded as a vertex of G[0], the 8-connected sampling grid of the original image. When level l is decimated to construct level $l+1$, each nonsurviving vertex is associated with one of the surviving vertices. So a vertex on any level is associated with a set of vertices, called its

"region" in the original image. These regions define a tessellation of the image. The cell $c[l+1] \in V[l+1]$ must represent a connected subset of cells $\{c_0[l],c_1[l],...c_u[l]\}$ at level l. Let $c_0[l] = c[l+1]$. When the graph is contracted to generate a new higher level, two constraints must be satisfied:

$$\forall i = 1, 2,...,u \qquad (c_i[l], c_0[l]) \in E[l]$$

$$\forall\, c_0[l] \in V[l], d_0[l] \in V[l] \qquad (c_0[l], d_0[l]) \notin E[l]$$

where c_0 and d_0 are survivors on level l.

After defining the survivors on level $l+1$ ($V[l+1]$), the edges at the next level ($E[l+1]$) must be defined . The constraint for an edge between vertices $c[l+1] \equiv c_0[l]$ and $d[l+1] \equiv d_0[l]$ in $G[l+1]$ is

$$\exists\, 0 \le i \le u, 0 \le j \le v \qquad (c_i[l],d_j[l]) \in E[l].$$

5.2 Creation of Stochastic Pyramids

The decimation process is dependent on the image data. We assume that every cell c_i (a vertex of $G[l]$) carries a value g_i characterizing its region of the image. Let c_0 on level l have r neighbors on level l, i.e., let its degree as a vertex of $G[l]$ be r. Every neighbor c_i, i = 0, ...,r of c_0 is examined, and it is decided whether it belongs to the same class as c_0 or not. This decision can depend in any desired way on the values g_i, for i=0,...,r. A binary number λ_i, where i=0,...,r, is associated with each neighbor, where $\lambda_i = 1$ if c_i belongs to the same class as c_0, and $\lambda_i = 0$ otherwise.

The decimation algorithm employs three variables for every cell: two binary state variables p and q, and a random variable uniformly distributed between [0,1] with outcomes x. The survivors are chosen by an iterative local process. Let k = 0,1,...be iteration index. Initially all $p_i(0) = 0$. A cell survives if at the end of the algorithm its $p_0(k)$ variable has the value 1.

Every iteration has two steps. First $q_0(k)$ is updated based on the states $p_i(k-1)$ of neighboring cells in the same class. $q_0(k)$ basically denotes whether a candidate is eligible

to be a survivor in the current iteration.

$q_0(k) = 1$ if $\lambda_i p_i(k-1) = 0 \quad \forall\, i = 0,...,r$

$q_0(k) = 0$ otherwise.

Then $p_0(k)$ is computed on the updated values of $q_i(k)$:

$p_0(k) = 1$ if $q_0(k)x_0(k) \max_{i=0,...r} (\lambda_i\, q_i(k)x_i(k)) > 0$

$p_0(k) = p_0(k-1)$ otherwise.

To become a survivor the outcome of the random variable x drawn by the cell must be local maximum among the outcomes drawn by the neighbors in the same class.

The iterative process is repeated until every cell has at least one survivor in its neighborhood.

5.3 Connected Component Analysis: Connection Machine Results

In a labeled image the pixels are classified into a small number of classes distinguished by different labels. A connected component is a maximal set of connected pixels sharing the same label. In this thesis the analysis is restricted to binary images i.e., images have only two labels. In the connected component analysis, every connected component is reduced to a separate root and adjacency relations among the components are extracted.

5.3.1 Algorithm

The technique used in this algorithm obtains the description of connected components in a binary image in O(log(class_size)) steps. The neighbors are classified according to whether they belong to the same class. Let the binary label of c_i be g_i. For r neighbors of a cell c_0, the class membership variables are defined as:

$\lambda_i = 1$ if $g_i = g_0$

$\lambda_i = 0$ if $g_i \neq g_0$.

Since the above definition of λ_i is symmetrical, it can be regarded as the weight of the edge (c_0 , c_i). The case $\lambda = 0$ is equivalent to removing the edge from E[*l*]. If E'[*l*] is the set of edges having $\lambda = 1$, and G'[*l*] = (V[*l*],E'{*l*}) then the connected components in the labeled image are represented by connected components in the graph G'[*l*], for all $l \geq 0$.

The subgraphs of *G'* are decimated independently by following the process described in the previous section. Each subgraph is recursively contracted into one vertex, the *root* of the connected component. Figure 5.1 shows an example of appying the algorithm for the creation of the different levels during the connected components extraction.

5.3.2 Connection Machine Results

The algorithm was implemented for six different images shown in Figure 5.2. Table 9 presents the results obtained from actual run on the connection machine. 'Total' represents the total time taken by the algorithm to be executed on CM-2. The time is in seconds. 'Levels' represents the highest level number up to which the hierarchy has to be built to extract the connected components. At most 16 PEs per router node were used for this application. The time required to build stochastic pyramids is much greater than that for conventional pyramids. This is because the structure is quite irregular; the parent-children links can not be derived at static time as with conventional pyramids, but they have to be defined at run time. A large number of addresses have to be communicated between adjacent levels while determining links at the next level. Also the communication of values have to be done by "get" operations rather than "send" operations to avoid possible collision not permitted in CM-2 for the "send" operation. The "get" operation is implemented as two "sends", and so is more expensive in CM-2. Also, the dilation between neighbors generally increases as one goes up the hierarchy. It can be seen in the Table 9 that the time taken by the algorithm increases almost proportionaly to the number of levels. If the program is run more than once for the same image, it may produce a different number of levels and so different timing because of the stochastic process involved in the algorithm.

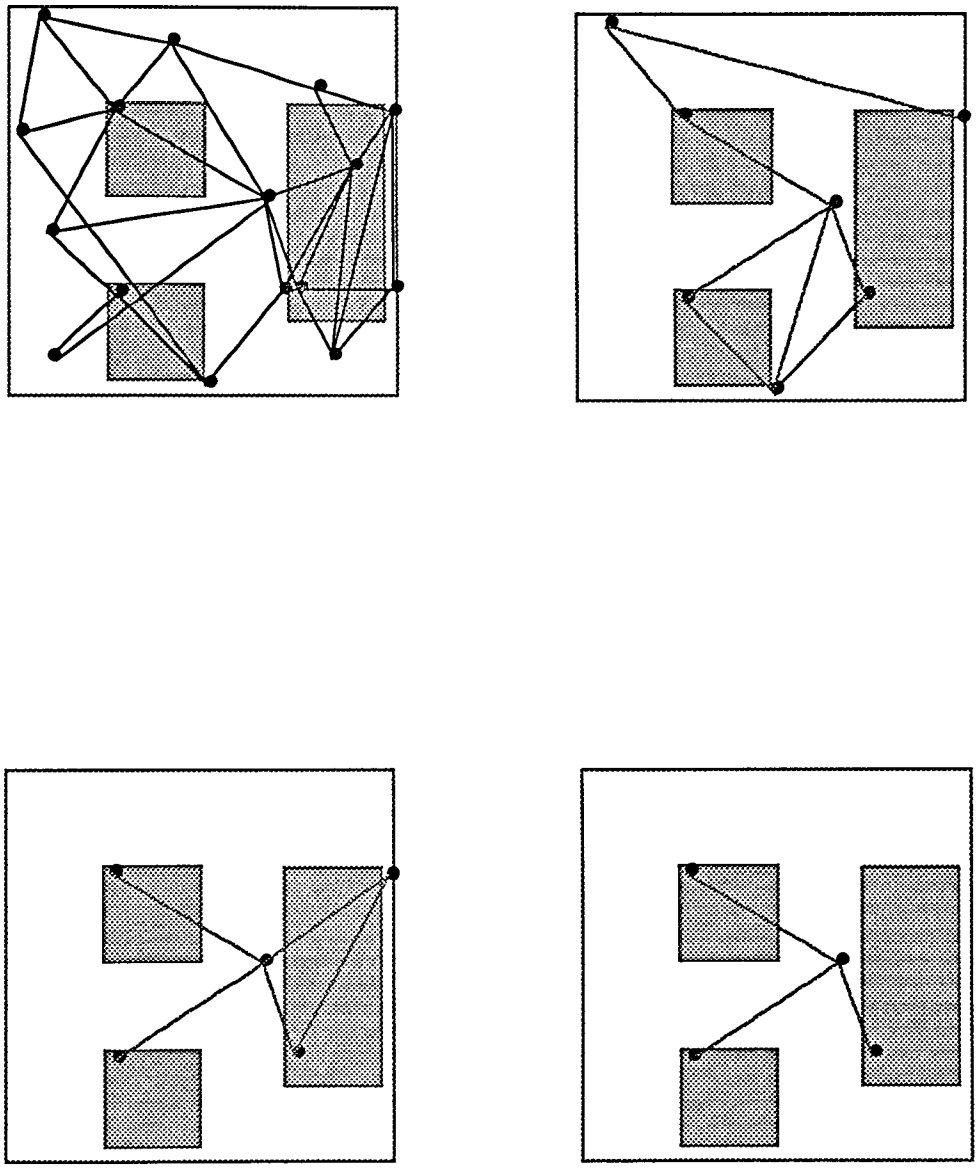
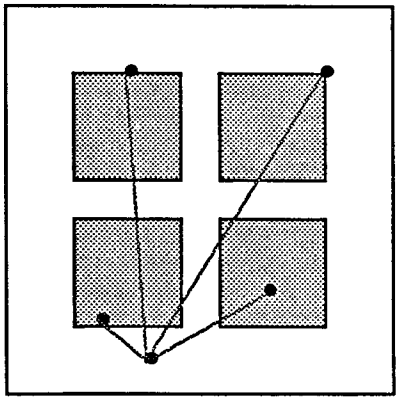
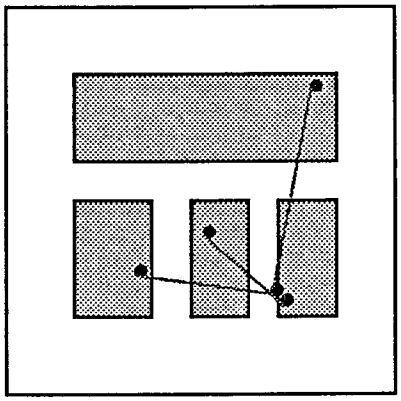


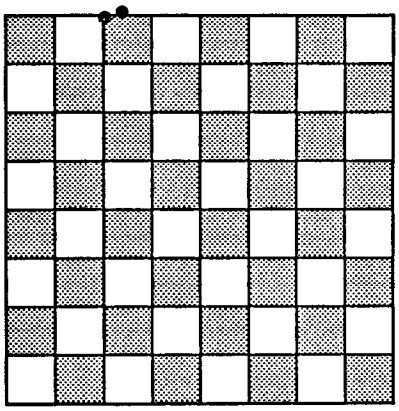
Figure 5.1 : Connected Components Extraction (Levels 2-5)



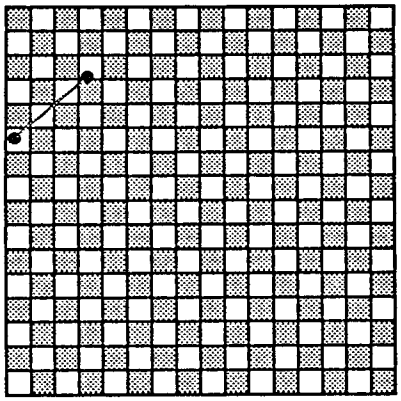
(a) Image I



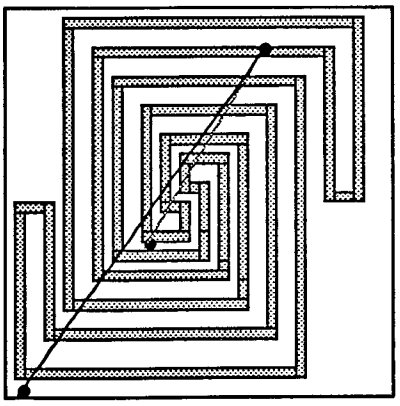
(b) Image II



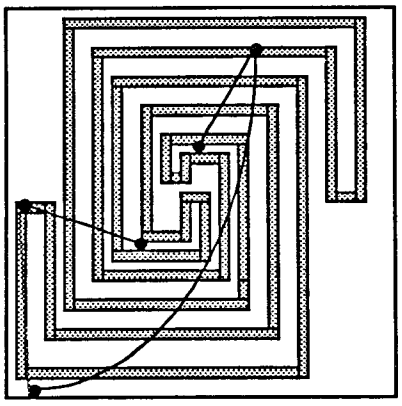
(c) Image III



(d) Image IV



(e) Image V



(f) Image VI

Figure 5.2 : Images with the Adjacency Graph of the Root Level Superimposed

The irregular connection graphs at different levels limit the SIMD capabilities of CM-2. The algorithm should produce much better results on an MIMD system. The decimation process and the process of establishing links at the next level utilize the Concurrent-Read capability of CM-2 considerably.

Table 9 : Results for Connected Components Analysis of Binary Labeled Images

Image	Levels	Total
I	9	36.74
II	9	34.53
III	10	41.76
IV	11	37.96
V	9	31.33
VI	8	32.56

CHAPTER 6

CONCLUSIONS

The problem of mapping multilevel structures onto the hypercube has been addressed in this thesis. The conventional pyramid belongs to the class of multilevel structures. Three algorithms that map pyramids onto the hypercube were extended for multilevel structures. A comparative analysis of the three algorithms has been carried out. The comparative analysis incorporates both analytical techniques and actual runs on a Connection Machine CM-2 system composed of 16 K processors. Algorithms I and II have low cost; the number of PEs in the target hypercube is the same as the number of PEs in the base of the multilevel structure. In contrast, Algorithm III requires double the number of PEs required by the first two algorithms. The major drawback of Algorithm I is that it can emulate only one level of the multilevel structure at a time and so it does not permit concurrent multilevel processing. Algorithm II can emulate multiple levels, excluding base, simultaneously. Finally, Algorithm III can emulate all levels simultaneously.

Connection Machine results for three computer vision algorithms were included for the comparison of the mapping algorithms. The chosen computer vision algorithms are good representatives of classes of algorithms that apply different types of multilevel processing. For the perimeter counting application, the three algorithms produce almost similar performance. Algorithm I performs the best for the convolution algorithm; Algorithm II does not perform as well because of larger dilation for parent-child links used frequently by the application algorithm. For the segmentation application, Algorithm I performs the best for all of the multilevel configurations. Algorithm III performs equally well for pyramids but performs worse for large reductions between levels. Algorithm II produces performance comparable to Algorithm I. Thus, when the application algorithm activates one level at a time, Algorithm I is the best choice as it yields the best performance at lower cost.

The performance of Algorithm II and Algorithm III for pipeline processing has also been compared. Pipelining can not be applied to Algorithm I. Pipelined processing provides much better results for both of the algorithms. For pipelined perimeter counting, the throughput of Algorithm III is higher than that of Algorithm II as it permits simultaneous multilevel processing that includes the base, and also has smaller dilation. For convolution with pipelining, the performance of Algorithm III is slightly better than that of Algorithm II. The cost of Algorithm II is half of that required by Algorithm III. So, Algorithm II is capable of yielding good performance at low cost for algorithms that apply concurrent multilevel processing.

Also, it was shown that multilevel structures provide much better performance in most of the cases than conventional pyramids. Therefore, the choice of appropriate multilevel structures for the implementation of algorithms on hypercube supercomputers becomes imperative.

Stochastic pyramids, alternate multilevel structures studied in this thesis, mold their structure to the input image while they are being built bottom-up, and so provide an interesting solution to artifacts found in conventional pyramids due to their rigid structure. The creation of stochastic pyramids on the hypercube requires much more time than that for the conventional pyramid. The hierarchy is constructed in O(log(image-size)) steps. The irregular structure is to some extent in contrast to the SIMD structure of the CM-2.

REFERENCES

1 Ziavras, S. G., "Techniques for Mapping Deterministic Algorithms onto Multilevel Systems" *International Conference on Parallel Processing*. (1990): 226-233.

2 Stout,. Q. F., "Hypercubes and Pyramids" *in Pyramidal Systems for Computer Vision Canoni and S. Levialdi (eds.), Spring-Verlag, Berlin, Heidelberg, Germany* (1986): 74-89

3 Patel, S.C., and S. G. Ziavras, "Comparative Analysis of Techniques that Map Hierarchical Structures into Hypercubes." *in Proc. Parallel Distributed Computing Systems Conf.*(1991): 295-299.

4 Ho, C. T., and S. L. Johnsson, "Dilation d Embedding of a Hyper-pyramid into a Hypercube." *in Proc Supercomputing*. (1989): 294-303 .

5 Montanvert, A., P. Meer , and A. Rosenfeld, "Hierarchical Image Analysis Using IrregularTessellations." *IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. 13, No. 4*. (1991): 307-316.

6 Ziavras, S. G., "Efficient Mapping Techniques for a Class of Hierarchical Systems." *IEEE Trans. Parallel Distributed Systems*, Accepted for Publication.

7 Cantoni V., and S. Levialdi, "Multiprocessor Computing for Images." *proc. IEEE Special Issue Computer Vision, Vol. 76, No. 8* (1988): 959-969.

8 Chan, T. F., and Y. Saad, "Multigrid Algorithms on Hypercube Multiprocessors." *IEEE Trans. Comput., Vol C-35, No. 11* (1988): 969-977.

9 Hwang, K., and F.A. Briggs, "Computer Architechture and Parallel Processing." *McGraw Hill publication.* (1984).

10 *Connection Machine Model CM-2 Technical Summary,* Version 6.0, Thinking Machines Corporation, Cambridge, MA (1990).

11 Lai, T.H., and W. White, "Mapping Pyramid Algorithms into Hypercubes." *Journal Parallel Distributed Computing, Vol. 9* (1990): 42-54.

12 Ziavras, S.G., and D. Shah, "High Performance Emulation of Hierarchical Structures

on Hypercube Supercomputers." *Concurrency:Practice and Experience*, Accepted for publication.

13 Ziavras, S.G., and M.A. Siddiqui, "Pyramid Mappings onto Hypercubes for Computer Vision: Connection Machine Comparative Study." *Concurrency: Practice and Experience*, Accepted for publication.

14 Ziavras, S. G., "On the Problem of Expanding Hypercube-Based Systems", *Journal of Parallel and Distributed Computing, Vol 16* (1992): 41-53.

15 Burt, P. J., T.-H. Hong, and A. Rosenfeld, "Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation." *IEEE Transactions on Systems, Man, and Cybernetics, Vol 11* (1981): 802-809.