1991

New Techniques in Scene Understanding and Parallel Image Processing.

Krishnakumar Narayanan
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313 761-4700 800 521-0600

Order Number 9219562

New techniques in scene understanding and parallel image processing

Narayanan, Krishnakumar, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

New Techniques in Scene Understanding and Parallel Image Processing

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by

Krishnakumar Narayanan

B.Sc., Thiagarajar College of Engineering, Madurai University, India, 1981
B.S., Indian Institute of Science, India, 1984
M.S., Louisiana State University, Louisiana, 1991

December, 1991

Acknowledgements

First and foremost, I thank my major advisor, Prof. Sitharama S. Iyengar, for his support, encouragement and his thoughtful criticisms and suggestions during the entire course of the research. I will always be grateful for this.

I also extend my thanks to the members of the graduate advisory committee: Prof. Charles Harlow, Prof. Donald Kraft, Dr. Doris Carver, and Dr. Andrew Hoppe.

I would like to thank Dr. Ronald Holyer and Matthew Lybanon, at the Dept. of NAVY, for providing the financial support for this research and their useful technical suggestions. I extend my appreciation to Sarah Peckinpaugh for testing the software.

I am very grateful to Dr. Govindarajan Ramaswamy whose encouragement and support helped me start graduate studies at LSU. I like to express my sincere gratitude to Dr. Sridhar Radhakrishnan and Dr. Mohan Sharma whose initial suggestions and helpful discussions aided in choosing the topics of study.

I am thankful to Dr. Rajnarayanan Subbiah who has accompanied me throughout my graduate study, especially, during those tedious exam preparations. I am grateful to my professional partner, Vinayak Hegde (soon to be Dr!) for collaborating with me in solving many of the problems addressed in the dissertation.

I like to thank my friends, Sankar, Sanjoy, Mahesh, Pramod, Sameer, Phatak, Venky, Sivakumar, Vijayan, Daryl, and Amit who have made my stay at LSU a pleasant and a remembering one.

My love goes to my parents who have always showed a remarkable enthusiasm for learning in their children. I am grateful to my aunt, Hemalatha for providing all the help and support I needed during my studies in India. My love also goes to my brothers and my sister.

And last but not least, my WIFE, Sunitha gets my love and appreciation for her understanding, patience and unwavering support during the final phase of my studies at LSU.

Finally, I dedicate this dissertation research to the memory of my grandfather, Mr. K.P. Ramakrishnan who has been and will always be my source of inspiration.

Table of Contents

List of Figures

Abstract

There has been tremendous research interest in the areas of computer and robotic vision. Scene understanding and parallel image processing are important paradigms in computer vision. New techniques are presented to solve some of the problems in these paradigms.

Automatic interpretation of features in a natural scene is the focus of the first part of the dissertation. The proposed interpretation technique consists of a context dependent feature labeling algorithm using non linear probabilistic relaxation, and an expert system. Traditionally, the output of the labeling is analyzed, and then recognized by a high level interpreter. In this new approach, the knowledge about the scene is utilized to resolve the inconsistencies introduced by the labeling algorithm. A feature labeling system based on this hybrid technique is designed and developed. The labeling system plays a vital role in the development of an automatic image interpretation system for oceanographic satellite images. An extensive study on the existing interpretation techniques has been made in the related areas such as remote sensing, medical diagnosis, astronomy, and oceanography and has been shown that our hybrid approach is unique and powerful.

The second part of the dissertation presents the results in the area of *parallel image processing*. A new approach for parallelizing vision tasks in the low and intermediate levels is introduced. The technique utilizes schemes to embed the inherent data or computational structure, used to solve the problem, into parallel architectures such as

hypercubes. The important characteristic of the technique is that the adjacent pixels in the image are mapped to nodes that are at a constant distance in the hypercube. Using the technique, parallel algorithms for neighbor-finding and digital distances are developed. A parallel hypercube sorting algorithm is obtained as an illustration of the technique. The research in developing these embedding algorithms has paved the way for efficient reconfiguration algorithms for hypercube architectures.

CHAPTER ONE

INTRODUCTION

1.1 Overview

There has been growing interest in recent years in the possible upgrading of machine *perception* using *visual* sensors. Unlike other sensors, sensing and processing the visual data is difficult and challenging. Incorporating visual sensory data to understand the environment is the fundamental issue in many of the *vision-based mobile robots*. Thus *computer vision* (also called machine vision) has become an integral part of the present-day robotic systems. Computer vision and many of its potential applications have become major topics of research within the *artificial intelligence* (AI) community. Consequently, the visual processing is decomposed into two parts:

- passive, domain-independent, data-driven image analysis (*low level vision*)
- domain-dependent goal-driven scene analysis (*high level vision*)

The goal of modern computer vision research is the identification and use of natural constraints or assumptions about the world to derive unambiguous output. The current research is also centered on implementing these visual processes on *parallel* computational architectures, since only in this way is the sophisticated performance of human vision remotely attainable.

Computer vision may be defined as: "*the process of extracting, characterizing, and interpreting information from images of two- or three-dimensional world.*"

An 'image' in our context is a two dimensional array of data resulting from sampling the projected instantiation of a local variable, the scene *brightness* function, obtained via a sensing device. The function values are either brightness values or vectors of brightness values sensed in different spectral bands, e.g. color images. In the black-and-white case these values are called *grey* values.

The process of computer vision may be subdivided into six principal areas: sensing, preprocessing, segmentation, description, recognition, and interpretation [Fu87a].

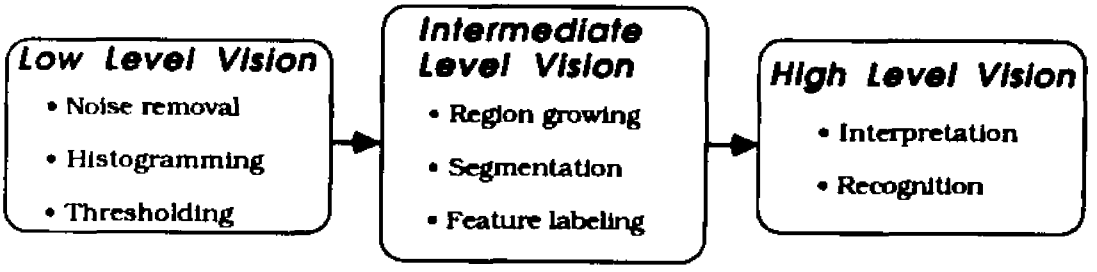


Figure 1.1 Three Levels of Processing in Computer Vision

- *Sensing* is the process that yields a visual image.
- *Preprocessing* deals with techniques such as noise removal and enhancement.
- *Segmentation* is the process that partitions an image into objects of interest.
- *Description* deals with the computation of features suitable for differentiating one type of object from another.
- *Recognition* is the process that identifies these objects.
- *interpretation* assigns meaning to an ensemble of recognized objects.

We may group these six processes into three levels according to the sophistication involved in their implementation.

• Low level vision

In this level we may place those processes requiring no knowledge on the scene domain. Accordingly sensing and preprocessing may be part of the low level vision.

• Intermediate level vision

We associate with intermediate level vision those processes that extract, characterize and label components in an image resulting from low level vision. We may treat segmentation, description, and recognition of individual objects as intermediate-level vision functions.

• High level vision

High-level vision involves processes that attempt to emulate cognition.

Figure 1.1 depicts these three levels and their associated processes in a typical computer vision system. The area of *digital image processing* includes the processes in both the low and intermediate levels.

There are basically two paradigms in modern computer vision systems:

- *AI based approach for problem solving*
- *Parallel and Distributed Processing* (PDP)

Informally, these two paradigms address the two basic questions, namely, *how to solve the problem ?* and *how to solve the problem faster ?* The results obtained in our research also may be classified under these two paradigms. More specifically, the first part of this dissertation presents our results in the area of *scene understanding* (first paradigm) and the second part in *parallel image processing* (second paradigm). The purpose of this chapter is to give necessary background and our contributions in these

two well known paradigms. The reminder of this section is devoted to give an overview on these paradigms.

The process of *interpretation* is the most challenging and interesting one among those six processes mentioned before. An interpretation is a high-level description of the scene from which the image was taken. For high level interpretation, the principal unit of information is a symbolic description of an object, or a set of image events, sometimes referred to as image *features* or symbolic tokens, extracted from the image. The description includes relationships both to other two dimensional symbol tokens extracted from the sensory data, such as lines, regions, and surfaces and to other objects in the three dimensional scene being viewed.

Automatic scene interpretation requires the construction of at least a partial description of the original environment, rather than a description of the image itself. Interpretation is made even more difficult by the fact that the image data is inherently ambiguous and incomplete. It is a well established fact that conventional domain-independent approaches are not suitable for interpretation. This calls for making use of the *knowledge* embedded in the scene for an unambiguous and consistent interpretation. One of our primary contributions is a *knowledge-based* interpretation technique and is the focus of the first part of the dissertation.

It is well known that the processes involved in machine vision are computationally intensive and there has always been a demand for faster computers. To date two popular methods have been used to achieve higher performance [Quin88a]. They are:

- increased speed of the electronic circuitry

- more number of operations that can take place concurrently

Though the first method produced very fast processors, more emphasis is given to incorporate the second. The concurrent execution of operations is normally achieved either through *pipelining* or *parallelism*. Pipelining increases concurrency by dividing a computation into a number of steps, while parallelism is the use of multiple resources to increase concurrency. A parallel computer is a computer designed for the purpose of parallel processing. The design of parallel algorithms becomes an important issue as numerous architectures are developed. Since the design and performance of a parallel algorithm depends very much on the architecture of the parallel machine, it is necessary to keep the architecture in mind when designing parallel algorithms. One approach to constructing parallel algorithms is to recognize parallelism in the existing sequential algorithms. The second part of the dissertation focuses on developing techniques for exploiting parallelism in image processing that would result in parallel algorithms for a class of problems in image processing.

1.2 Natural Scene Understanding Paradigm

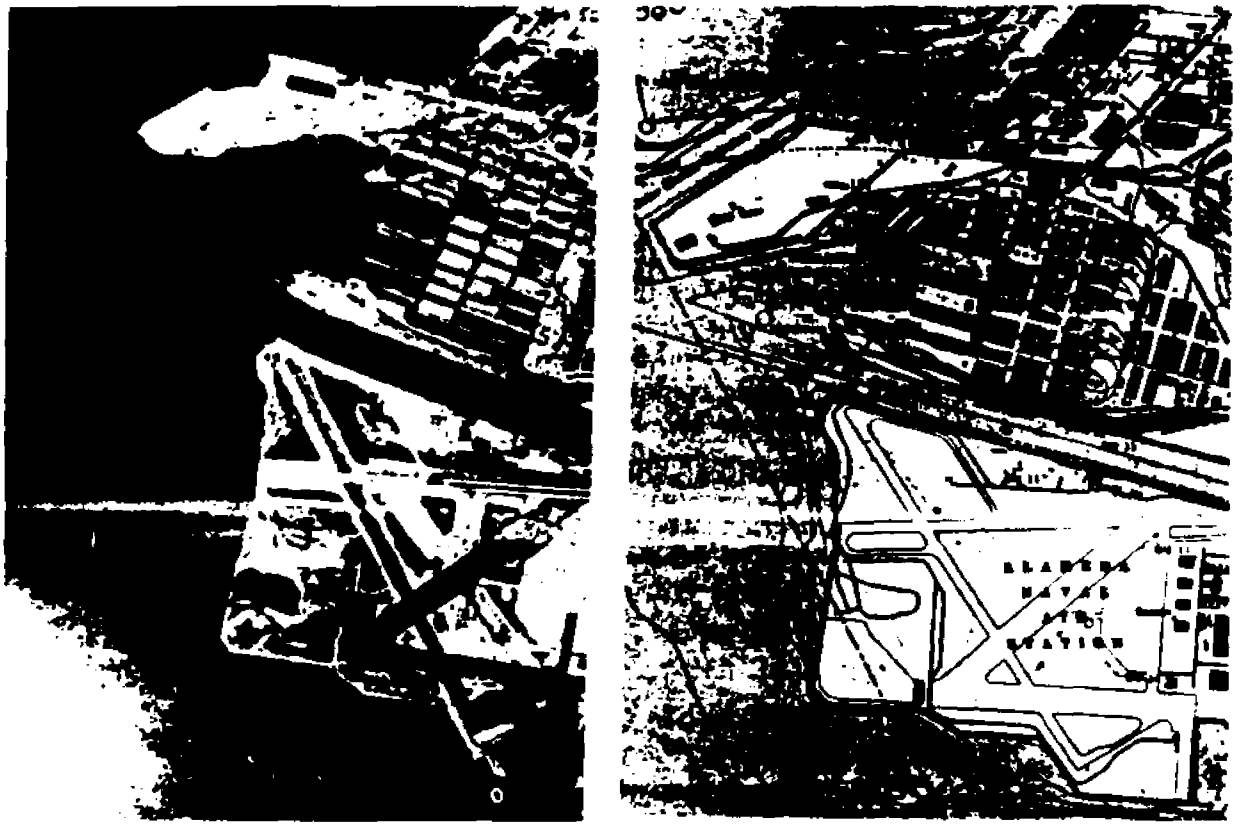
The purpose of this section is to address the fundamental issues and possible approaches for the first paradigm, namely, *AI-based approaches for problem solving*.

When we look at images or photographs of a natural scene, we see various objects of different sizes and shapes. Some of these objects may be readily identifiable while others may not, depending on our own individual perceptions and experience. When we can identify what we see on the pictures and communicate this information to others, we are practicing *scene interpretation*. A scene interpreter systematically

examines the image (or the photograph) along with other supporting materials such as maps and reports of field observations. Based on this study, an interpretation is made as to the physical nature of objects appearing in the scenes. We are interested in solving the scene interpretation problem with the help of computers.

The difficulty from the interpretive point of view is due largely to the lack of precise mathematical (i.e., typically geometrical) descriptions of the structures of interest, frequently coupled with their time-varying nature, and to the presence of high noise levels. The interpreter must detect a shape or pattern which is difficult to prescribe accurately in advance, which may be partially obscured or otherwise contaminated with noise, and which may have moved or changed shape - or both - since a previous observation (e.g., a beating heart in medical imagery, a weather system in satellite imagery).

Figure 1.2(a) shows an aerial photograph of a shipyard and 2.0(b) the corresponding map. Fortunately, this image is clear and the objects are not occluded. This may not be the case at all times, especially when we try to process satellite images, the atmospheric effects (viz., cloud cover) may have significantly contaminated the image. In such instances the interpretation becomes much more difficult. A systematic study of a natural scene such as the one shown in figure 1.2, usually involves several basic characteristics of features shown in the image of the scene. The exact characteristics useful for any specific task, and the manner in which they are considered, depend on the field of application. However, most applications consider the following basic characteristics, or variations of them: shape, size, position, texture, and time. We will describe these characteristics with this example image.



(a) (b)

Figure 1.2 Aerial Photograph of a Shipyard and its Map

- *Shape* is a very important criterion for image interpretation. It refers to the general form, configuration, or outline of individual objects. For example, the buildings seen in the figure 1.2(a) may be considered as regular polygons such as squares, rectangles, etc. In automatic image interpretation, shapes of the objects are very helpful to delineate them, and hence characterizing the shapes is an important task.

- *Size* of objects must be considered in the context of the scale of the imaged scene. For example, the number of pixels inside the object, say, the parking lot may represent the size.

- *Position* of objects in relation to other features can be very helpful in identification. For example, the Almada Naval Air Station is to the south of the road shown in the image. The knowledge about the relative positions of the objects are very vital for interpretation.

- *Texture* is produced by an aggregation of unit features that may be too small to be discerned individually on the photograph, such as tree leaves. These images are commonly found in remotely sensed images for forestry, drainage, etc. As the resolution of the image is reduced, the texture of an object becomes finer and ultimately disappears. The array of rectangular buildings shown in figure 1.2(a) is an example of a texture.

- *Time* is another important aspect of dynamic image interpretation. Temporal information is very helpful in identifying certain features whose basic features change over a period.

Though these are some of the basic characteristics of the images normally used in interpretation there are also additional items that are specific to applications. Also the knowledge about the objects present in the scene plays a major role in the present-day interpretation schemes. The focus of this dissertation is the knowledge-based image interpretation in natural scenes specifically in oceanographic satellite images.

In the first part of the dissertation, we explain the approach taken to develop an automatic interpretation system for oceanographic satellite images. The objective of our research at *Naval Oceanographic and Atmospheric Research Laboratory*

(NOARL), is to build a powerful automatic image interpretation system for oceanographic satellite images. More precisely, given an infrared image of the North Atlantic Ocean, the system should interpret the important features such as the north and south walls of the Gulf Stream, and the positions of cold and warm core eddies. In many cases, the user may have only minimal understanding of the oceanic features. Hence there is a strong motivation to develop an automatic image interpretation system to detect and describe the oceanic features. The major components that make up the system, their interfacing details and an illustrative example are given in chapters 2 and 3.

1.3 Image Processing Parallelization Paradigm

The purpose of this section is to address the fundamental issues and possible approaches for the second paradigm, namely, *Parallel and Distributed Processing*. Specifically, we focus on the motivation for performing the low level vision (digital image processing) tasks in parallel, and discuss the possible approaches to solve some of the problems in this paradigm.

Most digital image processing tasks are computationally intensive due to two reasons:

- amount of data to be processed is quite large
- algorithms are repetitive and time-consuming

For example, a typical digital image is of size 1024 X 1024 pixels (or bytes). Each image usually goes through several levels of processing, from the low level numeric operations to the high level symbolic processing. The algorithm for each level is quite time consuming. Therefore, the throughput required to meet this computational

demand is enormous. This is even more the case in real time applications where a sequence of image frames should be processed in a very short time.

It is interesting to note that some of the image processing tasks are easy to parallelize. For example, with the appropriate architectures, convolution, feature extraction, histogramming, and Hough Transforms can be performed in parallel. Mesh-connected computers based on SIMD model such as MPP or pipeline processors are able to handle these operations very efficiently. To effectively devote multiple processors to high level vision tasks, we expect to be led to shared-memory architectures, with each of a number of powerful processors capable of accessing all data representing the image. The important issue is to coordinate the multiple processors to complete the vision tasks efficiently and without redundancy.

There are two fundamental issues involved in designing parallel processing systems for image processing. They are:

- selecting an appropriate data structure for storing images
- selecting an appropriate architecture for processing

The reminder of this section is devoted to discuss these two issues in detail and explain the approach taken to address them.

There are two types of parallelism normally found in image processing applications:

- **data parallelism**

Exploiting the fact that many of the low level processing tasks are localized, the image data can be partitioned in such a way to realize parallelism. The focus of such

an approach is to provide a good data structure for storing images. *Quad tree* is a class of spatial data structures to store 2-D images. We have chosen quad tree as the storage structure. Figure 1.3 shows a sample digital image and its corresponding storage in the form of a quad tree. More on the actual research in storing and manipulation of images using quad trees is given in chapter 5. Recently, the methods that make use of data parallelism are gaining importance.

- **functional parallelism**

There are methods that exploit the inherent parallelism in the *computation*. The focus of these methods is to provide an intelligent partition of the image computations so that each one of these (partitions) may be executed in parallel. One such example is the use of "divide-and-conquer" strategy.

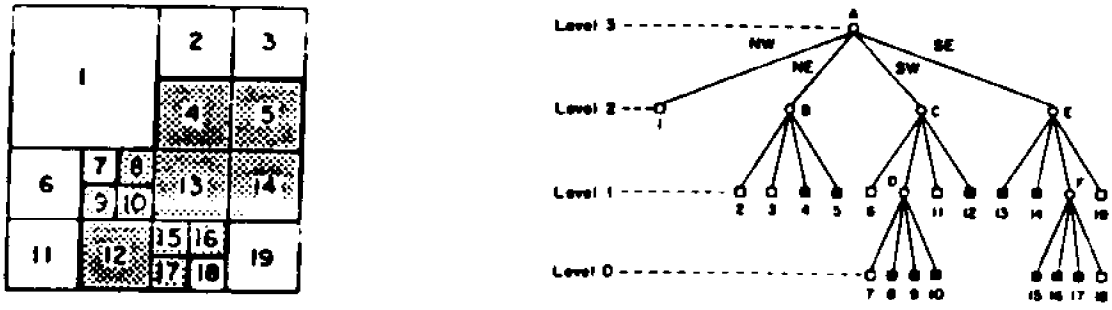


Figure 1.3 An Image and its Corresponding Quad Tree

The difference between the methods based on the above types of parallelisms is that the method based on the first can be applied to a variety of problems whereas the method based on the latter may be different for different problems. A good parallel image processing system would make use of both of these methods to achieve high performance.

The design and development of parallel architectures for image processing is an extremely important and difficult task. It requires careful examination, analysis, and development of both hardware architectures and software algorithms and programs. The most promising multi-computers and topologies used for image processing include *star networks, array processors, meshes, and binary n-cubes.* The exact descriptions and features of these architectures are discussed in chapter 4. Figure 1.4(a) shows a mesh-connected and 4.0(b) a hypercube multiprocessors.

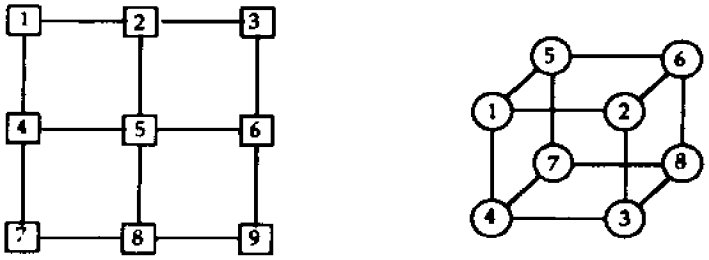


Figure 1.4 Mesh-Connected and Hypercube Multiprocessors

In this dissertation, we have chosen quad tree as the data structure for storing the images, and boolean hypercube as the underlying architecture for parallel processing. The details of our parallelization technique and the actual algorithms developed are the focus of the second part of this dissertation.

1.4 Contributions of the Dissertation

We divide this section into two parts. The first part presents the results in the area of natural scene understanding while the second part describes the results obtained in parallel algorithms for image processing. Figure 1.5 shows the contributions pictorially.

Automatic interpretation of features in a natural scene is quite complex. We have used techniques based on artificial intelligence to solve this problem. Our technique consists of a context dependent feature labeling algorithm using non linear probabilistic relaxation, and a rule-based expert system. Traditionally, the output of the labeling is analyzed, and recognized by a high level interpreter, for example, an expert system. In our *hybrid* approach, we make use of the knowledge about the scene (embedded in the expert system) to resolve the inconsistencies introduced by the labeling algorithm.

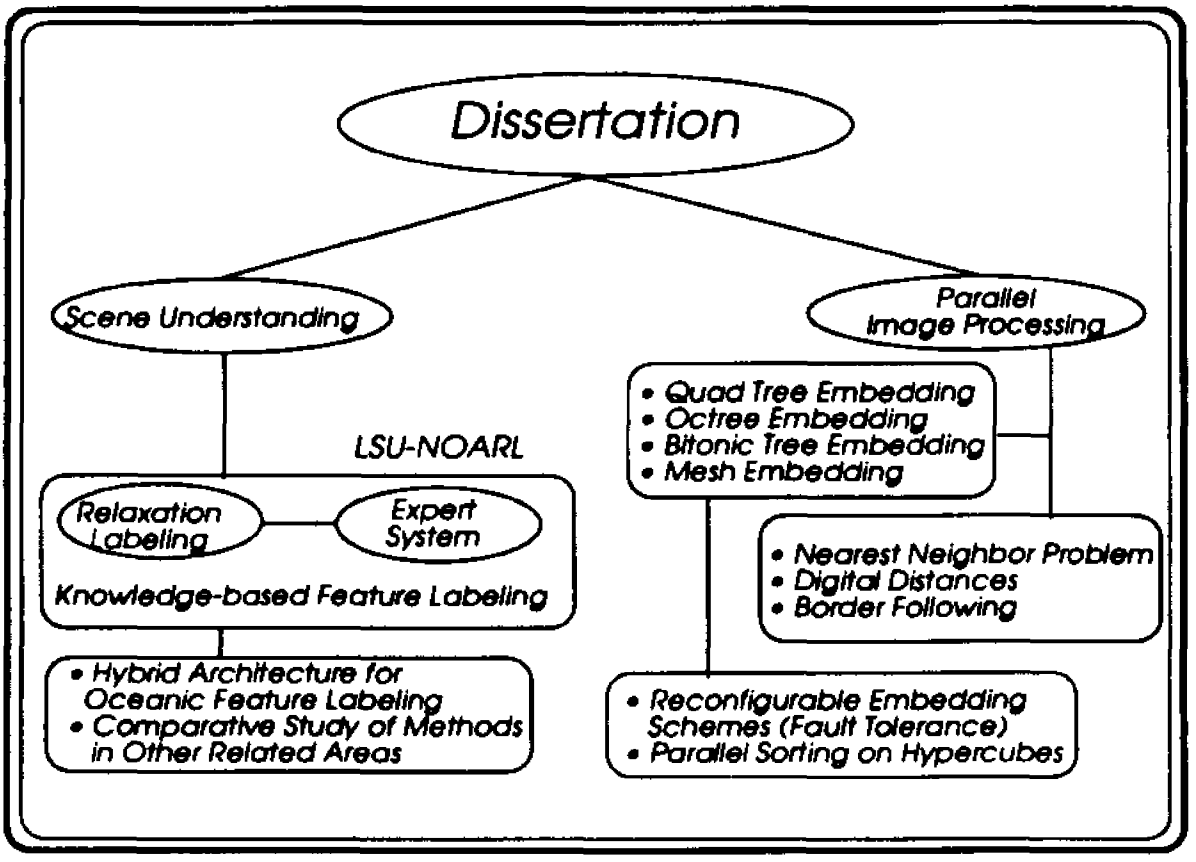


Figure 1.5 Contributions of the Dissertation

We have developed a feature labeling system based on this hybrid technique at NOARL, Stennis Space Center, Miss. Our labeling system plays a vital role in the

development of an automatic image interpretation system for oceanographic satellite images at NOARL. As a part of this research, we have made an extensive study on the existing interpretation techniques in the related areas such as remote sensing, medical diagnosis, astronomy, and oceanography and shown that our hybrid approach is unique and most powerful.

The second part of the dissertation presents the results in the area of *parallel image processing*. Efficient parallel algorithms can be developed by *embedding* the inherent data or computational structure of the problem to be solved. Based on this idea, we have designed parallel image processing algorithms for hypercube architectures. This is achieved by a clever mapping of pixels in a binary image into the processors of the hypercube.

As discussed in section 1.3, quad trees are an important class of spatial data structures and commonly used in image representation and storage. We have developed algorithms to embed quad trees into hypercubes. An important characteristic of this embedding is that the adjacent pixels are kept at a constant distance in the hypercube. Based on this embedding, we have designed parallel algorithms for the problems of *neighbor-finding and digital distances*. Similarly, we have developed embedding algorithms for *octrees* into hypercubes and shown that parallel algorithms for *three dimensional* image processing may be designed. As an example of embedding inherent computational structure, we have developed new *parallel sorting* algorithms for hypercubes based on a technique called adaptive bitonic sorting. The in-depth understanding of the problem of embedding has allowed us to develop efficient *reconfiguration* algorithms for hypercube architectures.

1.5 Scope and Organization of the Dissertation

The results obtained in the area of *scene understanding* constitute the first part of this dissertation.

Automatic interpretation of oceanic features is the focus of chapter 2. In this chapter we introduce the problems of interpreting oceanic features, the existing approaches, and the overall description of the various modules in our interpretation system. We have studied the existing approaches for interpretation in other related areas such as remote sensing, medical diagnosis, and astronomy. We compare the results of these techniques and show that our technique is unique and most powerful. The details on these comparisons are given in chapter 2.

In chapter 3, we describe the knowledge-based feature labeling technique in detail. We discuss the underlying concepts of our hybrid architecture for the labeling problem. We present the details of our implementation at NOARL Lab. We end this chapter with a short discussion on future research needed in this area.

The second part of the dissertation is devoted to results obtained in the area of *parallel image processing*.

In chapter 4, we first introduce the advantages of interconnection networks in parallel processing. We briefly discuss the problem of mapping algorithms onto architectures. The interconnection networks based on hypercube topology are taken as the architecture to explain our parallel algorithms. We have developed a novel technique for developing parallel algorithms for interconnection networks. The usefulness of our technique is demonstrated in this chapter, by providing a new sorting algorithm for hypercubes.

The key idea in developing parallel algorithms is to embed the underlying data or computational structure onto the interconnection network. We have chosen the quad tree and octree as example data structures to represent and store images. The results of embedding these structures into hypercubes are explained in chapter 5. We show how our embedding techniques have paved the way for the development of reconfiguration algorithms for parallel architectures.

In chapter 6, we provide parallel algorithms for some of the very often encountered problems in image processing such as neighbor-finding and digital distances. We explain our algorithms with the help of the theoretical results obtained in chapter 5.

We summarize the results in chapter 7. The need for further research in these two important areas is also discussed. The application of some of the techniques and algorithms developed in this dissertation to other problems is outlined in chapter 7.

We list all the references cited in this dissertation at the end of chapter 7.

PART I

SCENE UNDERSTANDING

Feature interpretation is vital for many of the existing scene understanding systems. Knowledge-based interpretation is becoming increasingly popular with the help of AI techniques. A system for automatic interpretation of features in oceanographic satellite images is described. The principal component of the system, the knowledge-based feature labeling algorithm, is explained. The proposed labeling technique uses a non-linear probabilistic relaxation scheme and a rule-based expert system. The performance of the labeling technique is compared with those of the existing systems.

CHAPTER TWO

AUTOMATIC INTERPRETATION OF OCEANOGRAPHIC SATELLITE IMAGES

2.1 Introduction

Computer-aided scene interpretation can be regarded as an *information processing task*. The *input* of such a task is a two-dimensional representation of the *scene* (the image) and the *output* a *symbolic* (non pictorial) *description* of the scene. Image and scene analysis techniques have been used a great deal for automated interpretation of digital imagery obtained from airborne or satellite sensors. The techniques used have been as varied as the types of imagery analyzed. A common analysis, usually associated with multi- spectral data (primarily from the Landsat series of satellites), has been the use of statistical techniques for the automated classification of farm crops, forest cover, etc. Similarly, infrared imagery of the sea surface from meteorological satellites has been analyzed using bayesian decision techniques to classify the sea surface into various oceanic water types.

Image understanding involves image processing as well as image interpretation tasks. Artificial Intelligence approaches have already been invoked as a response to solve the latter problem. Expert systems certainly constitute promising tools for image processing and interpretation. The essential features of an expert system are its modularity and its heuristic power. Modularity provides easy handling and updating of large data sets, while heuristic power allows solving the combinatorial explosion of facts by developing adequate reasoning strategies.

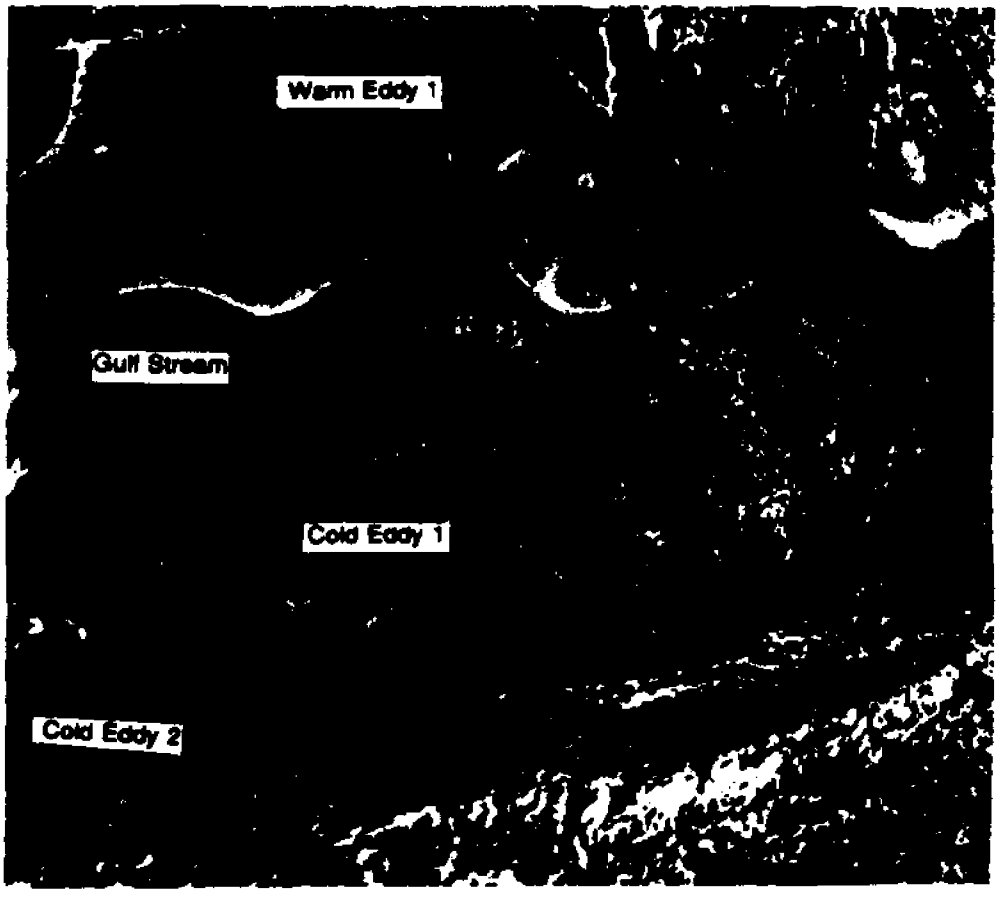


Figure 2.1 Sample IR Image of the North Atlantic Ocean

Satellite-borne sensors potentially offer many advantages for the study of oceanic processes. They provide global synoptic measurements of various oceanic surface properties, in contrast to the local measurements, possibly at a range of depths, provided by conventional oceanographic measurement techniques. *Thermal infrared*(IR) images of the ocean obtained from satellite sensors are widely used for the study of ocean dynamics. Figure 2.1 shows a sample IR image of the *Gulf Stream* obtained from the *Advanced Very High Resolution Radiometer* (AVHRR) aboard the NOAA-7 satellite. Brightness in this infrared image is inversely proportional to the ocean surface temperature (dark areas represent warmer temperatures and light areas represent colder temperatures). Vortices (areas of closed circulation) within this turbulent flow

pattern are called *eddies*. The Gulf Stream and its associated eddies are examples of mesoscale features ("mesoscale" is the name commonly applied to the features existing on spatial scales of the order of 50 to 300 km). Mesoscale features are important to the study of ocean dynamics, to fisherie and to many other diverse interests.

Current image analysis techniques rely on human interpretation of the satellite imagery. Human interpretation is obviously varied in its level of expertise and is highly labor-intensive. With the proliferation of high volume AVHRR image applications, it becomes highly desirable for certain applications to move from the labor-intensive manual interpretation of infrared imagery towards a capability for automated interpretation of these images. Several previous studies have addressed the automation of the analysis of infrared satellite imagery for mesoscale features. Gerson and Gaborski [Gers77a] and Gerson, et al. [Gers82a] investigated the detection of the Gulf Stream in infrared images from the Geostationary Operational Environmental Satellite (GOES). Gerson and Gaborski [Gers77a] used a hierarchical approach where 16x16 pixel (128x128 km) "frames" within the image were evaluated for the possibility of containing the Gulf Stream. Frames flagged as Gulf Stream possibilities were then further evaluated to determine the exact location of the Stream within the frame by looking at statistics based on 5x5 pixel "local neighborhoods". As an outgrowth of the work reported in [Gers77a] and [Gers82a] , Coulter [Coul83a] performed automated feature extraction studies using the higher resolution Advanced Very High Resolution Radiometer data. Janowitz [Jano85a] studied the automatic detection of the Gulf Stream eddies using Advanced Very High Resolution Radiometer data. Cornillon, et al. [Corn87a] describe procedures for processing large volumes of high-

resolution satellite IR images to obtain information for regional physical oceanographic studies. The underlying technique involves subjective steps. They have stressed the need for automating the process of interpretation.

Yet the problems involved in oceanographic image processing are formidable. All of the difficulties apply: oceanographic features move, change size, etc., there are no well-defined mathematical descriptions of their shapes, and cloud cover and other atmospheric effects typically interfere with clear observation of the features. Nichol [Nich87a] used a region adjacency graph to define spatial relationships between elementary connected regions of constant gray level called atoms. Although satisfactory emulation of human extraction of eddy structure is claimed for this method, Nichol [Nich87a] did point out that not all enclosed uniform areas identified by the method will correspond to real ocean structure. We addressed the problem of interpreting oceanic features, by first developing a feature labeling system based on *nonlinear probabilistic relaxation labeling* algorithm [Kris89a, Kris90a]. We [Kris89b, Kris90b] later proposed a *hybrid* architecture to solve the problem, which composes a feature labeling module and an expert system. A detailed description of our feature labeling system developed at NOARL is given in chapter 3.

2.2 Problem Statement

The objective of our research is to build a powerful automatic image interpretation system for oceanographic satellite images. More precisely, given an infrared image of the North Atlantic Ocean, the system should interpret the important features, namely, the north and south walls of the Gulf Stream, and the positions of cold and

warm core eddies. In many cases, the user may have only minimal understanding of the oceanic features. Hence there is a strong motivation to develop an automatic image interpretation system to detect and describe the oceanic features.

The automated interpretation system has five major components. They are:

(a) A *low level edge detector* - this edge detector is based on the gray level co-occurrence matrix and found to exhibit the fine structure rejection while retaining edge sharpness. The details on the edge detector is given in section 2.3.1. We also discuss another edge detector that takes advantage of both the regional approach and local approach while avoiding their drawbacks.

(b) A *feature labeling algorithm* - the edges that are located by the edge detector are labeled with the help of a nonlinear probabilistic relaxation technique. This technique blends the conventional image processing algorithms with the contextual information about the features. The details on the feature labeling scheme is given in section 2.3.2.

(c) An *expert system* - the positional information about the features are evaluated by the expert system for any inconsistency. This is a rule-based expert system that has a compilation of the knowledge about the oceanic features. Section 2.3.2 discusses the individual components of the expert system.

(d) A *Hough Transform* - the features that are identified as part of eddies are fitted into smooth circles with the help of Hough transforms. Section 2.3.4 describes the functionalities of this module.

(e) A *Complex-EOF module* - the features that are identified as north and south walls

of the Gulf Stream are fitted into a smooth curve using a set of complex-eof functions. The details of this module is given in section 2.3.5.

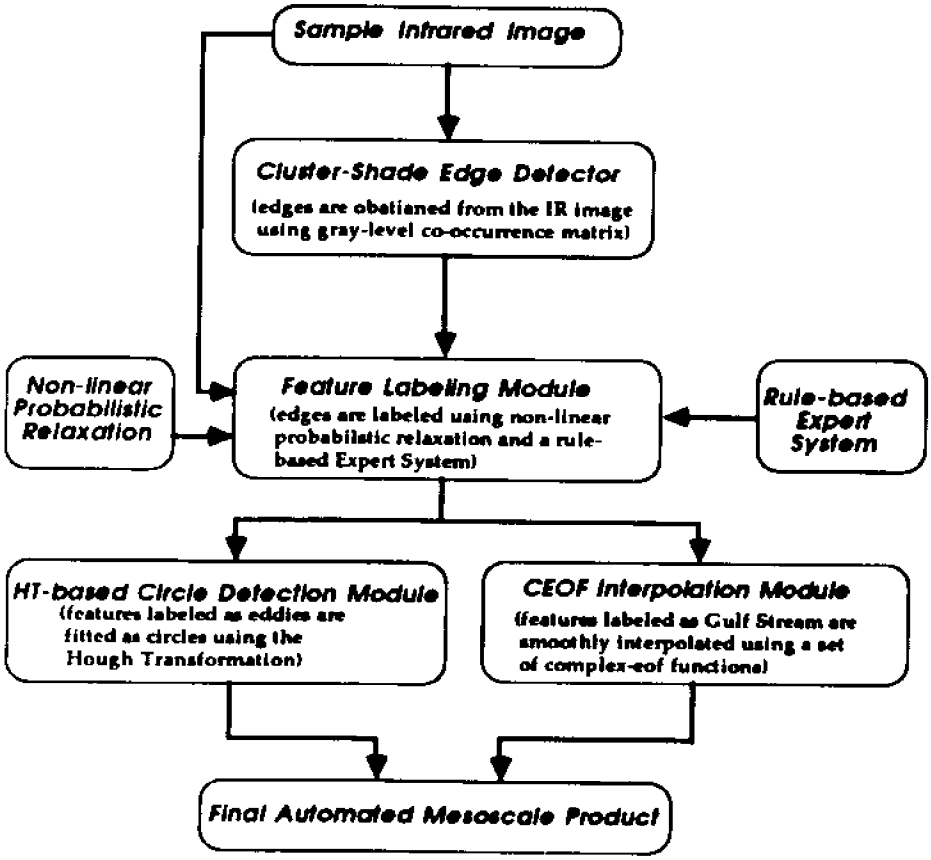


Figure 2.2 Modules of the Oceanic Interpretation System

2.3 Modules of the Interpretation System

The modules of our automatic interpretation system are described in this section. The modules and their interactions are pictorially given in figure 2.2.

2.3.1 Cluster Shade Edge Detector

In this section, we first discuss the features of the edge detection algorithm proposed by Holyer and Peckinpaugh [Holy87a]. The motivation behind the

development of such a new edge detector algorithm is to aid the analysis of oceanographic satellite images. The popular derivative-based edge operators viz Sobel's operator are shown to be too sensitive to edge fine-structure and to weak gradients to be useful in this application. The edge algorithm proposed by Holyer and Peckinpaugh [Holy87a] is based on the cluster shade texture measure, which is derived from the gray level co-occurrence matrix (GLC). The authors [Holy87a] have suggested that the edge detection technique based on the GLC matrix can be effectively used in automated detection of mesoscale features. The (i,j) th element of the GLC matrix , $P(i,j \mid \Delta x, \Delta y)$ is the relative frequency with which two image elements, separated by distance $(\Delta x, \Delta y)$ occur in the image, one with intensity level i and the other with intensity level j. The elements of the GLC matrix could be combined in many different ways to give a single numerical value that would be a measure of the edges present in the image. Holyer and Peckinpaugh [Holy87a] have used a cluster shade function which is found to be very effective in the edge detection process. The new edge algorithm computes the cluster shade function at each pixel. Then the edges are detected by finding the significant zero crossings in the cluster shade image.

The advantages of this new edge algorithm over the conventional derivative-based techniques are discussed in [Holy87a]. It is known that using large windows in derivative-based edge detector algorithms results in poor smoothing. This problem is circumvented in the new algorithm. Because edges are detected by finding zero crossings, precisely positioned lines result, even if the GLC matrix is calculated using a larger window. So, the desired edge detection characteristics of retaining sharp edges while eliminating edge detail is achieved with the help of the new algorithm. As an

input to our feature labeling algorithm, we used the image output generated by cluster shade algorithm, with a window size of 16x16 pixels and zero crossing threshold of 50. The edge magnitudes obtained from this new edge detector algorithm are used as an input to the feature labeling algorithm. In particular the edge magnitudes are used to evaluate the a priori probability values. Figure 2.3 gives the output of the edge detector when applied to the oceanic image shown in figure 2.1.

Figure 2.3 Output of the New Edge Detector

Cayula and Cornillon [Cayu90a] have developed an edge-detection algorithm for oceanographic satellite images. Their algorithm operates at three levels : picture level, window level, and local/pixel level. In the picture level, most obvious clouds are identified and tagged so that they do not participate at the lower levels. This is done

based on temperature and shape. Using techniques for unsupervised learning, the temperature distribution in each window is analyzed to determine the statistical relevance of each possible front. Finally, local edge operators are used to complete the contours found by the region-based algorithm. Since the local operations are used along with the window-based algorithm, the qualities of scale invariance and of adaptivity associated with the region-based approach is not lost. In [Cayu91a], the edge-detection algorithms described above are compared, and the results are discussed. The main difference is the algorithm given by Holyer and Peckinpaugh [Holy87a] operates at the local level only, while that by Cayula and Cornillon [Cayu90a] is a multilevel algorithm. Refer to [Cayu91a] for the details on the results of the comparison.

2.3.2 Knowledge-based Feature Labeling

Given the edge detector output from processing an oceanographic satellite image, the problem is to assign to each edge segment a label using certain knowledge. The labels are "north wall", "south wall", "warm eddy", "cold eddy", "shelf front", and "other". The knowledge to be used to direct the labeling such things as results of a previous analysis (temporal continuity), temperatures from the original image, bathymetry, rules on the general oceanographic context, and Gulf Stream climatology.

The knowledge-based feature labeling module consists of two sub modules: (a) non-linear probabilistic relaxation labeling, and (b) feedback from the expert system module. The first module uses relaxation labeling technique for initial labeling. This consists of two steps: (i) estimating the initial feature probabilities, and (ii) iteratively updating the probabilities. The initial probabilities are assigned with the help of

previous analysis, and the ground truth data. These probabilities are then iteratively updated using the non-linear probabilistic relaxation scheme.

The labeled features are then fed to the expert system module. This expert system is divided into two submodules. The first submodule consists of a knowledge base and an inference engine. The knowledge base has a thorough collection of facts and rules about the mesoscale features such as gulf stream, warm core rings etc., Based on the positional information obtained from the feature labeling module and the knowledge base, the inference engine attempts to interpret the dynamics of these features. Basically the inference engine is a pattern matching module.

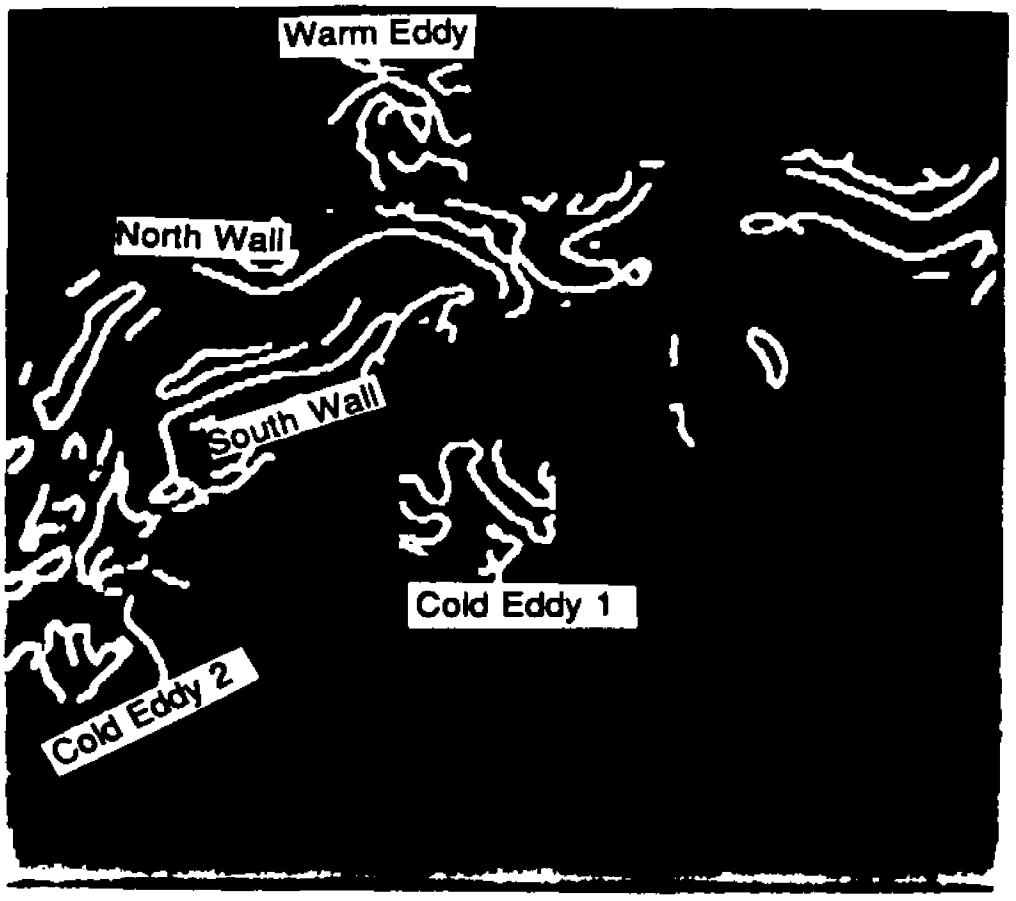


Figure 2.4 Output of the Feature Labeling Module

The second submodule evaluates the consistency of the labeling process with the help of the knowledge base as well as from the input from the operator. This evaluation results in assigning a confidence factor to each of the features detected. This factor is then "fed back" to the feature labeling module, in addition to the previous data analysis, to improve the consistency of the labeling process in future. Figure 2.4 shows the output of the feature labeling module.

A salient feature of this hybrid architecture is the feedback from the knowledge based system to the labeling module. Due to atmospheric effects like cloud cover, the oceanic features may not be identified by the feature labeling module. The position of the Gulf Stream and eddies may not be determined correctly. In such cases, the expert system provides an approximate position of these features. This helps in interpreting the subsequent images. Another feature is the dynamic interpretation of the features. The hybrid architecture allows the expert system to learn from previous analysis. Also the modular approach allows the user to maintain the system and incorporate changes without much difficulties. Details of our hybrid approach are explained in chapter 3.

2.3.3 Expert System for Ocean Dynamics

Interactive image processing is a powerful tool in the hands of an expert image interpreter. However, human interpretation is both varied in its level of expertise and labor-intensive. Expert judgement is required for oceanographic image interpretation because there is little or no standardization in either image enhancement or interactive techniques. NOARL has conducted research in the use of concepts from the field of artificial intelligence to develop knowledge-based "expert systems" to automate some

aspects of the oceanographic image interpretation task. That work is an attempt to remove human-labor-intensive and varying-skill-level elements from the interpretation of oceanographic image and other satellite data. Lybanon, et al. [Lyba86a] describe the motivation and some of the history of NOARL's work in this area.

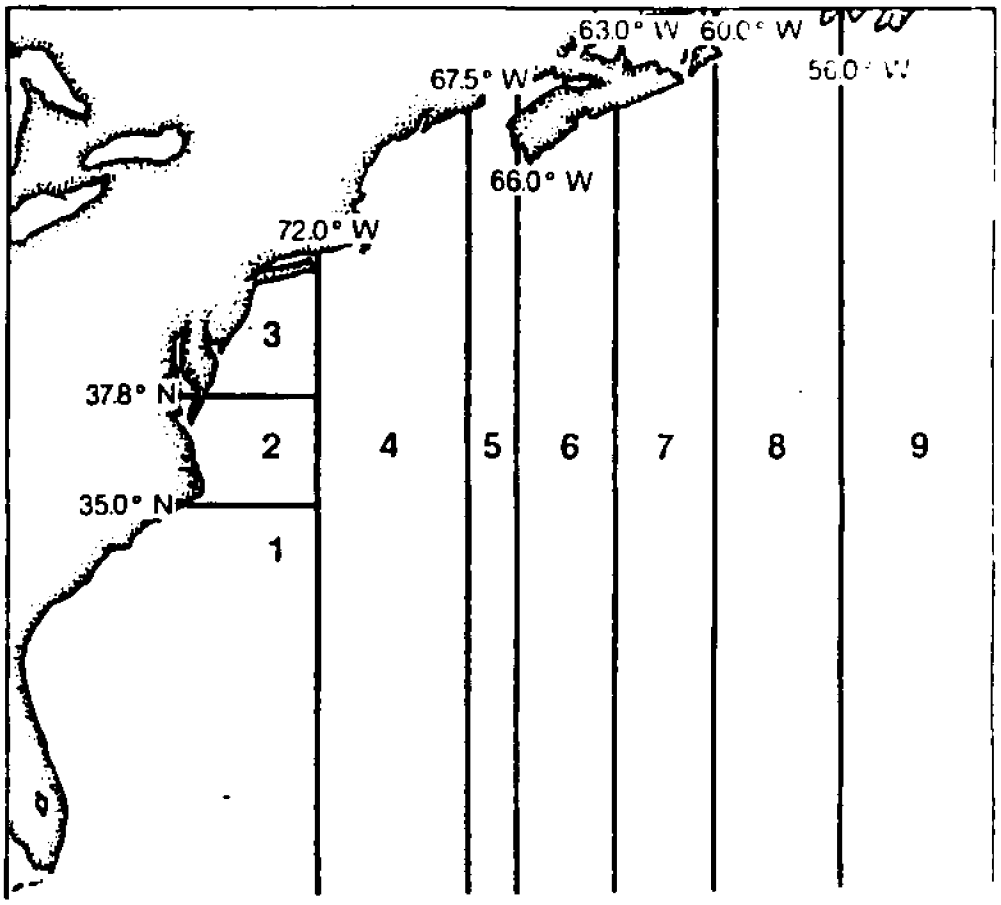


Figure 2.5 Operating Regions of the Expert System

The rule base of the expert system is derived from a published compilation of information, and the expert system's mode of operation is innovative. The rule base represents oceanographic knowledge about the evolution of mesoscale ocean features in the Gulf Stream region of the north Atlantic Ocean. The rules are applied in such a way that the expert system describes the kinematics of that evolution. The expert system is presently implemented in a combination of OPS83, C, and FORTRAN running

on a VAX 8300. The expert system uses the rules about the features to evolve an initial "state" to a later time. The new, hypothesized state can then be used as a new initial condition and the process is repeated. This process can be usefully carried out for several steps. This mode of operation is different from expert systems which use a (sometimes lengthy) chain of reasoning to reach a single conclusion, such as a system for medical diagnosis.

The domain of the expert system is the Gulf Stream region of the north Atlantic Ocean. The expert system has different rules for ring and Gulf Stream behavior in each of nine geographical regions. Figure 2.5 shows the operating regions of the expert system. Aside from the lines that separate them, the regions are bounded only by land. A ring's behavior as hypothesized by the expert system depends upon which region its center is in at the beginning of a time step. Basically, the motion has a region-dependent velocity vector. However, a ring that is closer than a certain critical distance from the Gulf Stream undergoes a modification of the basic motion. The Gulf Stream interaction rules are also region-dependent. The details depend upon how close the ring is to the Gulf Stream; the Gulf Stream interaction may result in a deflection or a looping motion, with possible coalescence with the Stream in some cases. Ring sizes decrease with time. The rate of decrease depends on the region and on whether there is Gulf Stream interaction. A ring that shrinks below a certain size disappears. Gulf Stream motion is modeled as downstream propagation of meanders, with region-dependent phase velocities and amplitude factors. While this is only a first-order model, there is justification in the literature for this behavior. Refer to [Lyba86a] for a complete description of the expert system.

2.3.4 Hough Transforms for Eddy Detection

Eddies are important features in oceanographic satellite images. Detection of eddies constitute a major component in interpreting these images.

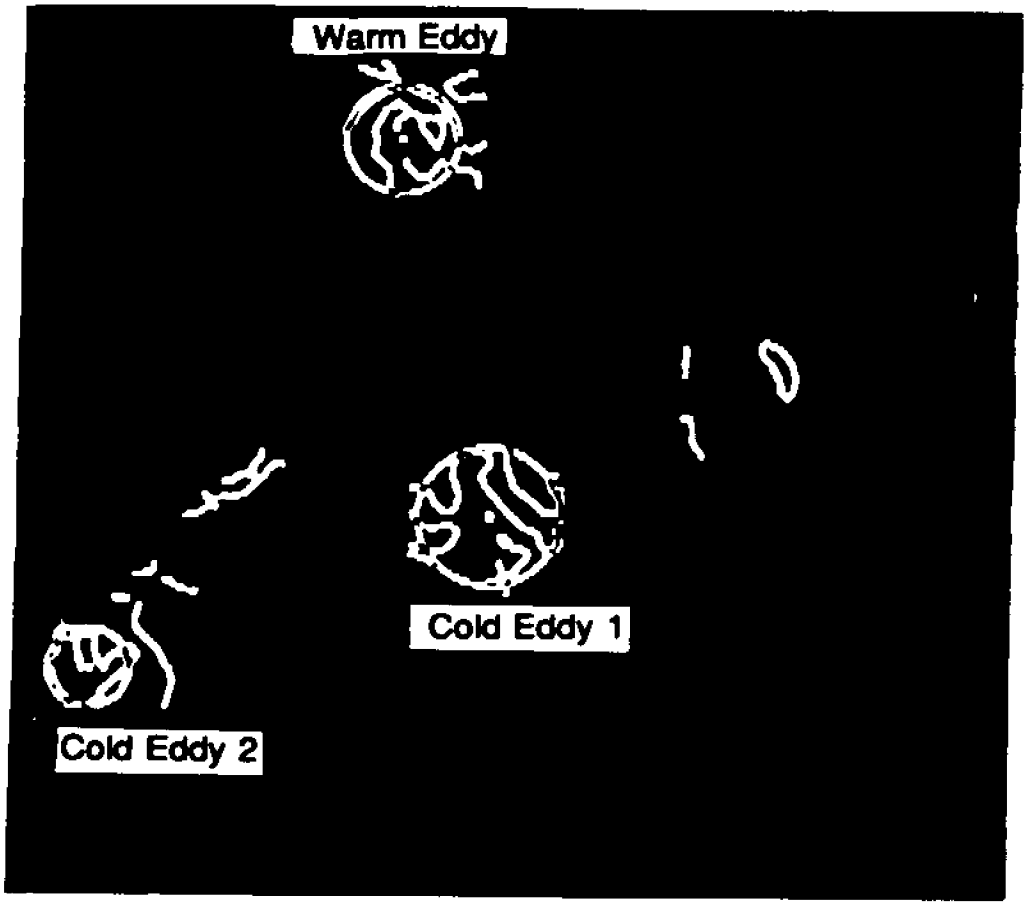


Figure 2.6 Output of the HT-based Eddy Detector

Conventional dilation/erosion operations have the following disadvantages: (a) they fill only small gaps; (b) dilation/erosion results in appendages to the resulting lines that have no physical significance, but are merely artifacts of the processing; (c) dilation/erosion does not take advantage of any a priori knowledge about the shape and size of eddies. In order to overcome all of these problems, a circle finding procedure based on the Hough Transform has been proposed. The Hough Transform is a technique commonly used in the field of computer vision for detecting lines and

shapes in digital imagery. The transform considers all possible instances of a given curve and rates each on how well it fits the data. This technique has been employed to detect tumors from chest radiographs in [Kimma]. Cross [Cros88a] has used the Hough Transform for the first time to detect circular geological features. Based on these results, we have developed an eddy-detection module that utilizes the circular Hough Transform. The module outputs the estimated center and radius of an eddy found in the labeled image. Figure 2.6 shows the output of the HT-based eddy detector.

2.3.5 Complex-EOF for Gulf Stream Detection

The edge pixels that are labeled as part of north and south walls are then interpolated using an optimization procedure. Figure 2.7 shows the output of this module. Molinelli and Flanigan [Moli87a] have developed this procedure based on complex empirical orthogonal functions (CEOFs). The CEOFs are used to remember Gulf-Stream-like shapes and these shapes are in turn used to interpolate realistic Gulf Stream positions between fixes by IR and *altimeter* sensors. A satellite altimeter measures the range from the satellite to the sea surface based on travel time of a radar pulse [Cant90a]. The output of the feature labeling module described in section 2.3.2 contains Gulf Stream inflection points. Each inflection point is identified as being based on an altimeter fix or, if no altimeter fix is available, on either an IR fix or an "estimated" location.

Carter [Cart85a] has demonstrated the advantages of using CEOFs to model and analyze the Gulf Stream. There are two inherent difficulties with the conventional

approach. They are: (a) Gulf Stream is inhomogeneous in nature and (b) it may be multiply valued in both latitude and longitude.

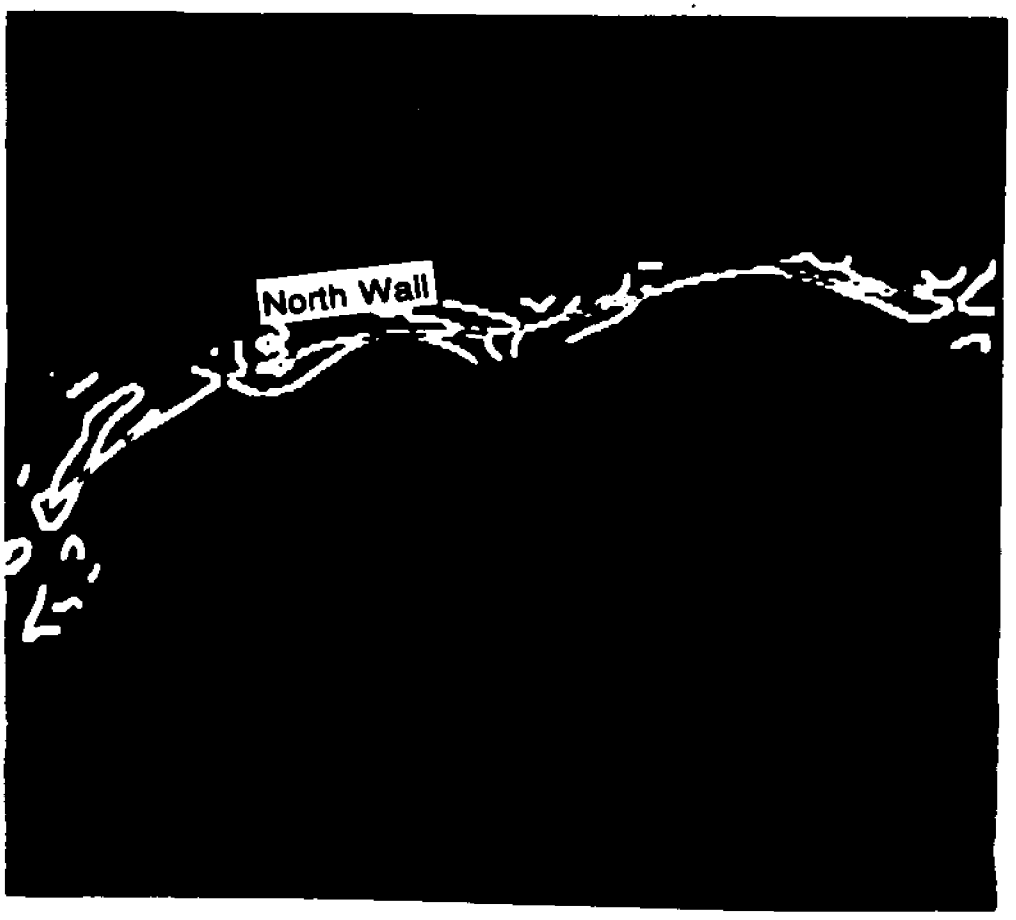


Figure 2.7 Output of the CEOF-based Gulf Stream Detector

The interpolation procedure developed by Molinelli and Flanigan [Moli87a] can handle Gulf Stream axes multiply valued in both latitude and longitude by making the distance downstream as the independent variable of their CEOF analysis. The CEOFs used in their procedure can be used to describe Gulf Stream shapes well with usually few (10) modes. These modes can be optimized from initial values with as few as 21 fixes on the position of the Gulf Stream axis, a number well within the usual amount (25 to 75) available from infrared and altimetric observations. The performance of this optimization procedure is compared to the existing procedures in [Moli87a] and

shown that this interpolation procedure works well in most of the cases.

Figure 2.8 Output of Automatic Interpretation System

The details of the mathematical framework, the descriptions on the experiments and simulations and comparison of results of this module is given in [Moli87a]. Recently Cantrell and Holyer [Cant90a] have developed an automated technique to locate the Gulf Stream in altimeter profiles. The results of this experiment is very encouraging. As a part of our research we plan to investigate this technique in combination with the feature labeling technique described in section 2.3.2. Figure 2.8 shows the final output of the automatic interpretation system for oceanographic satellite images.

2.4 Comparison of Interpretation Systems in Related areas

We consider application areas where natural scene interpretation is very vital. Figure 2.9 depicts the scope of our comparative study of existing systems in the areas of remote sensing, medical image processing, astronomy and oceanography. In this chapter we present the results of our comparative study on systems developed in these areas and highlight the advantages and features of our system.

There is a continuing need for efficient machine implemented analyses and interpretations of remotely sensed data. A review of the current approaches to machine interpretation of remotely sensed data is presented in [Tail86a]. Section 2.4.1 addresses the problem of interpreting remotely sensed imagery. It is generally believed that decision making in biomedical expert reasoning usually involves the use of heuristics from domain knowledge sources [Clan84a, Buch84a, Haye83a]. Most biomedical image interpretation systems use the classical statistical pattern recognition strategy in which a feature hyperspace is constructed for the problem followed by a statistical discriminant analysis. Knowledge based biomedical pattern recognition has been well studied by Wu, et.al [Wu89a]. We discuss the problems that are encountered in biomedical image interpretation and the approaches taken to solve them in section 2.4.2. Traditionally, the classification of galaxies by morphological type is accomplished through the use of human judgement exercised in the visual inspection of astronomical images. The outburst of a number of sensors that are provided by satellites, now calls for the automated interpretation of galaxies. Kurtz, et.al [Kurt90a] have developed a cognitive system for astronomical image interpretation. Section 2.4.3 describes the various approaches taken by researchers in the area of

astronomy to solve the interpretation problem. Satellite imagery of the oceans has become an invaluable tool for the oceanographer. The derivation of mesoscale ocean information from satellite data depends to a large extent on the correct interpretation of infrared oceanographic images. The problems that are associated with interpreting these features in the oceanographic images are already addressed in section 2.1 and 2.2. The various techniques used to solve the problem of segmentation/interpretation in these areas are summarized in figure 2.9.

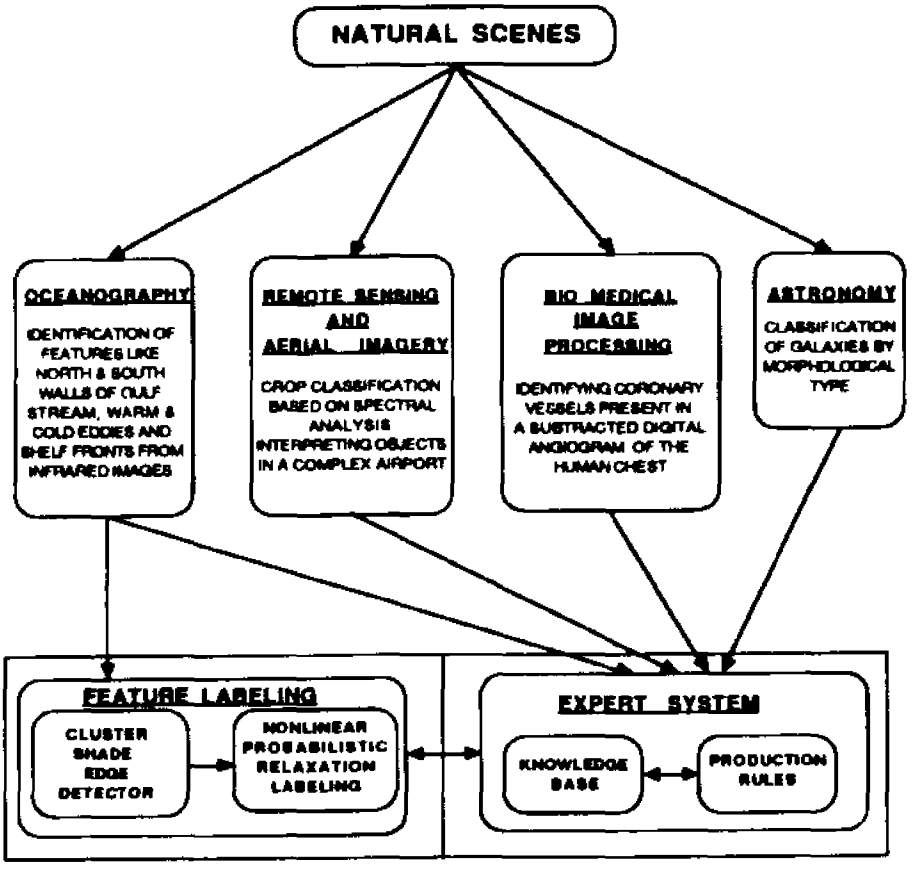


Figure 2.9 Study of Interpretation Systems in Related Areas

2.4.1 Remote Sensing and Aerial Imagery

Remote sensing and aerial imagery have wide applications in geological and soil mapping, land use/land cover mapping, agriculture, forestry, water resources planning, urban and region planning, and archeology. In most of these applications, the digital image of the scene under study is obtained from sensors positioned in a low flying aircraft or a satellite. The amount of digital data involved in remote sensing applications is quite large. Hardy [Hard85a] points out that the Multi-Spectral Scanner (MSS) on Landsats 1 and 2 produced 5 X 10^3 bit km^{-2} (208 pixels X 6 bits X 4 bands), whereas the recently launched Landsat-5 thematic mapper (TM) generates 5.4 X 10^4 bits km^{-2}. Apart from these satellite sensors, a considerable amount of digital data may be obtained from aircraft. The acquisition of large amounts of data from these sensors has also increased the demand for computer-aided interpretation of these remotely sensed images.

Most current machine analysis of remotely sensed multispectral imagery still relies solely on the supervised spectral analysis of pixels on an individual basis. Goldberg [Gold81a] surveyed some of the digital image processing techniques used in remotely sensed imagery, particularly, Landsat. It is also stated that multitemporal images are very useful to perform image analysis [Gold81a]. Temporal analysis, the exploitation of spatially registered imagery of different times, adds a new dimension of information. Studies have shown that temporal variations are clearly evident in Landsat imagery and can be effectively used. A good review of temporal analysis techniques for Landsat imagery is given in [Anut73a]. Haralick [Hara82a] discussed the problems in the area of pattern recognition of remotely sensed data, and surveyed

the most commonly used techniques to solve them. The most important steps involved in recognizing patterns in remotely sensed images are given in [Hara82a]. They are :

(a) define the class of entities of interest that discriminate the objects

(b) choose instruments or sensors that measure the environment in which the objects occur

(c) provide methodology to recognize an object in the class of objects of interest from those not in the class of objects of interest.

(d) construct a decision rule that identifies a particular object.

The paper [Hara82a] describes methods based on statistical pattern recognition and decision theory. Interpreting remotely sensed images based on AI techniques has become increasingly common. The survey on this area given in [Tail86a] agrees upon making use of scene knowledge to provide good and less ambiguous interpretations. Although the conventional classification methods are successful in certain cases, they fail to produce satisfactory results in many areas of remote sensing. As well as errors in classification existing, to various degrees, these methods have difficulty in handling contextual information and do not integrate well with ancillary data such as that found in geographical information systems. The most serious shortcoming of these techniques is that no knowledge is used or given about the objects in the scene. Tailor, et al. [Tail86a] have given a unified framework to design a knowledge based interpretation system for remotely sensed imagery. They discussed various approaches taken to solve the problem and weighed their merits and demerits within the remote sensing frame. It is stated that the area of knowledge-based image interpretation needs to be made stronger [Tail86a].

Aerial imagery is one of the important areas of remote sensing and needs more attention. One typical application of aerial imagery is the task of airport image analysis. Mckeown, et al. [McKe85a] have addressed the photointerpretation problem and developed techniques to solve it in a complex airport environment. The authors [McKe85a] believe that the integration of map knowledge, image processing tools, and rule-based control and recognition strategies will be shown to be a powerful computational organization for automated scene interpretation in aerial imagery. A number of techniques and systems have been developed to interpret aerial images [Broo81a, Naga79a, Mcke83a, Naga80a]. ACRONYM [Broo81a] is an example of a model-based system which incorporates viewpoint-insensitive mechanisms in terms of its model description. It maps edge-based image features to instances of object models. SPAM [McKe85a] (System for Photo Interpretation of Airports using MAPS) is an image interpretation system that makes use of domain-specific knowledge about the airport scenes. SPAM has three system components: an image/map database called MAPS, image processing tools and the rule based system. The MAPS [Mcke83a] database stores facts about man-made or natural feature existence and location which allows SPAM to perform geometric computation in *map space* rather than *image space*. The image processing tools that are used in SPAM are: a region-growing segmentation module, a road/road-like feature follower, a stereo analysis program and an interactive human segmentation system [McKe85a]. These tools identify feature-based islands of interest and help in further interpretation using the rule based system. Finally, the rule-based system guides the scene interpretation by generating successively more specific expectations based on image-processing results. Venkateswar

and Chellappa [Venk90a] have proposed a framework for interpreting aerial imagery. In their approach, the knowledge about the scene is represented using frames and the domain problem solving knowledge is in the form of production rules. Frames is an efficient knowledge representation language that has *slots* that can be filled with *fillers*. Hierarchical arrangement of data is a useful property of frames. The system uses an Assumption based Truth Maintenance System (ATMS) [Klee86a] for an efficient knowledge search to reduce ambiguities during the interpretation process. Chen, et al. [Chen88a] have described a system to interpret forest radar images by combining radar images with TM images. They use conventional classification and autocorrelation techniques for interpretations.

Segmentation of forward looking infrared images (FLIR) is given in [Hadd90a]. Haddon and Boyce [Hadd90a] developed a new segmentation by unifying region and boundary information present in an aerial image. The underlying segmentation algorithm consists of two stages. An initial segmentation is made based on the location of the intensities of each pixel and its neighbor within the co-occurrence matrix. Each pixel then has an associated tuple of probabilities: probability that it belongs to a region and the probability that it is a boundary pixel. A probabilistic relaxation labeling algorithm is then used to refine this initial segmentation. Again, the co-occurrence matrix is used to evaluate the compatibility coefficients for the relaxation labeling to ensure consistency between the two stages of segmentation. The paper [Hadd90a] points out that the unifying segmentation approach has proven to be good for images composed of regions which may be described by Gaussian statistics. The method is shown to be effective for a sequence of infrared images of natural terrain. Interest-

ingly, the research work reported by [Kris90a], follows a similar line of attack to solve the problem. Whereas the method used in [Hadd90a] depends on the feature information available in the image, Krishnakumar, et.al., [Kris90b] have made use of the contextual information that is embedded in the image. The problem addressed in [Kris90b] is quite complex, since the oceanic features are not stationary and the interpretation technique must take into account this aspect, namely, the dynamics of the ocean.

Expert system-based image interpretation is found to be an effective approach in the area of remote sensing. Many researchers [Good87a, McKe85a] have attempted to solve the photo interpretation problem in remotely sensed imagery. In [McKe85a], a rule-based system that uses map and domain-specific knowledge to interpret airport scened is presented. The system could extract and identify some runways, taxiways, grassy areas, and buildings from region-based segmentations. Based on these primitives, multiple plausible functional areas descriptions were developed. Finally, a layout of the airport is generated with the help of these multiple models. It is reported in [McKe85a] that both machine and hand segmentations compare favorably and appear to give a good description of the airport scene. Goodenough et.al., [Good87a] has developed two hierarchical expert systems, the Analyst Advisor and the Map Image Congruency Evaluation Advisor (MICE). The paper [Good87a] describes the architecture of the expert system to compare maps and images (MICE) and the expert system to advise on the extraction of resource information from remotely sensed data, the Analyst Advisor. The MICE expert system basically studies the differences between maps and images. The Analyst Advisor advises the user how to utilize the existing

image processing hardware and software in order to obtain desired resource information.

2.4.2 Medical Image Processing

Medical image processing is a branch of general image processing, and uses techniques, as such, generally available. While the signal usually sought in an image processing technique is reasonably well-defined, often, in medical image processing, it is poorly defined. Todd-Pokropek [Todd82a] explains the difference in the complexity of the problems encountered in medical images and other 'general' images with the help of an example segmentation of nuclear cardiology. The problem is to determine the volume of the left ventricle at various points in the cardiac cycle. Various approaches taken in solving such problems have been discussed in [Todd82a]. It is mentioned in [Todd82a] that a common problem in medical imaging is, knowing (roughly) where an object is, trying to define exactly what it is. So the use of a priori information is essential in solving these problems. Use of temporal information has proven to be very effective when cardiac images are to be analyzed [Todd82a]. Though the paper [Todd82a] discusses the problems associated with medical image processing, very little is said about interpreting these images with the help of artificial intelligence techniques. Levine, et al. [Levi83a] describe a rule-based image interpretation system for moving blood cells. The objective of their work was an image interpretation system capable of analyzing the structural changes in the morphology of cells from a sequence of pictures. The system was used successfully to analyze and study the pseudopod kinetics of white blood cells. There exist systems to analyze and

recognize blood vessels in angiograms [Stan86a, Wang87a, Cley88a]. Stansfield [Stan86a] developed a rule-based expert system for subtracted angiograms. Given a subtracted digital angiogram of the chest, ANGY [Stan86a] identifies and isolates the coronary vessels excluding irrelevant information present in the angiogram. Cleynenbreugel, et al. [Cley88a] have improved the segmentation of blood vessels on subtraction angiograms. Blood vessel segmentation plays a major role in many of the applications of medical imaging-based diagnosis. The knowledge based system described in [Cley88a] consists of three parts. The first part uses geometrical information about the line patterns. The second part utilizes the global knowledge about the blood vessels. The third part is built around domain specific knowledge partly available from radiologists. The paper [Cley88a] discusses the system in detail and also distinguishes the approach from that of ANGY [Stan86a].

Stansfield [Stan86a] has designed and developed a rule-based expert system for segmenting angiographic images. Segmentation of the coronary vessels from the digital angiograms has shown to be a very important step in interpreting these images. Stansfiled [Stan86a] has strongly emphasized the need for an expert system to help in this segmentation process. There are three main modules in the system proposed: the processing stage and the two stages embodied in the expert itself. In the low level processing stage the angiographic image is segmented with the help of edge and region analysis. The expert system is divided into two independent stages. In the low-level stage, a domain independent knowledge of image segmentation, grouping, and geometric relations is applied to the segmented image created by the processing stage. The second stage, then applies domain-dependent knowledge of cardiac anatomy and

physiology to interpret the segmented image. In this scheme, anatomical knowledge is embodied within the system in the form of the spatial relations between objects and the expected characteristics of the objects themselves. The ANGY [Stan86a] system is essentially a data-driven, or bottom up system. This is because the images contain a great deal of noise when the images are not complex and the cardiac anatomy varies greatly from subject to subject.

2.4.3 Astronomy

Astronomy is an observational science. There are three basic stages in astronomical image processing [Sedm84a]. They are:

1. Removal of instrumental signature
2. Processing aimed at visual inspection of the observations
3. Application of oriented processing aimed at any specific set of scientific tasks.

The astronomical observations are made in a spectral range extending from gamma-rays to metric radio wavelengths including exotic tests on neutrinos and gravitational waves. The constraints set by the levels of sources and background and by the technologies available for astronomical observations lead to a rather consolidated scene in the UV to near-IR and radio image, while the technological situation is rapidly evolving in gamma-ray, X-ray, and IR astronomy. There are two main classes of objects: those limited and those non-limited by the observational point spread function (PSF) [Sedm84a]. The segmentation and classification of two-dimensional images are straightforward for PSF-limited objects, while difficult or very difficult for practical astronomical images of extended blurred objects.

The present-day computerized astronomy is focussed on recognizing these extended blurred objects. Segmenting these is aimed at visual inspection and discrimination for classification. Though all fundamental image processing techniques like linear filtering, histogram processing, slicing, color coding, etc., are used in astronomy, they are not helpful for really quantitative discrimination and classification of fainter objects. This is because the fainter objects are the smaller ones and those more affected by blurring due to observational PSF. Sedmak [Sedm84a] also points out that the processing of astronomical objects is done on single spectral bands one at a time, even when images taken in several bands are available. There are very few systems developed for automatic interpretation of astronomical images. We feel that there should be more research done to address this fairly difficult problem.

The classification of galaxies by morphological type is an important problem of astrophysics. At present, this classification is accomplished through the use of human judgement exercised in the visual inspection of astronomical images. Kurtz, et.al [Kurt90a] have stressed the need for automating this interpreting task. They [Kurt90a] have described a cognitive system for astronomical image interpretation. Their system is dynamic in the sense that the syntactic, statistical and human judgement methodologies are integrated in a useful manner. The system proposed is based on what is known as *Redescription Structure* which makes the logic of data analysis and data reduction explicit. For a full description of the interpretation system, refer to [Kurt90a]. Malagnini, et.al. [Mala84a] have described a prototype system for faint object discrimination in astronomy. The method of analysis through a textural property description like the co-occurrence matrix, has proven efficient for discriminating

faint astronomical objects. Lea and Kellar [Leaa] describe a method to detect an object against the background sky in astronomical images. They apply the algorithm to the problem of classifying objects such as stars or galaxies. The technique is based upon the principles of mathematical morphology [Serra], specifically the image transformations of opening and closing, which were developed by Serra [Serra] in connection with texture analysis of digital images. The algorithm uses only integer arithmetic to smooth edges and intensities and find the edges of the objects. The authors illustrate that their method finds astronomical objects about as well as a standard (more complicated) photometric reduction program.

Technique	Area	Source of Image	Objects of Interest	Levels of Processing		
				Low	Intermediate	High
Krishnakumar, Iyengar, Holyer and Lybanon	OG	TIR	Gulf Stream Eddies Shelf Fronts	Cluster Shade Edge Detection	Non linear Probabilistic Relaxation Labeling, CEOF interpolation	Rule-based Expert Systems
McKeown, Harvey and McDermott [11]	RS	Aerial Photo.	Runways Terminals roads etc.	Region-Growing Segment--ation	Feature Follower, Stereo Analysis	Rule-based Production System
Kurtz, Mussio and Ossorio [6]	AS	Intensity Images from Detectors	Morphologi-cal features of Galaxies	Region Segmenta-tion of Ridges & Valleys	Mid-level Description	Rule-based Cognitive System
Chen, Simonett and Sun [18]	RS	Multi-spectral Thematic Mapper	Forest Features such as trees	Nearest Neighbor for Resampling	Clustering Technique for Classification	None
Wu, Suetens and Oosterlinck[5]	MD	ECG EEG X-rays	Chromo-somes	Noise Cleaning, Segment-ation	Statistical Pattern Classifier	Rule-based Verification of Classification
Cleynenbreugel Fierens Suetens and Oosterlinck[26]	MD	X-rays	Blood Vessels	Edge Detector	Knowledge-based System verifying the primitive description obtained in the low level primal sketch	

RS - Remote Sensing; MD - Medical Diagnosis; AS - Astronomy; OG - Oceanography

TIR - Thermal Infra-Red, ECG - Electrocardiogram, EEG - Electroencephalogram

Figure 15.0 Comparison of Techniques for Interpretation

CHAPTER THREE

KNOWLEDGE-BASED LABELING OF OCEANIC FEATURES

3.1 Introduction to the Hybrid Architecture

Feature labeling plays an important role in automated image interpretation. This is very evident from the functional diagram of our system to interpret features in oceanographic satellite images shown in figure 2.1.

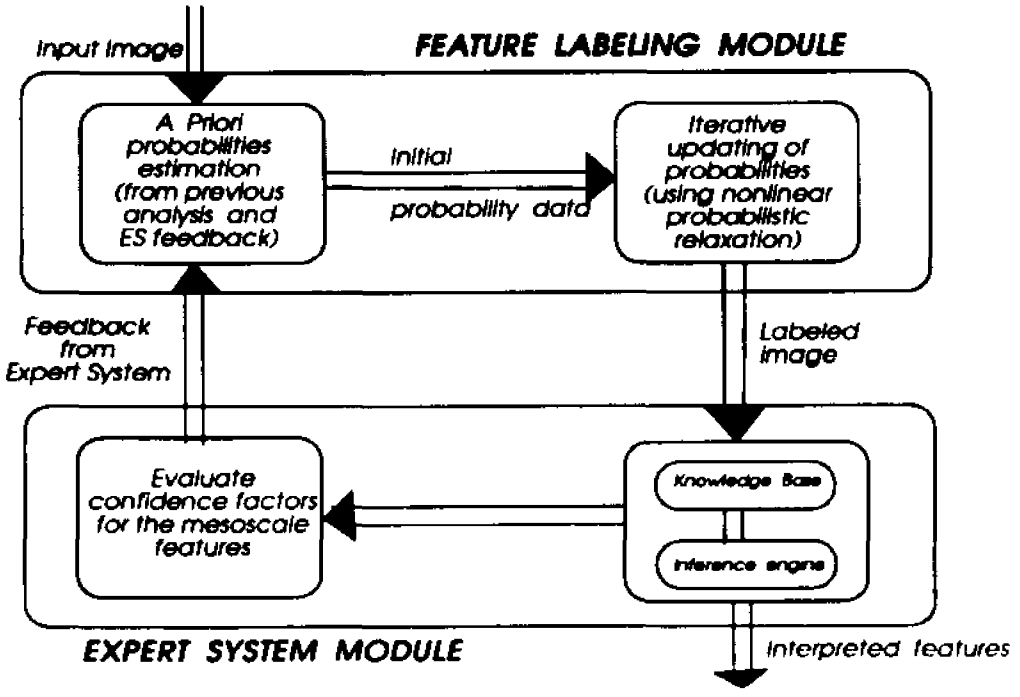


Figure 3.1 Hybrid Architecture for Oceanic Feature Labeling

The module for feature labeling closely interacts with all the other modules in our system. This is well understood from the fact that feature labeling process is inherently domain-dependent. It may also be seen that feature labeling is part of the intermediate level vision tasks. Hence the actual process itself could be a mixture of

conventional image processing techniques and a knowledge-based system. Our feature labeling is one such example. We utilize a hybrid architecture for labeling mesoscale features found in oceanographic satellite images. There are two major components in our architecture: (a) a non linear probabilistic relaxation algorithm and (b) a rule-based expert system. Figure 3.1 depicts the hybrid architecture used for labeling. A salient feature of our architecture is the feedback from the expert system to the labeling module. This greatly improves the performance of labeling, especially when features are obscured by clouds. We describe the architecture of the feature labeling system in the following sections.

3.2 Relaxation Labeling Algorithm

An important research area in image analysis and image interpretation technology is the development of methods that blend contextual information with conventional image processing algorithms. A literature survey clearly indicates that such a hybrid approach yields good results. Relaxation labeling is one such process. Relaxation labeling has been applied to a variety of image processing problems e.g., linear feature enhancement [Zuck77a] , edge enhancement [Scha77a] , image enhancement [Davi78a] , pixel classification [Eklu80a, Davi83a]. A recent survey article by Kittler and Illingworth [Kitt85a] on relaxation labeling highlights the importance of this area of research. The survey [Kitt85a] also points out the advantages and possible applications of relaxation methods. More importantly, the relaxation labeling approach was elegantly described by Rosenfeld, et al. [Rose76a] who investigated the problem of labeling the sides of a triangle and proposed a set of schemes to solve the problem.

The paper [Rose76a] concluded with the result that the nonlinear probabilistic relaxation schemes yield better results than the others. Hancock and Kittler [Hanc90a] have demonstrated that probabilistic relaxation can be successively applied to edge labeling. They also suggest two features that are key to success of such a labeling system. Firstly, a probabilistic framework to represent a world model to ensure internal consistency, and a dictionary of labeling possibilities for the entire context-conveying neighborhood for each object as the second. They [Hanc90a] have shown that the resulting algorithm combines evidence for the presence of edges by drawing on prior knowledge of the permitted label structures.

The goal of the relaxation process is to reduce the uncertainty (and improve the consistency) in the assignment of one of the labels to each object in a set of related objects. In the oceanographic feature classification problem, the classes are the various oceanographic features, namely the north and south walls of the Gulf Stream, cold eddy, warm eddy, shelf front and coastal boundary. Refer to figure 2.1 to identify the positions of these oceanic features in a typical image. The objects are the individual pixels in a set of registered multi-temporal images. The uncertainty could be due to the cloud cover or the overlap of the features not belonging to one of the classes, noise in the image, or other factors. We are attempting to label mesoscale features, but the ocean exhibits variability on all spatial scales. Thermal structure on scales smaller than mesoscale will interfere with the mesoscale feature labeling process. The underlying mathematical framework necessary for the relaxation labeling method is described in the following paragraphs.

Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ be the set of possible labels that may be assigned to each pixel x

in the IR image. Also we let $p_{\lambda}^{k}(x)$ denote the probability that the pixel at x(i,j) belongs to the object λ after k iterations of the relaxation algorithm.

There are two steps in executing the probabilistic relaxation algorithm. In the first step, a priori probabilities are evaluated with the help of ground truth data and / or a previous but recent mesoscale analysis. In the second step, these a priori probabilities are iteratively updated (relaxation) until a consistent labeling is reached. We now discuss these two steps in detail.

Step 1: Estimating a priori probabilities

Let $p_{\lambda}^{0}(x)$ denote the a priori value, that is, the probability that pixel x(i,j) belongs to the object λ at the zeroth iteration. The Bayesian probability equation is used to evaluate this value. The equation (3.2.1) is used to calculate $p_{\lambda}^{0}(x)$.

$$p_{\lambda}^{0}(x) = \frac{p(x \mid \lambda)\ P(\lambda)}{\Sigma_{\lambda}\ p(x \mid \lambda)\ P(\lambda)} \tag{3.2.1}$$

where $p(x \mid \lambda)$ denotes the conditional density function and $P(\lambda)$ the probability of occurrence of the object λ.

To evaluate the conditional density function $p(x \mid \lambda)$, a set of parameters is measured at the pixel x(i,j). Let X denote the parameter vector. The following parameters are used to form the vector X:

(1) vector from origin to pixel x(i,j), both the magnitude and direction.

(2) gray scale intensity value at the pixel x(i,j).

(3) the edge magnitude.

For each object, the mean vector μ_λ and the covariance matrix Σ_λ are computed. Also it is assumed that the conditional density function follows a normal distribution. Hence the conditional density function $p(x \mid \lambda)$ is evaluated using equation (3.2.2).

$$p(x \mid \lambda) = (2\pi \mid \Sigma_\lambda \mid)^{-1/2} \exp\left\{-\frac{1}{2} \quad (X-\mu_\lambda)' \Sigma_\lambda^{-1} (X-\mu_\lambda)\right\} \tag{3.2.2}$$

To compute $P(\lambda)$, relative areas of the objects are considered. The number of pixels in the object λ is n_λ. Then $P(\lambda)$ can be calculated using equation (3.2.3).

$$P(\lambda) = \frac{n_\lambda}{\Sigma_\lambda n_\lambda} \tag{3.2.3}$$

Step 2: Iterative updating algorithm

We now discuss the probability updating rule. The new estimate of the probability of λ at x(i,j) is given by (3.2.4).

$$p_\lambda^{k+1}(x) = \frac{p_\lambda^k(x)\,(1+q_\lambda^k(x))}{\Sigma_\lambda p_\lambda^k(x)\,(1+q_\lambda^k(x))} \tag{3.2.4}$$

where $q_{\lambda}^{k}(x)$ is called the update factor.

The updating factor for the estimate $p_{\lambda}^{k}(x)$ at the kth iteration is given by:

$$q_{\lambda}^{k}(x) = \frac{1}{m}\Sigma_{y}\Sigma_{\lambda'} r_{\lambda\lambda'}(x,y)p_{\lambda'}^{k}(y) \qquad (3.2.5)$$

where m is the number of objects. In this equation, $r_{\lambda\lambda'}(x,y)$ denote compatibility coefficients. These coefficients are computed as in [Davi83a, Rose76a]. According to the relaxation scheme, $r_{\lambda\lambda'}(x,y)$ is a measure of the probabilistic compatibility between label λ on point x and label λ' on point y, and has the following characteristics:

(1) If λ on x frequently co-occurs with λ' on y, then $r_{\lambda\lambda'}(x,y) > 0$, and if they always co-occur, then $r_{\lambda\lambda'}(x,y) = 1$.

(2) If λ on x rarely co-occurs with λ' on y, then $r_{\lambda\lambda'}(x,y) < 0$, and if they never co-occur, then $r_{\lambda\lambda'}(x,y) = -1$.

(3) If λ on x occurs independently of λ' on y, then $r_{\lambda\lambda'}(x,y) = 0$.

3.3 Implementation Details

The system starts up with the human interpretation of an image that is obtained prior to the current image. The initial probabilities are computed based on this previous analysis. Non linear probabilistic relaxation technique is applied to label the features in the current image. Figure 2.1 shows the IR image with all the features. The output of the edge detector algorithm is shown in figure 2.3. The positional details of

the features is then given to the expert system. Sometimes the IR image of the ocean may not be clear. That is, the features are covered by cloud. To analyze such an image, the feedback from the expert system is used. The position of the oceanic features is obtained by activating the inference engine of the the expert system. This approach is different from the conventional one, in which the information is provided by the low level module always. The concept of getting feedback from a high level module (like expert system) is totally new and justified in automatic image interpretation of oceanographic images. The output of the labeling algorithm is given in figure 2.4. The labeled image is then fed to the CEOF interpolation module and the Gulf Stream positions are found. Figure 2.7 shows the output of the interpolation module. Finally the labeled image is fed into the Circle detection module to locate all the eddies. The output of the Hough Circle detector module is shown in figure 2.6. The output of this module, namely the first automated mesoscale product, is shown in figure 2.8.

The automated interpretation system uses a VAX 8300 running VMS OS. The processing and analysis of satellite images are performed using the International Imaging Systems, System 600, a dedicated image processing system. The software modules are written in a combination of C, FORTRAN, and OPS83. As a part of our research, we have conducted several experiments with various image settings. The illustrative example shown in this paper is derived from one of these experiments.

3.4 Future Research in Oceanic Feature Labeling

The feature labeling module that we have implemented at NOARL performs well in the case of a clear and unambiguous image settings. After the software development is frozen, a series of tests was conducted using several image settings. The outcome of the tests indicated two problems with the feature labeling. They are:

- The feature labeling algorithm is biased to the initial analysis. In the case where the previous analysis is not good, the performance of the feature labeling is not satisfactory.
- The feature labeling algorithm behaves in an inconsistent manner with the human interpretation when the image is quite noisy.

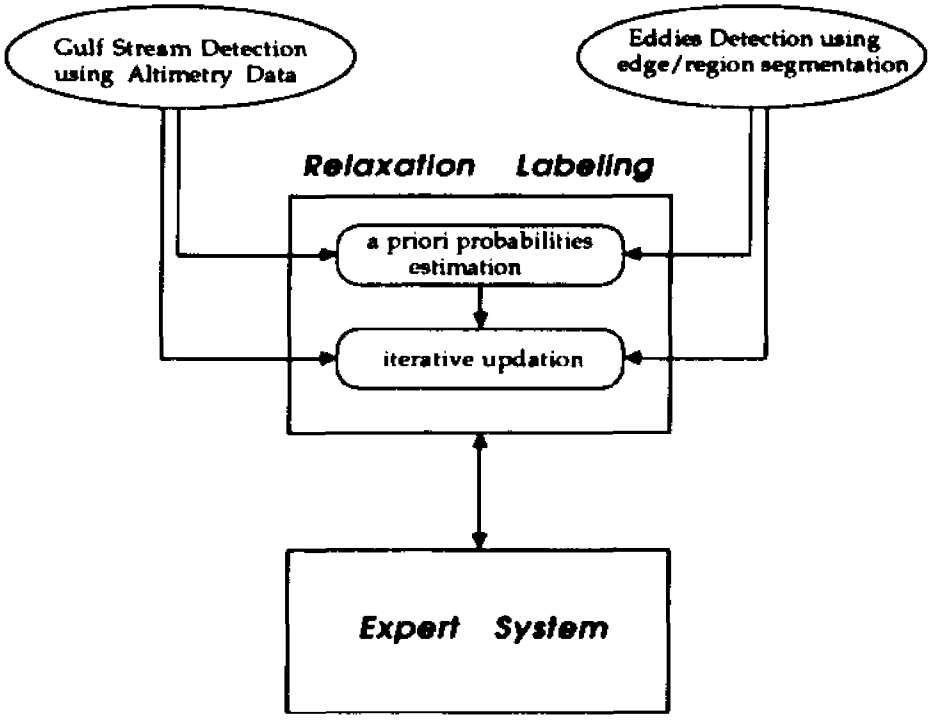


Figure 3.2 Future Research for Oceanic Feature Labeling

Our aim is to address these two fairly difficult problems in the coming years and come up with solutions to the problems stated earlier. In this section we propose several

methodologies to carry out this task. The reminder of this section is devoted to describe some of them. Figure 3.2 illustrates the proposed task.

3.4.1 Altimetry Data Fusion

A satellite altimeter measures the range from the satellite to the sea surface based on travel time of a radar pulse. An independent range measurement (relative to a reference ellipsoid) is derived from satellite tracking. The difference between these two range measurements gives the instantaneous height of the sea surface relative to the reference ellipsoid. Cantrell and Holyer [Cant90a] have recently proposed a new approach to determining the location of Gulf Stream in altimeter profiles. Automation of the Gulf Stream location problem is also accomplished by using the linear Hough transform as a detection and location paradigm. The results obtained from this technique can be fused with our feature labeling algorithm in order to label the North and South walls of the Gulf Stream with less ambiguity. There are two ways of doing this fusion:(1) the a priori probabilities of the walls can be evaluated using the output of the altimeter profiles or (2) the output of the algorithm given in [Cant90a] can be used as one of the parameters to obtain the compatibility coefficients which eventually changes the probabilities of the Gulf Stream pixels.

Any improvement in a typical relaxation algorithm-based system can be attempted using one of the two ways: *try to get a good initial guess* or *make use of additional information* to obtain good compatibility coefficients that are used in the relaxation updating rules. We intend to conduct experiments to take up both the approaches. Based on the performance of the labeling, we may choose the appropriate

method. Presently the initial probabilities are given more weight than the compatibility coefficients. This is one of the reasons why the performance depends on the initial guess. We propose to relax this bias and study the results.

In the first approach, we simply add the altimeter data as one of the feature vectors and study the performance of the labeling algorithm. Since the altimeter data contains a lot of information about the relative position of the features, it is expected to contribute in the labeling process. In the second approach, we make use of the sea depth as one of the parameters and evaluate the compatibility coefficients accordingly for the iterative updating rule.

3.4.2 Feature Labeling and Edge/Region Segmentation

The process of segmentation of a digital image has been executed in one of two fundamentally different ways: edge based and region based. In edge based segmentation the image is scanned for intensity gradients which represent borders between image features, while in region based segmentation features are detected by intensity uniformity between contiguous pixels. Recently Cambridge [Camb90a] proposed a new approach that uses both region and edge detection techniques in a mutually integrated way. This scheme seems to be effective in detecting the pieces of edges that are part of eddies in the Oceanographic Satellite Image. The results of this technique can be integrated with our feature labeling technique by using one of the two ways that are mentioned in the previous section. We propose that the three different techniques, namely, feature labeling, Gulf Stream detection and Eddies labeling can function in a co-operative manner to solve the difficult problem of "labeling" features in

Oceanographic images.

The second problem, namely, the labeling is not satisfactory when the image is noisy, can be tackled with the help of expert system. We understand that the expert system is functioning satisfactorily in many of the cases. Hence with projected positional information on the features, the labeling algorithm can do a much better job in labeling such noisy scenes.

3.4.3 Bathymetry Data

We also propose to fuse the bathymetry data to label the continental shelf fronts. This is basically a smooth curve in the infrared image which runs closer to the coastal boundary. We have already done some preliminary experiments with the bathymetry data in labeling the fronts. We intend to look into this aspect more in detail, because this could be a useful information to the labeling algorithm in terms of giving more reliable positional information about the mesoscale features.

PART II
PARALLEL IMAGE PROCESSING

There has been growing interest in developing highly parallel algorithms for problems in image processing and computer vision, specifically in low and intermediate levels. A new technique for parallelizing vision tasks is introduced. The technique utilizes schemes to embed the inherent data or computational structure, used to solve the problem, into parallel architectures such as hypercubes. The important characteristic of the technique is that the adjacent pixels in the image are mapped to nodes that are at a constant distance in the hypercube. Using the technique, parallel algorithms for neighbor-finding and digital distances are developed. A parallel hypercube sorting algorithm is obtained as an illustration of the technique.

CHAPTER FOUR

NEW TECHNIQUE FOR IP PARALLELIZATION

4.1 Introduction

We introduced the image processing parallelization paradigm in section 1.3. As mentioned in 1.3, there are two types of parallelism found in image processing applications: *data parallelism* and *functional parallelism*. In brief, data parallelism exploits the features of the *data structure* while functional parallelism makes use of the *inherent* parallelism in the *algorithm* itself. Our research mainly is focussed on developing parallel algorithms using the first type of parallelism, namely, the data parallelism. In this chapter we introduce our technique for parallelization in general. In the following chapters we present the application of our technique to various problems in image processing.

We first provide the necessary background in the area of parallel processing, and parallel architectures based on interconnection networks, in particular. In section 4.2, we introduce the well-known *mapping paradigm*, and discuss the various issues related to mapping. Our new generalized technique for parallelization is explained in section 4.3. There has been growing interest in applying the techniques developed in the area of *Computational Geometry* (CG) to some of the problems in pattern recognition and image processing [Tous86a, Akl89a]. It has been shown that most of the problems in CG can be transformed to the well-known *sorting* paradigm [Prep85a]. Hence developing good parallel sorting algorithms would pave the way for efficient

solutions to problems in image processing. Towards this objective, we have applied our new parallelization technique to the sorting problem, and obtained an efficient parallel algorithm for sorting. The results of our sorting technique are given in section 4.4.

4.2 Parallel Processing Overview

In the last three decades of computing, we have witnessed truly impressive advances in speed, size and sophistication of computers. The *speed* of the computers is of primary importance since many real-life applications are computationally intensive. The necessary step in searching for methods for improvement in computation speed is to allow parallelism or concurrency. *Parallel processing* is defined as an efficient form of information processing which emphasizes the exploitation of concurrent events in the computing process. Parallel computers are those that emphasize parallel processing. Parallel computer architectures may be divided into three configurations:

- Pipeline computers
- Array processors
- Multiprocessor systems

The primary sources of references for the following material are Hwang and Briggs [Hwan85a], and [Quin88a].

A pipeline computer performs overlapped computations to exploit temporal parallelism. In a typical digital computer system there are four major steps: *instruction*

fetch (IF) from the main memory; *instruction decoding* (ID), identifying the operation to be performed; *operand fetch* (OF), if needed in the execution; and then *execution* (EX) of the decoded arithmetic logic operation. In a pipelined computer, these four steps may be overlapped for successive instructions. This is shown diagrammatically in figure 4.1.

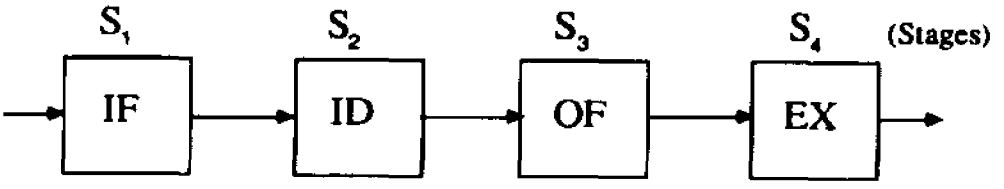


Figure 4.1 A pipelined processor

Due to the overlapped instruction and arithmetic execution, it is obvious that pipeline machines are better tuned to perform the same operations repeatedly through the pipeline. Control Data's Star-100 and Texas Instrument's Advanced Scientific Computer (ASC) are examples of pipelined (vector processors) computers.

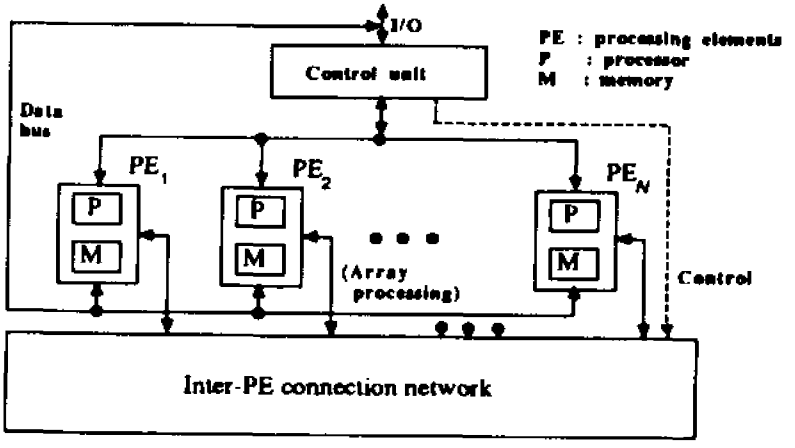


Figure 4.2 SIMD array processor organization

An array processor (also called SIMD-array processor) is a synchronous parallel computer with multiple arithmetic logic units (ALUs), called processing elements (PE), that can operate in parallel in a lock-step fashion. Spatial parallelism is achieved

by duplicating the ALU's. The PEs perform the same function at the same time. The PEs communicate each other with the help of a data-routing mechanism. Figure 4.2 depicts the structure of a typical array processor.

Each PE consists of an ALU with registers and local memory. The interconnection pattern to be established for solving a problem is under the control of the control unit (CU). The PEs in an array processor are arranged in the form of an interconnection network. Different array processors may use different interconnection networks among the PEs. Most commonly used interconnection structures are meshes, star, hypercube, and trees. In this dissertation, we consider mesh-connected and hypercube-connected array processors.

Multiprocessor systems (also called MIMD systems) are designed to have improved throughput, reliability, flexibility, and availability. Figure 4.3 shows the organization of a basic multiprocessor system.

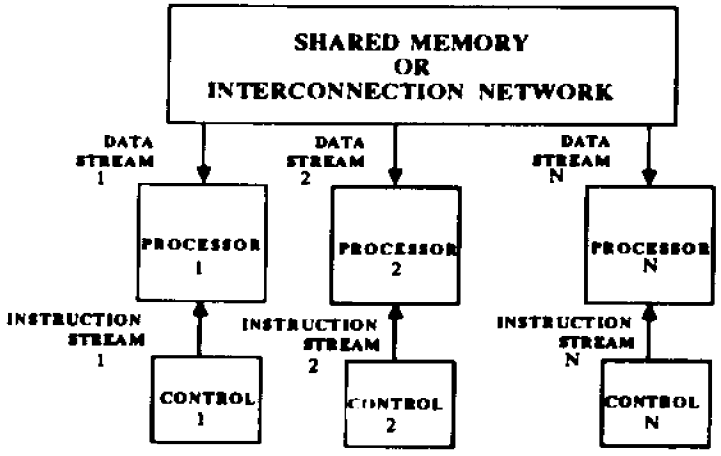


Figure 4.3 MIMD processor organization

The system contains two or more processors of comparable size. All processors share access to common sets of memory modules, I/O channels, and peripheral devices. Most importantly, the entire system must be controlled by a single integrated

operating system. Each processor has its own local memory and private devices. Interprocessor communications take place through shared memories or an interrupt network. Since multiprocessors are not considered in our research, we will not provide the details of these systems.

We now outline three important processor organizations - methods of connecting processors in a parallel computer.

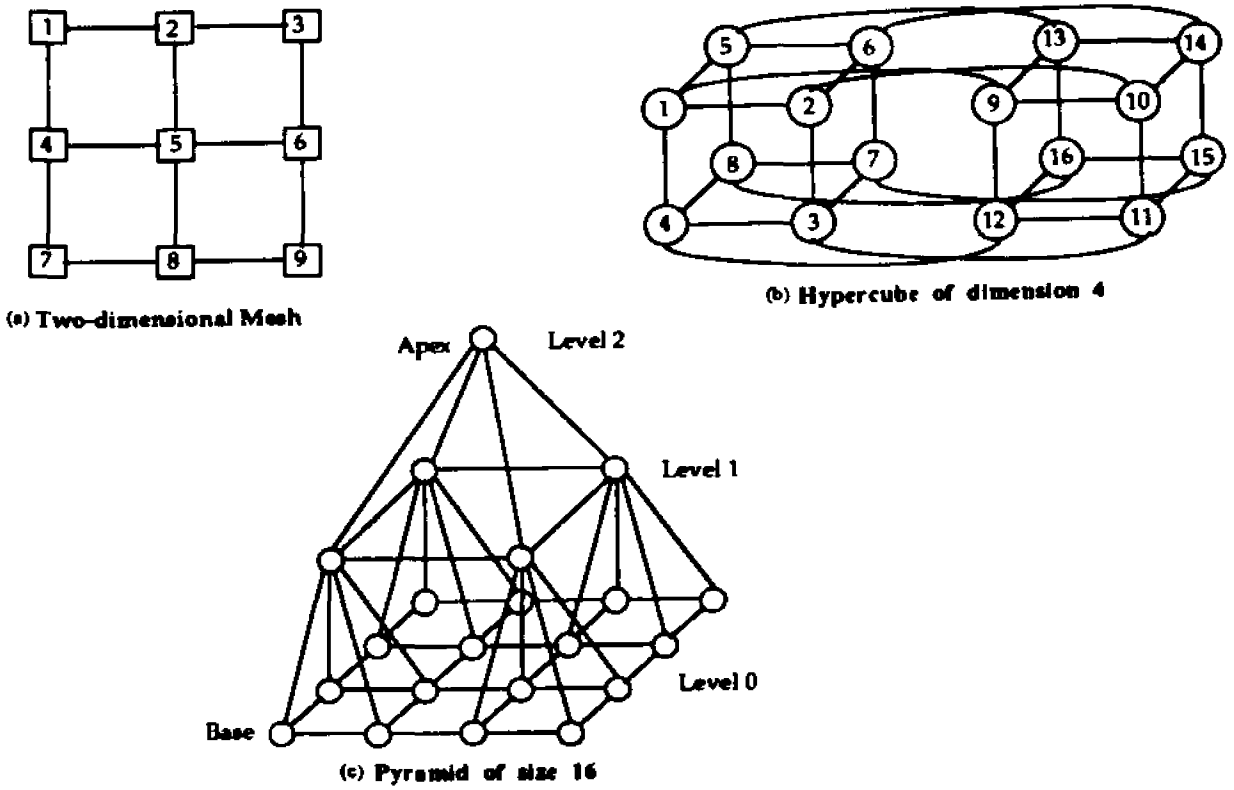


Figure 4.4 Three interconnection networks of processors

Mesh

In a mesh network, the nodes are arranged into a q-dimensional lattice. Communication is allowed only between neighboring nodes; hence interior nodes communicate with $2q$ other processors. Figure 4.4(a) illustrates a two-dimensional mesh. Mesh networks do have the disadvantage from a theoretical point of view that data routing requirements often prevent the development of $O(log^k n)$ parallel algorithms. For

example, sorting n elements on a two-dimensional mesh requires time $\Omega(\sqrt{n})$.

Pyramid

A pyramid network of size p is a complete 4-ary rooted tree of height $\log_4 p$ augmented with additional interprocessor links so that the processors in every tree level form a two- dimensional mesh network. It may be easily derived that the number of processors i a pyramid of size p is $\frac{4}{3}p - \frac{1}{3}$. The levels of the pyramid are numbered in ascending order. Figure 4.4(b) shows a pyramid of size 16. Pyramid-based parallel architectures are used in multi-resolution image processing applications, wherein, each level of the pyramid processes a specific resolution of the digital image.

Hypercube

In a hypercube of dimension d, there are 2^d processors. Assume that these are labeled $0,1,...,2^d - 1$. Two processors i and j are directly connected if and only if the binary representations of i and j differ in exactly one bit. Figure 4.4(c) illustrates a hypercube of dimension 4. Following are some of the reasons why hypercube-connected computers are more popular than others.

- The maximum distance between any two processors in a hypercube is d. While meshes use a smaller number of connections per processor, the maximum distance between the processor is larger.
- Most popular networks are easily simulated in a hypercube. Stated other words, algorithms developed for computers based on various other interconnections may be easily mapped into hypercubes.

- A hypercube is completely symmetric. Moreover, the recursive definition of hypercubes allow us to decompose a hypercube into smaller sub-hypercubes. This property makes it relatively easy to implement recursive divide-and-conquer algorithms on the hypercube.

The design of parallel algorithms for computers based on the above mentioned interconnections is an important issue in parallel processing. Since the design and performance of a parallel algorithm depends very much on the underlying representation of a computation in the form of a graph related structures, it is necessary to keep the data structured network configurations in mind when designing parallel algorithms. For details refer to [Moit87a]. The design of a parallel algorithm involves choosing appropriate data structures, allocating processors and finally the algorithm to do the computation. It is therefore essential that every parallel algorithm provides a clear specification of the architecture used, the data structure chosen and the mode of synchronization and communication.

4.3 Mapping Paradigm

Many of the parallel computer architectures based on array of processors are not fully-connected, that is a direct link does not connect each pair of processors. There are two reasons for this:

- the total number of links in fully-connected systems increases as the square of the number of processors, which is not acceptable in many cases.
- the number of I/O ports on each of the processors increases linearly with the number of processors, not possible because the number of I/O ports is generally

fixed at some constant value.

Hence it is a common practice to design architectures where the interconnection of processors follow some regular pattern, viz., mesh, hypercube, and pyramid. Developing parallel algorithms for these interconnection networks is of prime importance.

The design of a parallel algorithm involves the following two tasks [Moit87a]:

- Divide the problem into smaller subproblems
- Solve these subproblems on parallel machines.

This is achieved by creating a process for each of these subproblems, and executing these processes on the processors of the parallel machine. The task of decomposing a problem into smaller ones, without data dependency among them, is not easy. So processes need to communicate among themselves to solve the problem. The second task involves assigning processes to processors, synchronizing the processes, and controlling the communication among the processes. When assigning processes to processors, pairs of processes that communicate with each other should be placed, as far as possible, on processors that are directly connected. We call the assignment of processes to processors a *mapping* and the problem of maximizing the number of pairs of communicating processes that fall on pairs of directly connected processors the *mapping paradigm.*

An area that is very closely related to the mapping paradigm is the *graph embedding*. The field of graph embedding has emerged as one of the active area in theoretical computer science. Refer to [Rose80a] for a more details on the techniques for graph embedding. In fact, the theoretical framework for our parallelization technique

is based on *embedding* strategies. We now provide a few definition that are used in most of the embedding techniques.

The structure of computations in an algorithm is modeled by a graph, called the *guest graph*. This graph depicts the required interaction between the data elements of the computation. The topology of an interconnection network is depicted by a graph, called the *host graph*. The guest graph is embedded in the host graph for execution. The quality of an embedding is often measured by two parameters: (a) dilation and (b) expansion. The expansion is a measure of processor utilization where as the dilation affects the communication cost.

The embedding function f maps each node in the guest graph $G=(V_G, E_G)$ into a *unique* node in the host graph $H=(V_H, E_H)$. V_G and V_H denote the node sets of the guest graph and the host graph respectively, and E_G and E_H the edge sets. Let v_1 and v_2 be nodes in G. Since the embedding is a 1-1 mapping of V_G onto V_H, let $f(v_1)$ and $f(v_2)$ be the nodes in H that are images of v_1 and v_2 respectively. Let the distance, $d(v_i, v_j)$ between v_i and v_j be defined as the number of edges in the shortest path connecting v_i and v_j in either G or H.

Definition 4.1: The dilation d_f is defined as

$$d_f = \max \frac{hd(f(v_1), f(v_2))}{d(v_1, v_2)}$$

Definition 4.2: The expansion e_f is defined as

$$e_f = \frac{|V_H|}{|V_G|}$$

We now proceed to the transformation technique as given in section 4.4.

4.4 Transformation of PRAM-Algorithms

Parallel Random Access Memory (PRAM) is a popular theoretical model used for parallel computation. The PRAM model consists of several independent sequential processors, each with its own private memory, communicating with each other through a shared (global) memory. During the execution of a parallel algorithm all the processors gain access to the shared memory for reading input data, for reading or writing intermediate results, and for writing final results. Depending on whether or not two or more processors are allowed to gain access to the same memory location simultaneously, we can distinguish four variants of PRAM given as follows [Akl89a]:

- *Exclusive-Read Exclusive- Write* (EREW). Access to memory locations is exclusive. In other words, no two processors are allowed simultaneously to read from or write into the same memory location.
- *Concurrent-Read Exclusive-Write* (CREW). Multiple processors are allowed to read from the same memory location but the right to write is sill exclusive.
- *Exclusive_Read Concurrent-Write* (ERCW). Multiple processors are allowed to write into the same memory location but read accesses remain exclusive.
- *Concurrent-Read Concurrent_Write* (CRCW). Both multiple-read and multiple-write privileges are granted.

Karp and Ramachandran [Karp88a] provide a complete details on PRAMs.

The time complexity of a sequential algorithm is decided by the amount of com-

putation needed. But the time complexity in a typical parallel algorithm consists of two parts: *ComPutational Complexity (CPC)* and *CoMmunication Complexity (CMC)*. The CPC is the time required to solve the subproblems, whereas the CMC is the time required for interprocess communication. For those instances where the time complexity is dominated by CPC, theoretical models have been developed which emphasize mainly on computation. PRAM is one such model. The design of algorithms for *Standard Interconnection Networks* (SIN) should also consider CMC. Efficient processor allocation, data assignment, and data routing techniques are used to minimize CMC. The minimization of CMC may be achieved by designing SINs that have dedicated links for processors to communicate with each other. However, it is neither practical to come up with different SINs for every new problem nor to provide links between every two processors. SINs are developed to provide most of the desirable features needed for many applications. It may be necessary to develop different algorithms for the same application on various SINs. The inability of providing dedicated networks for each problem, and the wide spread availability of general-purpose SINs motivate us to design parallel algorithms for these SINs.

There is no universal method for designing parallel algorithms [Moit87a]. There are several efficient parallel algorithms available for PRAMs [Moit87a, Karp88a]. It is difficult to develop algorithms for each problem and for each interconnection network. So it is desirable to systematically derive parallel algorithms for these networks from their PRAM-versions. We investigate techniques that systematically transforms algorithms designed for PRAMs to those for SINs. PRAMs use shared memory and in these models the cost of accessing memory is assumed to be constant. However in

case of SINs the communication is based on message-passing between processors. The cost of data transfer between any two arbitrary processors in a network is not a *constant*, but a function of the size of the network. For example, in case of a hypercube network, this is a *logarithmic* function of the *number of processors*. It will be interesting to study the existence of transformations of PRAM algorithms to those of SINs that would keep the cost of data transfer, a constant. The resulting parallel algorithm would maintain the same time complexity on SINs, unlike a naive implementation that would increase the cost due to data communication. We present our transformation technique in the following section.

4.4.1 Description of our technique

Parallel algorithms for SINs are traditionally designed by mapping the underlying PGs developed for PRAMs, into SINs. Many of these mapping schemes consider only the pattern of computation and not that of communication. On the other hand, in our technique [Hegd91a], we extract the hidden pattern of communication in the PG and augment it with the edges between the processes which need to communicate. The resulting augmented process graph (APG) is then used for mapping. A good mapping will embed an APG into SIN with minimal dilation, preferably, a constant one. We propose to study the algorithms developed for PRAMs, derive a corresponding APG, and investigate if a constant dilation mapping exists for APG into the given interconnection network. If one such mapping exists, the complexity of the resulting algorithm will not be increased from that of the PRAM algorithm because of the communication needs.

As mentioned in section 4.3, the given problem is divided into smaller processes. These processes have inter-dependencies either due to data or computation. The processes and their inter-dependencies may be modeled as a graph structure called *Process Graph* (PG) which is formally defined as follows.

Definition 4.3: A process graph, $PG(V,E)$ is a graph where V is a set of processes, and E is the set of edges representing the relationship (either data or computation) among these processes.

Note that our definition of process graph is very similar to that of problem graph defined in [Bokh81a, Lee87a, Pint87a].

Definition 4.4: An $APG(V,E)$ is a graph where V is a set of processes and is the same as that in PG, and E is the set of edges defined as $E = E' \cup E''$. E' is the set of edges in the process graph and E'', called *augmented edge set*, is the set of edges defined over the processes which need to communicate, and not adjacent in the PG.

Definition 4.5: A *Communication Edge* in an $APG(V,E)$, $e(v_1, v_2)$, is an edge connecting the processes v_1 and v_2, which communicate, where $v_1, v_2 \in V$ and $e \in E$.

We describe the three major steps in our algorithm. The technique is also shown in the form of a flow chart given in figure 4.5.

- **Identify the process graph**

In this stage, we identify the structure of the Process Graph by understanding the inherent data or computation partitioning, and the inter-dependencies among the sub-problems. Usually, the underlying data structure in the PRAM algorithm reveals the structure of the PG. For example, the algorithms that make use of divide-and-conquer

strategy may have an m-ary tree as the underlying structure for data and/or computation.

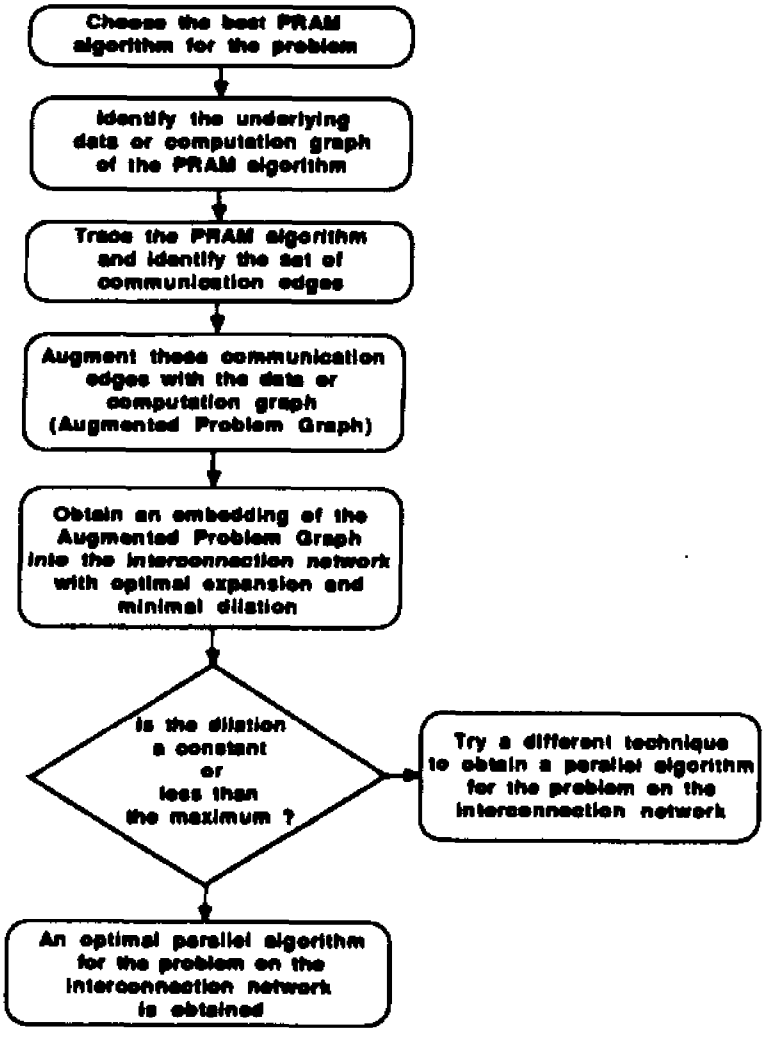


Figure 4.5 Transformation of a PRAM-algorithm

- **Derive the augmented process graph**

The PG as obtained from step 1, is augmented with *communication edges* to derive the *Augmented Process Graph* (APG). When processes of a PG are mapped onto processors of a SIN, care taken to place the processes which need to communicate in processors nearby, will reduce the communication overhead. Toward this objective, *communication edges* are introduced whenever two processes need to

communicate with each other during the execution of the PRAM algorithm.

- **Embed the augmented process graph into the network**

In this step, we investigate techniques for embedding the *Augmented Process Graph* (APG) as obtained from step 2, into the Standard Interconnection Network (SIN). We design an algorithm that embeds the APG into the given SIN with minimal dilation and optimal expansion. Our objective is to find an algorithm that results in constant dilation and optimum expansion. If successful, we would have an optimal parallel algorithm for the problem on the given network.

In summary, we have presented a methodology to transform an algorithm developed for PRAM into that for the given interconnection network. We next illustrate our technique by giving a parallel sorting algorithm for hypercubes in section 5.4. It is interesting to note that our illustrative example algorithm may be applied to a variety of problems which use divide-and-conquer strategy.

4.5 Hypercube Sorting : An Illustration

Several parallel sorting algorithms exist for hypercube network [Hirs78a, Dey90a, Vali75a, Nass81a]. The most commonly used algorithm is based on implementation of bitonic sorting [Batc68a, Quin88a]. Recently Bilard and Nicolau [Bila89a] have developed optimal parallel algorithms for PRAMs using adaptive bitonic sorting. The latter algorithm [Bila89a] reduces the number of comparisons necessary for sorting, by avoiding the redundant comparisons in Batcher's [Batc68a] algorithm. We choose the PRAM algorithm given by Bilardi and Nicolau to illustrate our transformation technique. We use clever embedding schemes to avoid the

logarithmic increase in the communication cost. In this section, we present a new sorting algorithm [Hegd91a] for hypercubes by transforming a bitonic sorting algorithm for PRAM. The proposed algorithm has a time complexity of $O(log^2 N)$, TP product of $O(N \log^2 N)$, and uses less than $2Nlog\ N$ comparisons. The reminder of this section is organized as follows. In section 4.5.1, we introduce the notations and definition needed to develop the sorting algorithm. We then obtaining the sorting algorithm in section 4.5.2, using the three steps outlined in section 4.4.1. We finally state and prove a theorem that establishes the required time and processor complexities in section 4.5.3.

4.5.1 Definitions and Notations

Bitonic sorting is a parallel algorithm based on a merge-sort scheme using bitonic merging [Batc68a].

Definition 4.6: A *bitonic sequence* is a sequence of numbers $a_0, \ldots, a_{n-1}$ with the property that (1) there exists an index i, $0 \le i \le n-1$, such that a_0 through a_i is monotonically increasing and a_i through a_{n-1} is monotonically decreasing or else (2) there exists a cyclic shift of indices so that the first condition is satisfied.

The bitonic merge algorithm transforms a bitonic sequence into a sorted list. The unsorted $n = 2^k$ elements, can be viewed as n sequences of length 1 which is to be merged. Successive merging of these sequences is carried out until the sorted list is obtained. The time complexity of the bitonic sorting algorithm is $\Theta(log^2 n)$. The details of the bitonic sorting algorithm are given in [Batc68a].

The number of comparisons and the number of exchanges are of $O(N \log N)$ in bitonic merging. Bilardi and Nicolau [Bila89a] have given an *adaptive* bitonic merging technique, wherein the number of comparisons as well as the number of exchanges are of $O(N)$. This reduction is achieved using the following two observations:

- A subset of less than $2N$ of the comparisons performed by Batcher's algorithm is sufficient.
- There is a regular pattern of data exchanges in the Batcher's network.

Based on the second observation, Bilardi and Nicolau use a data structure called *bitonic tree* in their algorithm. This data structure allows them to accomplish many data exchanges by a small number of subtree exchanges. Further details on this algorithm may be found in [Bila89a].

Definition 4.7: A *bitonic tree* [Bila89a] may be defined as a binary tree where each node contains an element from a totally ordered set, and the sequence of elements encountered in the in-order traversal of the tree is bitonic.

We refer to *bitonic tree* of height k as $BT(k)$. We show $BT(3)$ in figure 4.6(a).

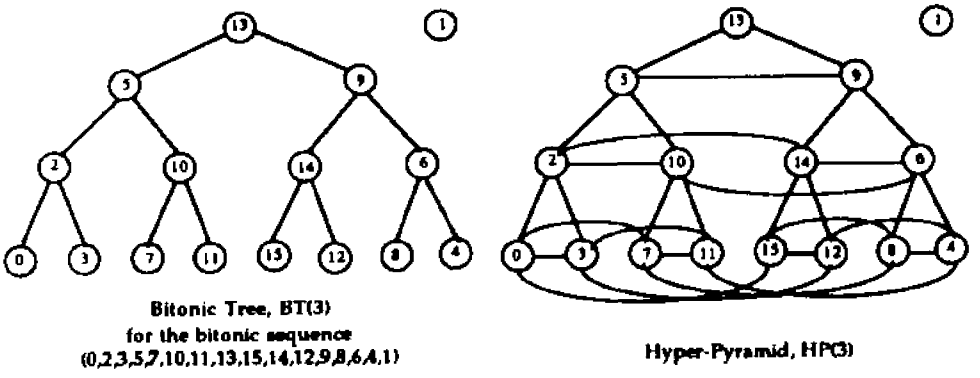


Figure 4.6 Bitonic tree and its corresponding Hyper-Pyramid

We denote a n-dimensional Boolean cube by $H(n)$. Ho and Johnsson [Ho89a] have given a dilation d embedding of a *hyper-pyramid* into a hypercube. A *hyper-pyramid* contains pyramids as proper subgraphs.

A hyper-pyramid may be formally defined as follows [Ho89a] :

Definition 4.8: A $P(k, d)$ *hyper-pyramid* is a level structure of k boolean cubes where the cube at level i is of dimension id, and a node at level $i-1$ connects to every node in d dimensional boolean subcubes at level i, except for the leaf level k.

Ho and Johnsson show that a $P(k, d)$ can be embedded in a boolean cube with minimal expansion and dilation $\max(d,2)$. We refer to $P(k, 1)$ as $HP(k)$ and it will be used throughout our discussion. $HP(3)$ is given in figure 4.6(b).

4.5.2 Transformation of Adaptive Bitonic Sorting Algorithm

We first recall our transformation technique. We derive an APG by augmenting the PG with communication edges and then maps this APG into SINs. A good mapping will embed an APG into SIN with minimal dilation, preferably, a constant one. We propose to study the algorithms developed for PRAMs, derive a corresponding APG, and investigate if a constant dilation mapping exists for APG into the given interconnection network. If one such mapping exists, the complexity of the resulting algorithm will not be increased from that of the PRAM algorithm because of the communication needs.

Since our illustration follows the algorithm given by Bilard and Nicolau [Bila89a] closely, we reproduce their algorithm in figure 4.11, with necessary modifications to run on hypercube. We refer to this algorithm as MABS. Note that

there are two recursive procedures in MABS : *bisort* and *bimerge*. More on how the algorithm is modified is given in section 4.5.3. We use hypercubes as the target interconnection network. Three stages of our technique are given below.

Stage 1: *Identify Process Graph*

In this stage, we identify the structure of the Process Graph by understanding the inherent data or computation partitioning, and the inter-dependencies among the subproblems. Usually, the underlying data structure in the PRAM algorithm reveals the structure of the PG. In the MABS, the PG used is clearly a bitonic tree (BT). The vertices of the BT contain the elements to be sorted, while the edges of the BT reflect the organization of the data.

Stage 2: *Derive Augmented Process Graph*

The PG as obtained from stage 1, is augmented with *communication edges* to derive the *Augmented Process Graph* (APG). When processes of a PG are mapped onto processors of a SIN, care taken to place the processes which need to communicate in processors nearby, will reduce the communication overhead. Toward this objective, *communication edges* are introduced whenever two processes need to communicate with each other during the execution of the PRAM algorithm.

MABS uses bitonic tree as the Process Graph. *Communication edges* are added between any two vertices of the bitonic tree, if there is a need for either data comparison or exchange. The resulting graph is called the *Augmented Bitonic Graph* (ABG), which is formally defined as follows.

Definition 4.9: An *Augmented Bitonic Graph*, $ABG(V,E)$, where V is the set of vertices of the bitonic tree (PG of MABS) and $E = E' \cup E''$, where E' is the set of edges of the bitonic tree, and E'' is the set of *communication edges* defined over the bitonic tree.

Definition 4.10: The $ABG(k)$ is an *Augmented Bitonic Graph* obtained by augmenting *Communication edges* with the $BT(k)$ (bitonic tree of height k).

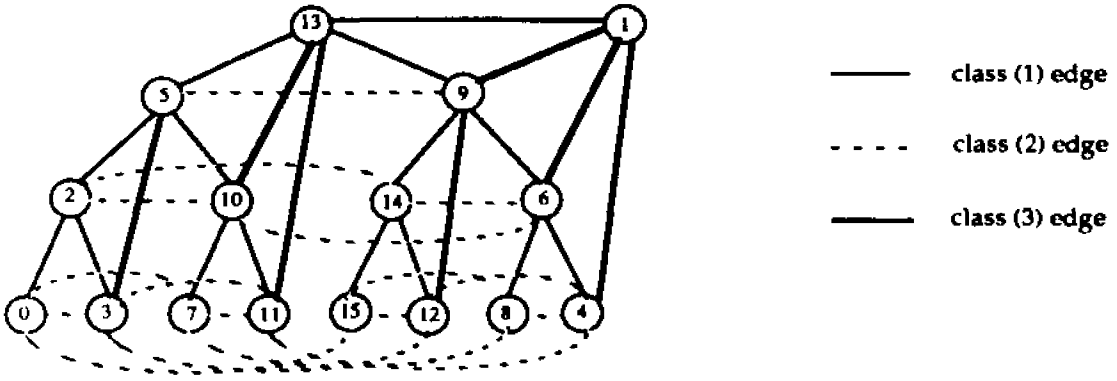


Figure 4.7 Augmented Bitonic Graph, ABG(3)

We characterize the set of edges in the $ABG(k)$ into three classes:

(1) the set of edges in the underlying bitonic tree, $BT(k)$;

(2) the union of the set of edges in the hypercube of dimension i at level i of the $BT(k)$;

(3) the set of edges between the *spare* and the *right child* at each recursive call of the *bisort* procedure. Note that at each recursive call on the sub roots, the recursive call on the *left child* uses the *current root* as the *spare*, and that on the *right child* uses the *current spare* as the *spare*.

This characterization may be used to obtain an *Augmented Bitonic Graph*, $ABG(k)$ from a bitonic tree, $BT(k)$ of arbitrary height, k. The figure 4.7 depicts $ABG(3)$. In this

figure, class (1) edges are shown as thin lines, class (2) as dotted lines, and class (3) as thick lines.

Stage 3: *Embed Augmented Process Graph into SIN*

In this stage, we investigate techniques for embedding the *Augmented Process Graph* (APG) as obtained from stage 2, into the Standard Interconnection Network (SIN). In our illustrative sorting example, the APG is the *Augmented Bitonic Graph* (ABG), and the SIN is a hypercube. We use an embedding technique that is similar to the one used by Ho and Johnsson [Ho89a] for embedding a hyper-pyramid into a hypercube. We now state the following lemma which leads us to the actual embedding algorithm.

Lemma 4.1: The Hyper-Pyramid of height k and degree 1, $HP(k)$ is a subgraph of the *Augmented Bitonic Graph* of height k, $ABG(k)$.

In the $ABG(k)$, we can recognize a *real root* (given as rr_i, $1 \le i \le 2$ for $ABG(m)$ in figure 4.9), and a *spare root* (given as sr_i, $1 \le i \le 2$ for $ABG(m)$ in figure 4.9) as used in [Bila89a]. We provide an algorithm that embeds the underlying bitonic tree of $ABG(k)$ such that the *real root* and the vertices of the *sub tree* rooted at the *left child* of the *real root* and the *spare root* and the vertices of the *sub tree* rooted at the *right child* of the *real root* into two similar disjoint sub cubes of dimension $n-1$, in $H(n)$. Our algorithm recursively embeds the $ABG(k)$ into $H(n)$ and is given in theorem 4.1. We use induction on the height k of the $ABG(k)$ to prove theorem 4.1. Our basis of induction, unlike given in [Ho89a], is given for $k=2$, though our theorem holds good for $k = 0,1$ trivially. The basis is so chosen in order to clarify the embedding of class (3) edges

mentioned in the characterization of $ABG(k)$ as given Stage 2. As stated in [Ho89a] , this embedding technique is similar to those used by Bhatt and Leiserson [Bhat84a] and Bhatt and Ipsen [Bhat85a]. We now state the following theorem 4.1.

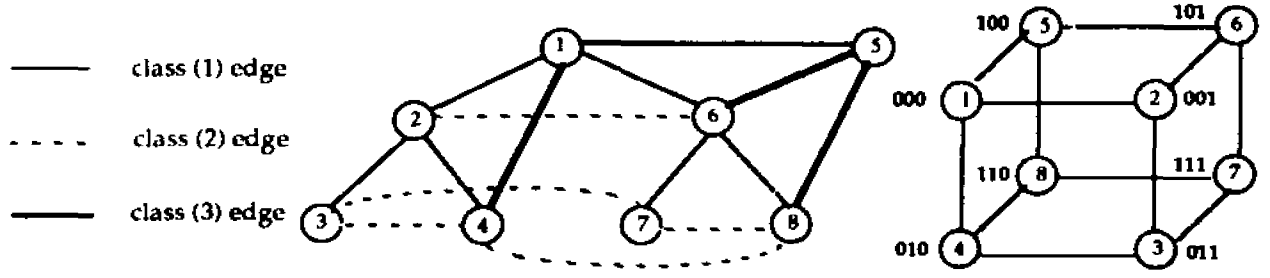


Figure 4.8 Embedding ABG(3) into H(3)

Theorem 4.1: An *Augmented Bitonic Graph*, $ABG(k)$ can be embedded into a hypercube, $H(n)$ with dilation 2.

Proof: Let f_k be the function that maps an $ABG(k)$ into an $H(n)$ with dilation 2. We define f_k by a recursive construction on k, and prove the theorem by induction.

Basis: For k=2 the actual embedding of $ABG(2)$ into $H(3)$ is given in figure 4.8. The theorem 5.1 clearly holds good for our basis as the function f_k is given below for k=2.

$f_2(1) = 000$; $f_2(2) = 001$; $f_2(3) = 011$; $f_2(4) = 010$; $f_2(5) = 100$; $f_2(6) = 101$; $f_2(7) = 111$; $f_2(8) = 110$;

where, $f_2(i)$, $1 \le i \le 8$ is the mapping function applied on the vertex i of the $ABG(2)$, and the right-hand-side of the above relations refer to the binary addresses of the nodes in $H(3)$.

Hypothesis: For $k \le m$, an $ABG(k)$ can be embedded into an $H(n)$ with dilation two, and the *real root* and the *spare root* are mapped to adjacent cube nodes.

Induction Step: Assume that there exists an embedding function f_m which satisfies the induction hypothesis. In order to embed an $ABG(m+1)$ into $H(n+1)$, we make the following observations. The remainder of the proof uses the figure 4.9.

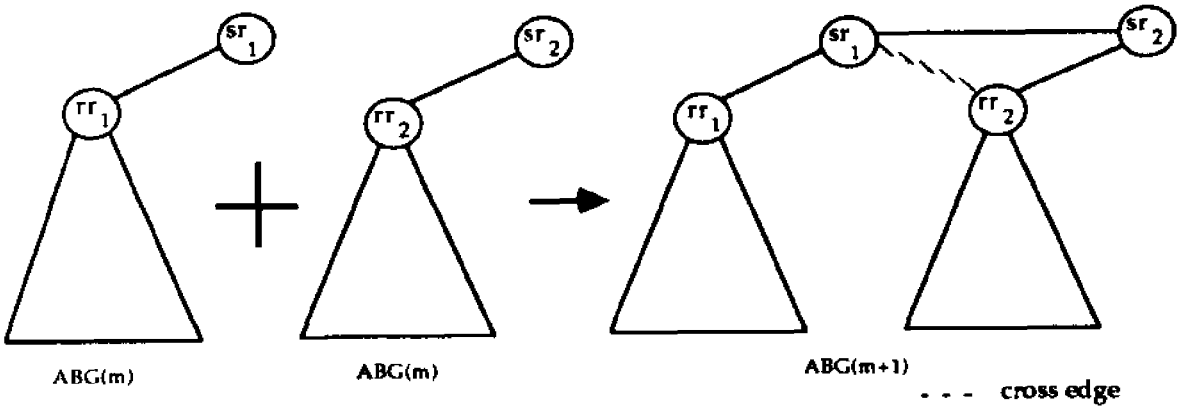


Figure 4.9 Induction Step for Theorem 4.1

Observation 1: An *Augmented Bitonic Graph*, $ABG(m+1)$ may be obtained from two identical copies of $ABG(m)$. Let rr_1 and sr_1 be the *real root* and the *spare root* of the first copy, while rr_2 and sr_2 refer to those of the second copy. In $ABG(m+1)$, sr_1 is the *real root*, and sr_2, the *spare root*. Note that rr_1 and rr_2 become the *left child* and *right child* of the $ABG(m+1)$ respectively.

Observation 2: A Hypercube, $H(n+1)$ may be obtained from two identical copies of $H(n)$. It follows from the recursive definition of hypercubes, that these two copies refer to the maximal sub cubes, and we refer to any two adjacent nodes that belong to different copies as similar nodes.

Claim: It is sufficient to show that the edge between sr_1 and rr_2 of $ABG(m+1)$ is dilated to 2, to prove the induction step.

Proof: Let us represent a vertex in $ABG(m+1)$ using three parameters: (1) copy number, c (1 or 2), (2) level of the vertex in the underlying bitonic tree of $ABG(m)$, l

$(0 \le l \le m-1)$, and (3) the index of the vertex in that level, i. For example, the vertex $v(1,l,i)$ is in the first copy, at level l, and its index is i. We refer to an edge, $e\ (u(c_1,l_1,i_1),v(c_2,l_2,i_2))$ as a *similar edge* where $c_1 \neq c_2$, $l_1 = l_2$, and $i_1 = i_2$. Two $ABG(m)$'s of $ABG(m+1)$ can be embedded into $H(n)$ using the induction hypothesis and the Observation 1. The remaining edges of $ABG(m+1)$ that need to be embedded in $H(n+1)$ fall into the following two types.

Type I: The set of similar edges of $ABG(m+1)$.

Type II: The edge between sr_1 and rr_2. We refer to this edge as *cross edge*.

In our embedding algorithm, the *Type I* edges are dilated to 1 since the two end vertices of these edges are mapped onto two adjacent nodes in the hypercube, $H(n)$. This follows from Observation 2. Now we are left with only the *Type II* edge and hence the claim. We now consider the *Type II* edge. Since sr_1 and sr_2 are connected by a *similar edge* (dilated to 1), and sr_2 and rr_2 are adjacent (from hypothesis) the *Type II* edge is dilated to 2. Hence the theorem.■

4.5.3 Sorting on Hypercube

In this section, we formally present a new sorting algorithm on hypercube. The idea is to first embed the $ABG(k)$ into a hypercube, $H(n)$ with minimal expansion and dilation of 2 using Theorem 4.1, and execute the modified adaptive bitonic sorting algorithm, MABS given in figure 4.11, using the schedule for the parallel version given in [Bila89a]. The modifications which are necessary to execute these procedures on hypercubes are provided as comments in MABS. The actual implementation uses *message-passing* instead of using *shared-memory*. We assume that data

exchange which is done by swapping the values in [Bila89a] is achieved by using *send* and *receive* communication primitives. The ability of the *controller* in the hypercube network to exchange values between a selected sub-set of nodes, using the masking facility, is used to simulate *sub-tree pointer-exchanges* used in [Bila89a].

Algorithm BSH ;

/* Input: A sequence of N elements from a totally ordered set */
/* one element being stored in a node in the Hypercube */
/* Output: The list of sorted numbers. */

begin
1. Construct an Augmented Bitonic Graph, $ABG(k)$ with N vertices;
2. Embed the $ABG(k)$ into the hypercube, $H(n)$;
3. Execute the parallel version of *bisort* and *bimerge* algorithms ;
4. Output the list of sorted numbers by an in-order traversal of the underlying bitonic tree, $BT(k)$.

end.

Figure 4.10 Bitonic Sorting Algorithm on Hypercubes (BSH)

Thus we have a new sorting algorithm for hypercubes which uses less number of comparisons and data exchanges than that of other implementations of bitonic sorting on hypercubes [Nass81a, Quin88a]. The actual algorithm, *BSH* is given in figure 4.10.

Theorem 4.2: The algorithm *BSH* has a time complexity of $O(log^2 N)$ and TP product $O(N \log^2 N)$, and the number of comparisons used is less than $2N log\ N$.

Proof: The time complexity, TP product, and the number of comparisons follow directly from the underlying adaptive bitonic sorting algorithm given in [Bila89a]. The constant dilation embedding of $ABG(k)$ into the hypercube ensures that the order of the time complexity and hence the TP product do not change. Hence the proof.■

Algorithm MABS

```
procedure bisort(root, spare, dir) ;
begin
1. if (root ↑ left = nil) then  test-and-swap(root, spare, dir)    /* down at leaves */
2. else begin
3. bisort(root ↑ left, root, dir) ; /* send message to left child */
4. bisort(root ↑ right, spare, ~dir) ;  /* send message to right child */
5. bimerge(root, spare, dir)
6. end
end;

procedure bimerge(root, spare, dir) ;
begin
1. rightexchange := (root ↑ value > spare ↑ value) XOR dir;
2. if rightexchange then swap_value(root, spare) ;
3. pl := root ↑ left ; pr := root ↑ right ;
4. while (pl <> nil) do begin
5.    elementexchange := (pl ↑ value > pr ↑ value) XOR dir;
6.    if rightexchange then
7.       if elementexchange then begin
8.       swap_value(pl, pr) ; /* using send and receive primitives */
9.       swap_right(pl, pr) ; /* sub-tree exchange using masking */
10.       pl := pl ↑ left ; pr := pr ↑ left ;
         end
11.   else begin
12.      pl := pl ↑ right ; pr := pr ↑ right ;
         end
13. else
14.      if elementexchange then begin
15.       swap_value(pl, pr) ; /* using send and receive primitives */
16.       swap_left(pl, pr) ; /* sub-tree exchange using masking */
17.       pl := pl ↑ right ; pr := pr ↑ right ;
         end
18.   else begin
19.      pl := pl ↑ left ; pr := pr ↑ left ;
         end
   end  /* while */
20. if (root ↑ left <> nil) then begin
21.   bimerge(root ↑ left, root, dir) ;
22.   bimerge(root ↑ right, spare, dir) ;
   end
  end; /* bimerge */
```

Figure 4.11 Algorithm Modified Adaptive Bitonic Sorting (MABS)

CHAPTER FIVE

A NEW FRAMEWORK USING SPATIAL DATA STRUCTURES

5.1 Introduction

The field of graph embedding has emerged as one of the active research areas in theoretical computer science. It is gaining importance as more and more interconnection networks based on regular graph structures are developed. The application of the techniques developed in graph embedding to many other areas in computer science needs more attention. The primary *motivation* of our research is derived from the fact that *"many of the existing graph embedding techniques have not been fully utilized in several applications"*. In view of this objective, we have investigated embedding techniques that may be utilized in solving some of the classical problems in parallel computer vision and image processing.

In this research, we have utilized the techniques to develop parallel algorithms as described in chapter 4. Recall that our technique consists of three phases:

1. *Identify the underlying data/computation structure*
2. *Augment the communication pattern to the structure*
3. *Embed the augmented structure into the network*

The first and third phases generally do not depend on the problem. This is at least the case in image processing. In this chapter, we focus on the first and the third phases of our technique. The second phase, namely, identifying the pattern of communication very much depends on the problem and hence is postponed to chapter 6.

We consider two data structures commonly used in computer vision and image processing applications. They are: *quad trees* and *octrees*. Quad trees are generally used to store and manipulate 2-dimensional digital images, while octrees are suitable for 3-dimensional image processing and graphics. We choose interconnection networks based on *boolean hypercubes* as the target parallel architecture. In this chapter, we focus on some of the new techniques for embedding these data structures into hypercubes.

The remainder of this chapter is organized as follows. The definitions and notations used in our algorithms are given in section 5.2. Section 5.2.1 discusses the importance of quad trees in computer vision applications with specific examples. Sections 5.2.2 and 5.2.3 presents our embedding algorithms in detail. As a bye-product, we have extracted some interesting features from our embedding techniques and given in section section 5.2.4. These are useful in the area of fault-tolerance for parallel architectures. Section 5.3.1 describes the use of octrees in 3-dimensional image processing and computer vision. The algorithm to embed octrees into hypercubes is explained in section 5.3.2.

5.2 Quad Trees as Spatial Data Structures for 2D Image Processing

Quadtrees have received a lot of attention in recent years as an efficient data structure for a variety of image processing applications. In the quadtree representation of an image the root represents an entire square region, and each node has either four children, each representing a quadrant of its parent, or is a leaf representing a quadrant. Such a quad tree used to represent the region is called a *region quadtree*. We

recall the example image, and its corresponding quadtree given in figure 1.3. We reproduce the same example [Same89a] for the purpose of illustration in figure 5.1.

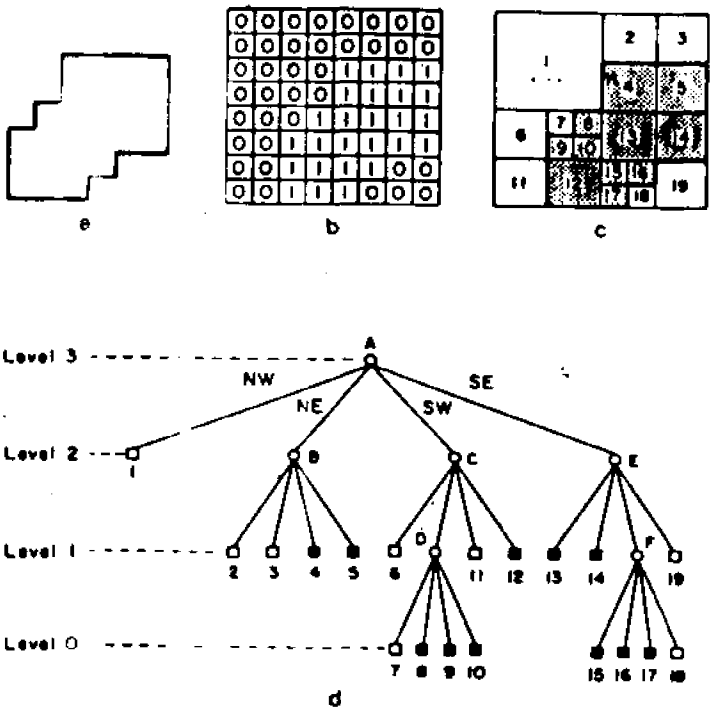


Figure 5.1 An Example of a *region quadtree*

We consider a region (part of an image) shown in figure 5.1(a). The corresponding binary array for the region is given in figure 5.1(b). Note that a 1 in the array correspond to a pixel in the region and a 0 to a pixel outside the region. Figure 5.1(d) shows the quadtree corresponding to the region. In the tree representation, the root node corresponds to the entire array. Each child of a node represents a quadrant (labeled in order NW, NE, SW, SE) of the region represented by that node. The leaf nodes of the tree correspond to those blocks for which no further subdivision is necessary. A leaf node will be black or white depending on whether its corresponding block is entirely inside or entirely outside the region.

Klinger and Dyer [Klin76a] provide a good bibliography of the history of quadtrees. The reader is referred to the comprehensive paper on quadtrees by Samet

[Same84a]. Jones and Iyengar [Jone84a] have studied efficient ways of storing quadtrees. The issues in the related work are representations of different features of images, space efficient quadtrees [Garg82a, Jone84a, Same83a] , and developing time efficient algorithms to perform some common operations on images [Same81a, Same81b, Same82a]. We use a different representation of a binary image. We use a complete quadtree to represent an image. The leaves of the quadtree correspond to the individual pixels in the image, and the internal nodes act as communication links. There are basically two reasons why we use a complete quadtree as a data structure:

- The problems in image processing which are related to distances between pixels, such as, *neighbor-finding*, *digital distances*, etc. do not need the region quadtree representation.
- It is relatively easier to embed a complete quadtree than an arbitrary quadtree (in case of region quadtree) into an interconnection network. We denote such a representation as *pixel quadtree*, and an example is given in figure 5.2.

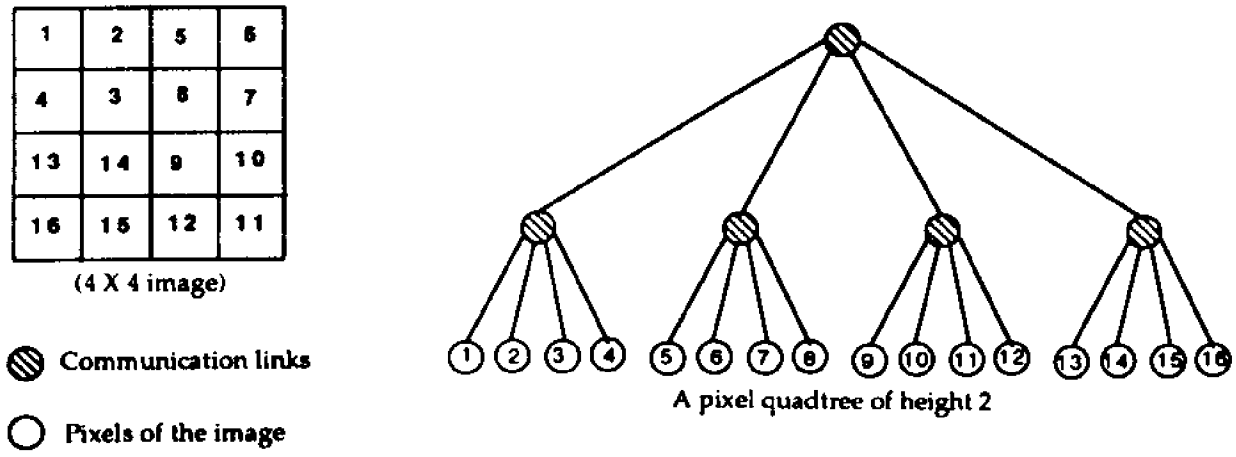


Figure 5.2 An Example of a *pixel quadtree*.

5.2.1 Notations and Preliminaries

Complete Quadtree, $CQT(h)$

We denote a Complete Quadtree of height h, as $CQT(h)$. The root of $CQT(h)$ is at level 0, and the leaves are at level h. The number of vertices at level i is 4^i. We now introduce some terminology pertaining to $CQT(h)$.

- The number of leaves, l is given by, $l = 4^h$.
- The number of vertices at level $h-1$ (i.e., the parents of leaves) denoted by, p, is given as: $p = 4^{h-1}$.
- The total number of vertices in $CQT(h)$ denoted by, m, is given as: $m = \frac{4^{h+1} - 1}{3}$.

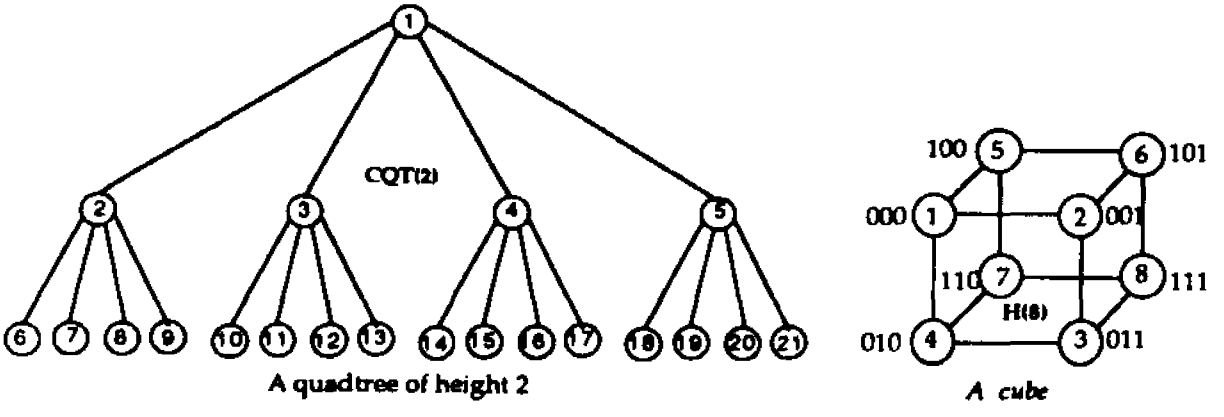


Figure 5.3 A Quadtree and a Hypercube

We perform a breadth-first-search on $CQT(h)$ and assign to each vertex, its BFS-Number, called *BFSN*. For example, the *BFSN* of root is 1, its children are 2,3,4 and 5, and so on. We refer every vertex by its *BFSN*. Figure 5.3(a) shows an example quad-tree with $h = 2$; $l = 16$; $p = 4$; $m = 21$; The numbers shown inside the circles indicate the BFSN's.

Boolean Hypercube, $H(n)$

We denote a hypercube with n nodes, as $H(n)$. We represent every node in $H(n)$ by a unique binary number (k-bit address) $b_k, b_{k-1} \cdots, b_1$, where k is the dimension of $H(n)$. Note that a node is adjacent to k nodes in $H(n)$. Also two nodes in $H(n)$ are adjacent when their addresses differ in exactly one of k bits. $H(8)$ is called a *cube*. Figure 5.3(b) shows a cube.

Our quadtree embedding algorithm builds an array, $B[1..m]$, where $B[i]$ contains the address of a unique node in $H(n)$ for the vertex of $CQT(h)$ whose *BFSN* is i. We now state a simple lemma that is useful in following the algorithms.

Lemma 5.1: The height of $CQT(h)$ and the dimension, k of $H(n)$ are related as: $k = 2h + 1$.

Proof: We prove by induction on h.

Basis: The claim is true for $h = 1$. We need a *cube* ($k = 3$) to embed $CQT(1)$.

Hypothesis: We let the claim true for $CQT(h-1)$. Assume that we need a hypercube of dimension k'. Then we have $k' = 2(h-1) + 1$....(i)

Step: Now consider embedding $CQT(h)$ into a hypercube of dimension k. $CQT(h)$ is obtained by taking 4 copies of $CQT(h-1)$ and having a new root that is connected to all the 4 roots of $CQT(h-1)$'s. Hence the number of nodes in the hypercube to embed $CQT(h)$ is 4 times that is used to embed $CQT(h-1)$. That is, $k = k' + 2$....(ii)

From (i) and (ii) we get $k = 2h + 1$, and hence the proof.■

We now state a corollary which follows from the above lemma.

Corollary 5.1.1: The number of *cubes*, p, is equal to *one-fourth* of l (the leaves).

We present a few more definitions below.

- The addresses of the eight nodes in a *cube* have identical bits in their $k-3$ most significant position. We call the binary number formed by taking these $k-3$ bits, the *address* of the *cube*.
- All *cubes* that have 2 m.s.bs identical are said to be in a *quadrant*, and these 2 m.s.bs identify that *quadrant*. Further, every *quadrant* is recursively divided into four *sub-quadrants* by making use of the next 2 m.s.bs until no such division is possible.
- Two *cubes* are said to be *similar*, if the bits in their addresses are identical, except two successive bits starting at the same odd bit position. Note that these two differing bits will identify a *quadrant* or a *sub-quadrant*. For example, if addresses of *cube1* and *cube2* are 01 01 00 00 01 and 01 11 00 00 01 then they are *similar*.
- In any *quadrant* or *sub-quadrant* we get four *similar cubes*. In each of these *cubes*, all the eight nodes are identified using three least significant bits (l.s.bs). The nodes having three identical l.s.bs in these four *cubes* are said be *similar*. A *square* is formed among *similar nodes*.

In this section, we give two different embedding algorithms. *Algorithm_2D* embeds the $CQT(h)$ into $H(n)$ with a dilation of 2, while *Algorithm_3D* does with a dilation of 3. The algorithms are explained in detail with embedding a $CQT(2)$ into $H(32)$ as an illustrative example. Section 5.2.2 describes the *Algorithm_2D* while sec-

tion 5.2.3 explains the *Algorithm_3D*.

5.2.2 A 2-Dilation Algorithm to Embed Quad Trees into Hypercubes

This section is primarily concerned on a 2-dilation embedding of a complete quadtree into its nearest hypercube. Ho and Johnssson [Ho89a] have shown that a $CQT(h)$ can be embedded into $H(n)$ with dilation 2. However, specific algorithms to do the embedding are not found in the literature. We give one such algorithm here.

Theorem 5.1: A complete quadtree of height h, $CQT(h)$ may be embedded into its nearest hypercube, $H(n)$ with at most dilation 2.

Proof: Our embedding function, f_h, builds an array, $B[1..m]$, where $B[i]$ contains the address of a unique node in $H(n)$ for the vertex of $CQT(h)$ whose *BFSN* is i, and m refers to the total number of vertices in $CQT(h)$. Formally, f_h is defined as follows:

$f_h : i \rightarrow B[i]$

where

$B[i] = b_k b_{k-1} \cdots b_1$

h is the height of $CQT(h)$;

i is the BFSN of any vertex in $CQT(h)$;

$k = log(n)$ is the dimension of $H(n)$.

We obtain f_h by a recursive construction on h and prove the theorem by induction on the height of $CQT(h)$.

Basis:

The basis of induction is given for $h = 1$. The nearest hypercube needed to embed

$CQT(1)$ is $H(8)$ as given by lemma 5.1.1. The binary representation of any node in $H(8)$ will have 3-bit address, say b_3,b_2,b_1. We give a 2-dilation constructive embedding of $CQT(1)$ into $H(8)$ as follows:

Let the root of $CQT(1)$ be assigned to the node with address 000 in $H(8)$, i.e, $b_3 = b_2 = b_1 = 0$. The four children of the root of $CQT(1)$ have to be assigned to nodes with unique addresses which is different from that of the root. This is achieved by using the permutations of the bits b_3 and b_2 and assigning a 1 for b_1 when both b_3 and b_2 are complements of the corresponding bits of the root, while assigning a 0 for b_1 otherwise. Note that such an assignment is unique and maintains a dilation of 2. More specifically, the four permutations of the bits b_3 and b_2 are: 00, 01, 11, 10. The permutation in which both the bits are complements of that of root is 11. Hence the b_1 will be a 0 for the child with $b_3 b_2 = 11$, and the b_1 will be a 1 for all other children. For $h = 1$ the actual embedding of $CQT(1)$ into $H(8)$ is given in figure 5.4 (b).

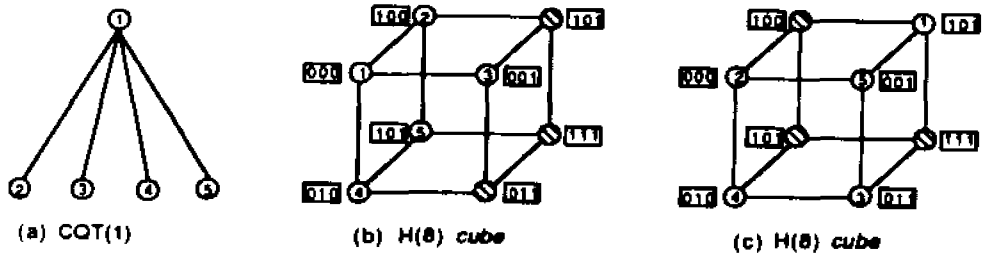


Figure 5.4 Embedding of $CQT(1)$ into $H(8)$

The array, B obtained using the function $f_h : i \rightarrow B[i]$ is given below for $h = 1$:

$B[1]$=000; $B[2]$=001; $B[3]$=011; $B[4]$=110; $B[5]$=101;

The theorem 5.1 clearly holds good for our basis.

Hypothesis: For any height $h' < h$, a $CQT(h')$ can be embedded into a hypercube $H(n')$ with dilation 2, where n' is the number of nodes in the nearest hypercube needed to embed $CQT(h')$. Note that the number of bits needed to address any node in $H(n')$ is $k' = 2h' + 1$ from lemma 5.1.1 and $k' = log(n')$.

Induction Step: We shall prove that $CQT(h)$ can be embedded into $H(n)$ with dilation 2 by obtaining f_h defined before. Assume that there exists an embedding function $f_{h'}$ which satisfies the induction hypothesis. Let $h' = h-1$. We make the following observations needed to prove the theorem.

Observation 5.1.1: The $CQT(h)$ can be obtained from $CQT(h-1)$ by adding four children to each of the leaves in $CQT(h-1)$. In other words, all the newly added nodes become the leaves of $CQT(h)$.

Observation 5.1.2: From Lemma 5.1.1, the height of a $CQT(h)$, and the dimension, k of the hypercube $H(n)$ are related as $h=2k+1$. Since $k=\log(n)$ refers to the number of bits in the binary representation of the nodes of the hypercube, $k = k' + 2$ where k' is the dimension of the nearest hypercube needed to embed $CQT(h-1)$.

We provide the embedding function, f_h for the vertices of $CQT(h)$ in two steps as follows:

Step 1: Embed internal vertices of $CQT(h)$

Let v_i be an internal vertex in $CQT(h)$. Let $v_{i'}$ be the vertex in $CQT(h-1)$ with the same BFSN as that of v_i. Let X' be the binary representation of the node assigned for $v_{i'}$ using $f_{h'}$. We obtain the binary representation, X of v_i by concatenating X' with two 0's, i.e. $X = X'00$. Formally, f_h for all the internal vertices of $CQT(h)$ may be given as:

$f_h : i \rightarrow B[i]$

where

$B[i] = CONCAT(B'[i], 00)$;

CONCAT is a concatenation function;

$B'[i]$ is obtained using f_{h-1} as $f_{h-1} : i \rightarrow B'[i]$.

Claim 5.1.1: Step 1 embeds all the internal vertices of $CQT(h)$ into unique nodes in $H(n)$ and maintains a dilation of 2.

Proof: Since the array B generated by f_h has a 1-1 correspondence with the addresses of the nodes in $H(n)$, it is sufficient to prove that the elements in B are distinct. From the hypothesis it follows that the elements in B' are distinct and the embedding has a dilation of 2. Since $B[i]$ is obtained by concatenating $B'[i]$ with 00, every element in B is distinct and maintains the dilation 2. Hence the claim.■

Step 2: Embed leaves of $CQT(h)$

Consider a set of four leaves in $CQT(h)$ which have a common parent. Let p be the BFSN of the parent, and let $CH[i], \forall i = 1,4$ refer to these four leaves ordered by their BFSN's. We obtain $B'[p]$ using f_{h-1}. We obtain the node assignments for the children depending on whether the l.s.b of $B'[p]$ is a 0 or a 1.

Case I: l.s.b of $B'[p]$ is a 0

The f_h for these four leaves may be given as:

$f_h : BFSN_i \rightarrow B[BFSN_i], \forall i = 1,4$

where

$BFSN_i$ is the BFSN of the $CH[i]$;

$B[BFSN_1] = CONCAT(EXPLOD(B'[p],1),001)$;

$B[BFSN_2] = CONCAT(EXPLOD(B'[p],1),011)$;

$B[BFSN_3] = CONCAT(EXPLOD(B'[p],1),110)$;

$B[BFSN_4] = CONCAT(EXPLOD(B'[p],1),101)$;

CONCAT is a concatenation function;

EXPLOD(S,i) is a function which takes two arguments, an array, S, and an index, $i < |S|$, and returns a new array of size $|S| - i$ in which all the i l.s.b's of S are removed.

Case II: l.s.b of $B'[p]$ is a 1

The f_h for these four leaves may be given as:

$f_h : BFSN_i \rightarrow B[BFSN_i], \forall i = 1,4$

where

$BFSN_i$ is the BFSN of the $CH[i]$;

$B[BFSN_1] = CONCAT(EXPLOD(B'[p],1),001)$;

$B[BFSN_2] = CONCAT(EXPLOD(B'[p],1),010)$;

$B[BFSN_3] = CONCAT(EXPLOD(B'[p],1),111)$;

$B[BFSN_4] = CONCAT(EXPLOD(B'[p],1),101)$;

CONCAT is a concatenation function;

EXPLOD(S,i) is a function which takes two arguments, an array, S, and an index, $i < |S|$, and returns a new array of size $|S| - i$ in which all the i l.s.b's of S are removed.

The procedure given for a set of four leaves may be repeated for every other set of four leaves.

Claim 5.1.2: Step 2 embeds all the leaves of $CQT(h)$ into unique nodes in $H(n)$ and maintains a dilation of 2.

Proof: Since the array B generated by f_h in step 2 has a 1-1 correspondence with the addresses of the nodes in $H(n)$, it is sufficient to prove that the elements in B are distinct. From the hypothesis it follows that the elements in B' are distinct and the embedding has a dilation of 2. Consider any set of four leaves as mentioned above. The addresses of these four leaves will be different in their $(k-3)$ m.s.b's, from those of leaves in the other sets since the address of the parents of the latter would be different in B'. The addresses of these four leaves are obviously different in their three l.s.b's. Hence every element in B is distinct.

Since the parent, p of these four leaves is an internal vertex in $CQT(h)$, the three l.s.b's of $B[p]$ can be either 000 or 100, which are considered in Case I and Case II respectively. In both cases it may be verified that the addresses $B[p]$ and $B[BFSN_i]$, $\forall\, i = 1,4$ differ in at most two bit positions. Hence step 2 maintains a dilation of 2. Hence the claim.■

5.2.3 A 3-Dilation Algorithm to Embed Quad Trees into Hypercubes

This section describes an algorithm to embed a $CQT(h)$ into $H(n)$ with at most 3 dilation. The technique is simpler than the algorithm *Embed_2D*. The important characteristics of this algorithm is that all four children of a root (or subroot) are placed in a *square* of the hypercube. The root is placed adjacent to one of its children. This results in a 3-dilation embedding, because one of the children will be at a distance 3 from its parent. The embedding of $CQT(1)$ into $H(8)$ is given in figure 5.4(c).

Note that the children of the root of $CQT(1)$ are placed in a square, 000–001–011–010. The embedding of $CQT(2)$ into $H(32)$ is shown in figure 5.5.

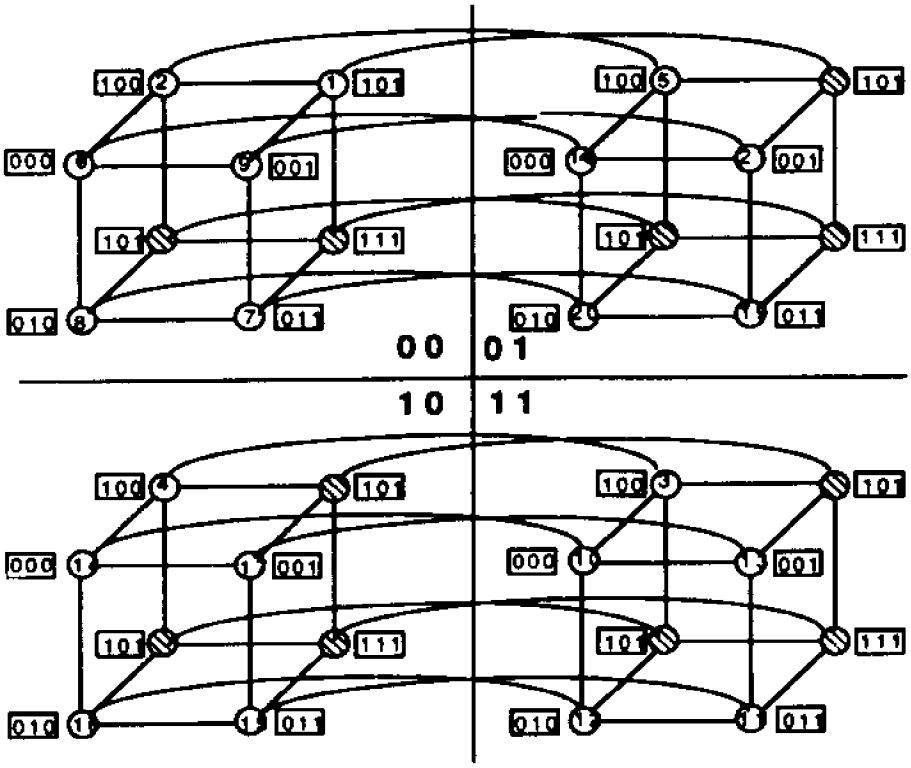


Figure 5.5 Embedding of $CQT(2)$ into $H(32)$

We now give an informal discussion on the algorithm *Embed_3D* with the help of figure 5.5. The root (1) is assigned to the hypercube node in the rightmost corner of the quadrant **00**, whose address is 00101. This is done by the very first step in *Embed_3D*. Then we call a recursive procedure *Embed_children_3D* to place the children of the root. As we stated earlier, the children are placed in the hypercube nodes that form a *square*. The subroots (2),(3),(4) and (5) are placed in quadrants **00**,**11**,**10** and **01** at addresses 00100,11100,10100 and 01100 respectively. Note that these nodes form a *square*. The children of these subroots are placed similarly in their respective quadrants. In order to avoid duplication, the procedures *Find_child_index* and *Vertex_index* are not included in the algorithm *Embed_3D*. Note that the

interconnections among the hypercube nodes in different quadrants are not shown in the figure 5.5 for the sake of clarity.

The actual algorithm is given in figure 5.6. We conclude this section with a theorem.

Theorem 5.2: The algorithm, *Embed_3D* embeds a complete quadtree of height h into its nearest hypercube with at most three dilation.

Proof: The recursive procedure, *Embed_children_3D* is called with the three parameters: (i) BFSN of the root (or subroot), (ii) the level of the root (or subroot), and (iii) the number of bits to work with (which is initially all the k bits). Notice in the algorithm, this recursive procedure is called with different values of these parameters at different times. Since the assignment of addresses to the vertices depends on these parameters, the embedding is one-to-one. When the number of bits to work with ($=k$) becomes 3, the recursive call terminates. Since k is an odd number (from lemma 5.1.1) and it is reduced by 2 in every recursive step, eventually the algorithm terminates.■

```
Algorithm Embed_3D ;
/*    Input:
/*       1. A complete quadtree of height h  (CQT(h))
/*       2. A hypercube with n  nodes (H(n) is the optimal)
/*    Output:
/*       An array, B[1..m] of binary numbers of length k = log(n) where m  is
/*       the number of vertices in CQT(h).
/*       B[i] contains the address of a unique node in H(n) for the vertex of
/*       CQT(h) whose BFSN is i.
begin
      Set B[1] = 00 01 01 01 ··· 01 101 ;
      Embed_children_3D(1 , 0 , k );
end.
Procedure Embed_children_3D (index , level , k) ;
begin
      c = B[index];       /* make a copy of the node address of the root     */
                          /* or the subroot in a binary number, c .      */
      Find_child_index (children , index , level) ;
       /* From the index and the current level of the root (or the subroot),
       /* find out the indices of its children and store them in the array , 'children'.
      if ( k = 3 ) then
      begin
            c_k = 0;
            c_{k-1} = 0 ; c_{k-2} = 0 ; B[children[1]] = c ; c_{k-1} = 1 ; c_{k-2} = 1 ; B[children[2]] = c ;
            c_{k-1} = 1 ; c_{k-2} = 0 ; B[children[3]] = c ; c_{k-1} = 0 ; c_{k-2} = 1 ; B[children[4]] = c ;
      end
      else if ( k = 5 ) then
            begin c_1 = 0 ;  Embed_recursive(c , level , k) ; end
      else
            begin c_{k-2} = 0 ; c_{k-3} = 0 ;  Embed_recursive(c , level , k) ; end;
end;
Procedure Embed_recursive (c , level , k) ;
begin
      c_k = 0 ; c_{k-1} = 0 ; B[children[1]] = c ;
      Embed_children_3D(children[1] , (level + 1), (k − 2));
      c_k = 1 ; c_{k-1} = 1 ; B[children[2]] = c ;
      Embed_children_3D(children[2] , (level + 1), (k − 2));
      c_k = 1 ; c_{k-1} = 0 ; B[children[3]] = c ;
      Embed_children_3D(children[3] , (level + 1), (k − 2));
      c_k = 0 ; c_{k-1} = 1 ; B[children[4]] = c ;
      Embed_children_3D(children[4] , (level + 1), (k − 2));
end;
```

Figure 5.6 3-Dilation Embedding Algorithm

5.2.4 Fault Tolerant Characteristics of Our Embedding

The approaches taken to provide fault tolerance in interconnection networks fall into the following three categories : *architecture-based*, *algorithm-based* and *reconfiguration-based*. The motivation to develop schemes based on the first approach is that the computational graph structure is fixed and hosts may be built in a such a way that, in case of faults, it can maintain the topology of the computational graph. Spare nodes and edges may be added to the basic host architecture. The resultant architecture will facilitate fault recovery by providing subgraphs isomorphic to the original graph [Dutt88a]. On the other hand, in algorithm-based techniques, the underlying graph structure is made robust using redundant paths without altering the architecture of the host [Horo81a, Desp78a]. In some cases, the fault detection and recovery are in-built in the algorithm itself. Bastani, et.al [Yen91a] have described a parallel sorting algorithm with inherent fault tolerance. In reconfiguration schemes, there are two stages: fault detection and relocation of faulty nodes to free nodes. These schemes provide intelligent mapping of the computational graph into the inter-connection network so that reconfiguration becomes easy. There is a growing need for designing reconfiguration algorithms for commonly used interconnection networks such as hypercubes. Chen, et.al., [Chen88a] have provided efficient reconfiguration algorithms for grids into hypercubes. The algorithm [Chen88a] enables the system to recover from a single failure in one-step by locating a free node in the hypercube. The time taken by the system to reconfigure itself after the detection of faults is the primary concern of the algorithm given in [Chen88a].

The previous techniques do not address the problem of maintaining the quality of initial mapping, after reconfiguration. In this dissertation, we investigate methodologies to develop reconfigurable mapping strategies that maintain certain important qualities of the initial mapping. As an example, we give reconfigurable mapping algorithms for embedding a complete quadtree into the hypercube in a faulty environment. The characteristic feature of our technique [Kris91a] is that the dilation in the reconfigured system is a constant. We have also provided the theory of reconfiguration in parallel architectures for a class of faults [Pras89a].

Fault Model

In this section we state the model used for fault tolerance. We explain the underlying assumptions made regarding the faults, during the execution of our reconfiguration algorithms.

- We assume that the links are fault-free.
- Reconfiguration is carried out after the faults are detected.
- The model tolerates faults up to a maximum of available *free nodes* after the initial embedding.
- We assume that reliable fault diagnosis mechanisms are available.
- Any node that detects a fault may initiate the reconfiguration algorithm.
- Once a *free node* is assigned (or used) in the recovery process, it can not be reassigned for recovering any other faults in future.
- We identify a class of faults, called *homogeneous faults*, that occur very frequently in computations based on tree structures. Faults occur only at the leaves

of the tree and further they are restricted for even distribution at each level of the tree. The motivation for considering such class of faults is equal probability of faults at any leaf, the number of *free nodes* is less than the total number of leaves, and *free nodes* are evenly distributed at each level of the tree.

Reconfiguration Algorithm

Following are the three principal steps involved in reconfiguration.

(1) *Find free node distribution:* Collect the number of *free nodes* in each of the *cubes* and form a sequence, called FN. FN is an array of size $(m-l)$. The indices of FN and B are the same, namely, the BFSN of the current vertex. $FN[1]$ contains the total number of *free nodes* in the hypercube and $FN[2]$,$FN[3]$,$FN[4]$,$FN[5]$ have the number of *free nodes* in quadrants **00**,**11**,**10** and **01** respectively. This counting process continues until a *cube* is encountered. Though the FNs are identical, the actual *free nodes* in these two embeddings are not the same (can be easily verified).

(2) *Find fault distribution:* From the given homogeneous fault sequence obtain a fault distribution, called FT. This is again similar to obtaining FN, except that it may vary from time to time based on the occurrence of faults.

(3) *Reconfiguration:* This step performs the actual reconfiguration. The idea here is to transform the fault distribution to the free node distribution. These two distributions are stored in arrays, and the indices are the BFSN of the quadtree vertices. Hence, we can think of these distributions arranged in the form a quadtree of height $h-1$. We use figure 5.7 to discuss the reconfiguration strategy.

We now discuss the transformation of *FT* to *FN* with the help of figure 5.7.

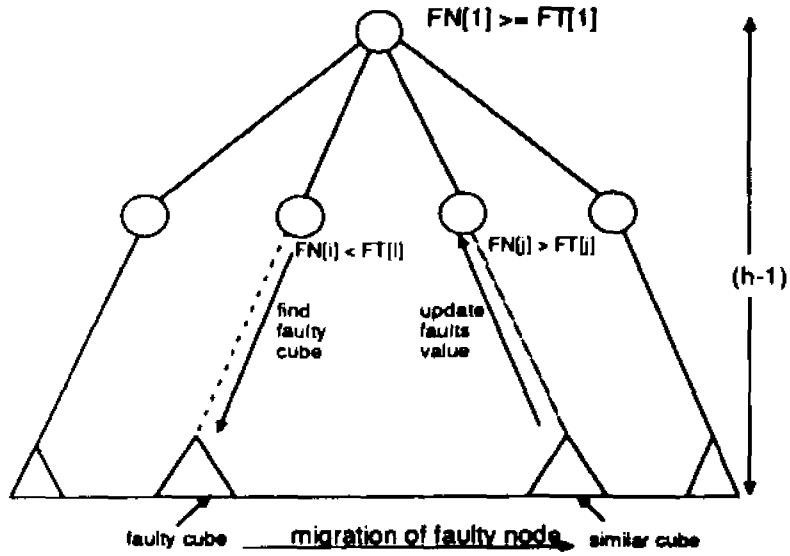


Figure 5.7 Reconfiguration Process

1. The distribution lists are traversed in a breadth-first fashion, and at each level the contents of these lists with the same index, are compared.

2. If the free node value is greater or equal to that of fault value then the algorithm does nothing.

3. Consider the case when the free node value is smaller than the fault value. Let i be the index (or vertex in the tree). That is $FN[i] < FT[i]$. Then there must exist an index j such that i and j have a common parent and $FN[j] > FT[j]$. This situation implies that there is a *cube* in the quadrant where the subtree rooted at i is placed, and has more faults than the free nodes. We now traverse down the subtree rooted at i and locate this *faulty cube*.

4. Locate the *similar cube* in the subquadrant where the subtree rooted at j is placed. Note that there is at least an *extra free node* in this *cube*.

5. Transfer the *extra* fault in the *faulty cube* to the *similar cube* located in the previous

step.

6. Update the faults distribution in both the subtrees rooted at i and j.

7. Repeat the above steps for all the levels of the quadtree.

We conclude this chapter with the following theorem.

Theorem 5.3: The reconfiguration algorithm maintains a constant dilation of 3.

Proof: We claim that the migration of *extra* fault to an *extra free node* does not result in more dilation. It can be easily seen from the fact that the migration takes place among *similar cubes*. From the definition of *similar cubes*, it can be shown that the parents of the *faulty leaf* and the *extra free node* are adjacent in the hypercube. Hence, the claim.■

5.3 Octrees as Spatial Data Structures for 3D Image Processing

5.3.1 3D Data Representation and Manipulation

The representation of 3D objects is essential for image understanding, computer graphics, solid modeling, CAD/CAM systems, and other related areas. There are three major approaches to the representation of objects [Requ83a]:

- Constructive Solid Geometry (CSG)
- Boundary representations
- Spatial enumeration methods

The above approaches can also be classified into two schemes: *volumetric descriptions* and *surface descriptions*. Advantages and disadvantages of each category are given in [Requ80a]. A particular method included in the category of spatial enumeration methods is the use of octrees, which have received attention in the literature. For example, to perform motion trajectory planning in robotics, it is generally required to have an approximate representation of the space occupied by objects in the workspace. Octrees are the most commonly used hierarchical spatial data structure to represent approximate shape of an object in 3-D world.

Octrees have many attractive properties [Meag82a]:

- Octrees are convenient for the representation of highly irregular objects, including those with internal cavities
- They permit efficient execution of Boolean operations, null object detection, and the calculation of integral properties

- They are convenient to input approximate objects by automatic means

The complete description of other schemes, and the various methods of storing and manipulating octrees is beyond the scope of this dissertation. We restrict our discussion to the simple representation of 3D data using octrees. We describe how efficient parallel three dimensional image processing algorithms can be developed using our new parallelization technique and using octrees as the underlying data structure.

We now formally define an octree and how the 3D data is represented by using octrees.

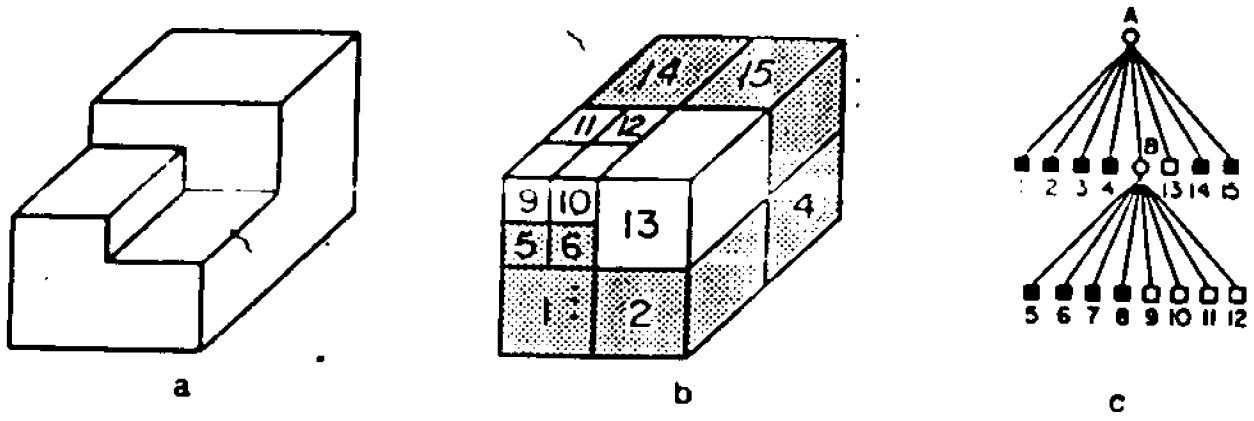


Figure 5.8 An Example of a *region octree*

The concept of shape representation by using octrees is basically obtained from that of quadtrees [Same89a]. We start with a $2^n X 2^n X 2^n$ object array of unit cubes (termed *voxels*). The octree is based on the successive subdivision of an object array into octants. If the array does not consist entirely of 1s or entirely 0s, it is then further subdivided into octants, and so on, until cubes are obtained that consist of 1s or of 0s. Figure 5.8 illustrates this subdivision process. A sample three dimensional object is given in figure 5.8(a). It is decomposed as mentioned before and shown in figure 5.8(b). The corresponding octree is shown in figure 5.8(c). In the octree, the root node represents

the entire object and the leaf nodes correspond to those cubes of the array for which no further subdivision is necessary.

5.3.2 Algorithm to Embed Octrees into Hypercubes

This section is primarily concerned on a 2-dilation embedding of a complete octree into its nearest hypercube. Ho and Johnssson [Ho89a] have shown that a $COT(h)$ can be embedded into $H(n)$ with dilation 3. In fact, the algorithm given in [Ho89a] embeds a complete m-ary tree into its nearest hypercube with a dilation of $(log\ n)$. In case of *quadtrees* this result is optimum since the smallest dilation needed to embed a $CQT(h)$ into $H(n)$ is 2. On the other hand, this algorithm [Ho89a] may not yield an optimum dilation, while embedding octrees into hypercubes. Also, it was stated as a conjecture whether $(log\ n)$ is the *lower bound* for embedding a complete m-ary tree into hypercubes. In this section, we attempt to disprove the conjecture by providing a 2-dilation embedding of a $COT(h)$ into $H(n)$.

We now provide the necessary notations and definitions regarding a complete octree, $COT(h)$ of height h.

Complete Octree, $COT(h)$

We denote a Complete Octree of height h, as $COT(h)$. The root of $COT(h)$ is at level 0, and the leaves are at level h. The number of vertices at level i is 8^i. We now introduce some terminologies pertaining to $COT(h)$.

- The number of leaves, l is given by, $l = 8^h$.

- The number of vertices at level $h-1$ (i.e., the parents of leaves) denoted by, p, is given as: $p = 8^{h-1}$.
- The total number of vertices in $COT(h)$ denoted by, m, is given as: $m = \frac{8^{h+1}-1}{7}$.

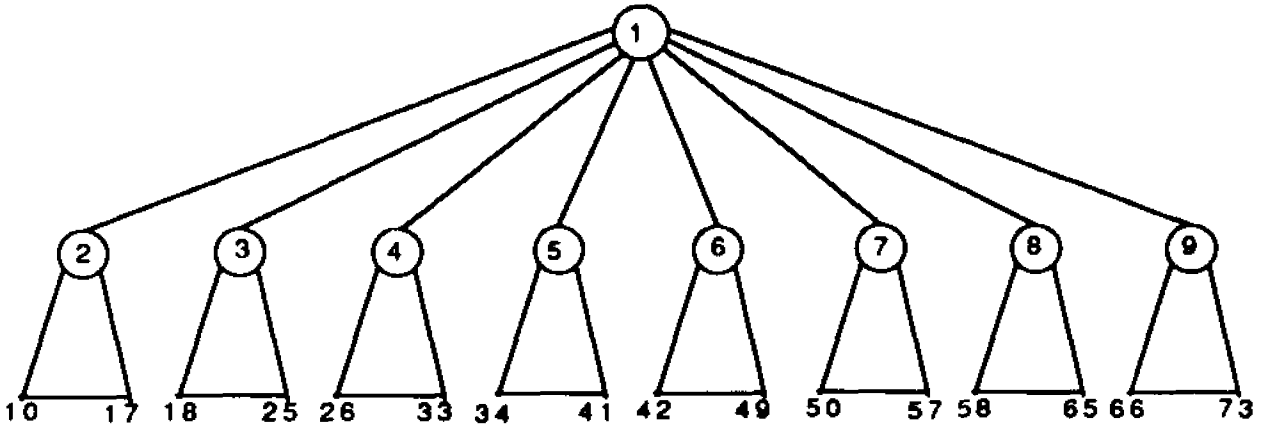


Figure 5.9 A Complete Octree, $COT(h)$

We perform a breadth-first-search on $COT(h)$ and assign to each vertex, its BFS-Number, called *BFSN*. For example, the *BFSN* of root is 1, its children are 2,3,4,5,6,7, and 8 and so on. We refer every vertex by its *BFSN*. Figure 5.9 shows an example octree with $h = 2$; $l = 64$; $p = 8$; $m = 73$; The numbers shown inside the circles indicate the BFSN's.

Our octree embedding algorithm builds an array, $B[1..m]$, where $B[i]$ contains the address of a unique node in $H(n)$ for the vertex of $COT(h)$ whose *BFSN* is i. We now state a simple lemma that is useful in following the algorithms.

Lemma 5.2: The height of $COT(h)$ and the dimension, k of $H(n)$ are related as: $k = 3h + 1$.

Proof: We prove by induction on h.

Basis: The claim is true for $h = 1$. We need a hypercube, $H(16)$ to embed $COT(1)$.

Hypothesis: We let the claim true for $COT(h-1)$. Assume that we need a hypercube of dimension k'. Then we have $k' = 3(h-1) + 1$....(i)

Step: Now consider embedding $COT(h)$ into a hypercube of dimension k. $COT(h)$ is obtained by taking 8 copies of $COT(h-1)$ and having a new root that is connected to all the 8 roots of $COT(h-1)$'s. Hence the number of nodes in the hypercube to embed $COT(h)$ is 8 times that is used to embed $COT(h-1)$. That is, $k = k' + 3$....(ii)

From (i) and (ii) we get $k = 3h + 1$, and hence the proof.■

We now formally present the embedding algorithm.

Theorem 5.4: A complete octree of height h, $COT(h)$ may be embedded into its nearest hypercube, $H(n)$ with at most dilation 2.

Proof: Our embedding function, f_h, builds an array, $B[1..m]$, where $B[i]$ contains the address of a unique node in $H(n)$ for the vertex of $COT(h)$ whose *BFSN* is i, and m refers to the total number of vertices in $COT(h)$. Formally, f_h is defined as follows:

$f_h : i \rightarrow B[i]$

where

$B[i] = b_k b_{k-1} \cdots b_1$

h is the height of $COT(h)$;

i is the BFSN of any vertex in $COT(h)$;

$k = log(n)$ is the dimension of $H(n)$.

We obtain f_h by a recursive construction on h and prove the theorem by induction on the height of $COT(h)$.

Basis:

The basis of induction is given for $h = 1$. The nearest hypercube needed to embed $COT(1)$ is $H(16)$ as given by lemma 5.2. The binary representation of any node in $H(16)$ will have 4-bit address, say b_4,b_3,b_2,b_1. We give a 2-dilation constructive embedding of $COT(1)$ into $H(16)$ as follows:

Let the root of $COT(1)$ be assigned to the node with address 0000 in $H(16)$, i.e, $b_4 = b_3 = b_2 = b_1 = 0$. We call the left most child as the *first child*, its adjacent node as the *second child* and so on. The eight children of the root of $COT(1)$ have to be assigned to nodes with unique addresses which is different from that of the root. This is achieved by using the permutations of the bits b_3, b_2 and b_2 and assigning a 1 to b_4 for the *last two* children, and assigning a 0 to b_4 for the *first six* children. Note that such an assignment is unique and maintains a dilation of 2. For $h = 1$, the actual embedding of $COT(1)$ into $H(16)$ is given in figure 5.10. Note that the interconnections among the hypercube nodes in different subcubes are not shown for clarity.

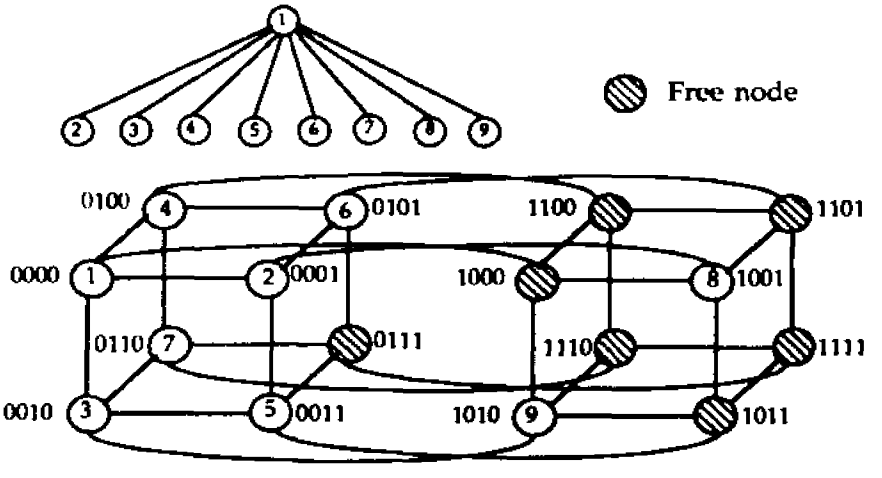


Figure 5.10 Embedding of $COT(1)$ into $H(16)$

The array, B obtained using the function $f_h : i \rightarrow B[i]$ is given below for $h = 1$:

$B[1]$=0000;

$B[2]$=0001; $B[3]$=0010; $B[4]$=0100; $B[5]$=0011;

B [6]=0101; B [7]=0110; B [8]=1001; B [9]=1010;

The theorem 5.4 clearly holds good for our basis.

Hypothesis: For any height $h' < h$, a $COT(h')$ can be embedded into a hypercube $H(n')$ with dilation 2, where n' is the number of nodes in the nearest hypercube needed to embed $COT(h')$. Note that the number of bits needed to address any node in $H(n')$ is $k' = 3h' + 1$ from lemma 5.2 and $k' = log(n')$.

Induction Step: We shall prove that $COT(h)$ can be embedded into $H(n)$ with dilation 2 by obtaining f_h defined before. Assume that there exists an embedding function $f_{h'}$ which satisfies the induction hypothesis. Let $h' = h-1$. We make the following observations needed to prove the theorem.

Observation 5.3.1: The $COT(h)$ can be obtained from $COT(h-1)$ by adding eight children to each of the leaves in $COT(h-1)$. In other words, all the newly added nodes become the leaves of $COT(h)$.

Observation 5.3.2: From Lemma 5.2, the height of a $COT(h)$, and the dimension, k of the hypercube $H(n)$ are related as $h=3k+1$. Since $k=\log(n)$ refers to the number of bits in the binary representation of the nodes of the hypercube, $k = k' + 3$ where k' is the dimension of the nearest hypercube needed to embed $COT(h-1)$.

We provide the embedding function, f_h for the vertices of $COT(h)$ in two steps as follows:

Step 1: Embed internal vertices of $COT(h)$

Let v_i be an internal vertex in $COT(h)$. Let $v_{i'}$ be the vertex in $COT(h-1)$ with the same BFSN as that of v_i. Let X' be the binary representation of the node assigned for

$v_{i'}$ using $f_{h'}$. We obtain the binary representation, X of v_i by concatenating X' with three 0's, i.e. $X = 000\,X'$. Formally, f_h for all the internal vertices of $COT(h)$ may be given as:

$f_h : i \rightarrow B[i]$

where

$B[i] = CONCAT(000\,, B'[i])$;

CONCAT is a concatenation function;

$B'[i]$ is obtained using f_{h-1} as $f_{h-1} : i \rightarrow B'[i]$.

Claim 5.3.1: Step 1 embeds all the internal vertices of $COT(h)$ into unique nodes in $H(n)$ and maintains a dilation of 2.

Proof: Since the array B generated by f_h has a 1-1 correspondence with the addresses of the nodes in $H(n)$, it is sufficient to prove that the elements in B are distinct. From the hypothesis it follows that the elements in B' are distinct and the embedding has a dilation of 2. Since $B[i]$ is obtained by concatenating 000 with $B'[i]$, every element in B is distinct and maintains the dilation 2. Hence the claim.■

Step 2: Embed leaves of $COT(h)$

Consider a set of eight leaves in $COT(h)$ which have a common parent. Let p be the BFSN of the parent, and let $CH[i]$, $\forall i$ =1,8 refer to these eight leaves ordered by their BFSN's. We obtain $B'[p]$ using f_{h-1}. We obtain the node assignments for the children as follows:

(1) The first 3 m.s.b's of the children one to six will be 001,010,100,011,101,110 respectively.

(2) The last 3 l.s.b's of the children one to six will be obtained by permuting the last 3 l.s.b's of the parent p in such a way that the resulting hamming distance between the parent and any of these children is at most 2.

(3) The first 3 m.s.b's of the children seven and eight will be 001,010 respectively.

(4) The last 3 l.s.b's of the children seven and eight will be the same as that of the last 3 l.s.b's of the children first and second.

The procedure given for a set of eight leaves may be repeated for every other set of eight leaves.

Claim 5.3.2: Step 2 embeds all the leaves of $COT(h)$ into unique nodes in $H(n)$ and maintains a dilation of 2.

Proof: Since the array B generated by f_h in step 2 has a 1-1 correspondence with the addresses of the nodes in $H(n)$, it is sufficient to prove that the elements in B are distinct. From the hypothesis it follows that the elements in B' are distinct and the embedding has a dilation of 2. Consider any set of eight leaves as mentioned above. The addresses of these eight leaves will be different in their $(k-3)$ m.s.b's, from those of leaves in the other sets since the address of the parents of the latter would be different in B'. The addresses of these eight leaves are obviously different in their three l.s.b's. Hence every element in B is distinct. Hence step 2 maintains a dilation of 2. Hence the claim.■

CHAPTER SIX

PARALLEL ALGORITHMS FOR SOME PROBLEMS IN IMAGE PROCESSING

2.1 Introduction

There has been growing interest in developing faster algorithms for the problems in computer vision. Current research is focussed on deriving parallel algorithms for the problems in *low* and *intermediate* level vision tasks due to the following reasons:

- low level operations exhibit *inherent* parallelism.
- it is easy to exploit data parallelism because most of the low level operations are data intensive.
- these low and intermediate level operations are performed *repetitively* and hence needed to parallelized.

Though designing parallel algorithms for the high level tasks would be useful, it is still in the very early stage of research. The main reason is that *symbolic computations* (constitute most of the high level operations) are difficult to parallelize as opposed to *numeric computations* (as found in low and intermediate level operations). In addition, there are no universal algorithms for the problems in low and intermediate level processing. More specifically, choosing an algorithm and an architecture is a difficult task. Also, there is no systematic way of designing parallel algorithm for these problems. So, researchers are still focusing on low and intermediate level tasks for parallelization. Ranka and Sahni [Rank90a] have compiled a set of parallel algorithms using the hypercube architecture for some of the problems in image processing and

pattern recognition such as *convolution, template matching, clustering* and *image transformations.*

In this dissertation, we develop parallel algorithms for some of the low level image processing tasks. Our algorithms are developed on the following basis:

- Pixel quadtrees are used to store the digital image
- Binary digital images are considered
- Parallel architecture based on hypercube interconnection network is used

We consider three classical problems in low level image processing: *nearest neighbor finding and computing digital distances.* The methodology for designing parallel algorithms is the same which is explained later in the chapter. Section 6.2 discusses the algorithm for parallel nearest neighbor finding problem. Computing digital distances in parallel is the focus of section 6.3.

6.2 Parallel Nearest Neighbor Finding Problem

Many of the low level image processing tasks such as convolution, and template matching are localized. In other words, the final value of a pixel after a single operation in these tasks depends on the pixel values of its neighbors. There are various definitions for a neighbor in digital image processing. We consider both the 4-connectivity or 8-connectivity approaches.

- In a 4-connectivity paradigm, a pixel has 4 neighbors in the directions left, right, up and down. They are also represented as in directions East, West, North and South respectively.

- In a 8-connectivity paradigm, a pixel has 8 neighbors in directions left, right, up, down, left-up-diagonal, left-down-diagonal, right-up-diagonal, and right-down-diagonal. They are also represented as East, West, North, South, North-West, South-West, North-East, and North-West respectively.

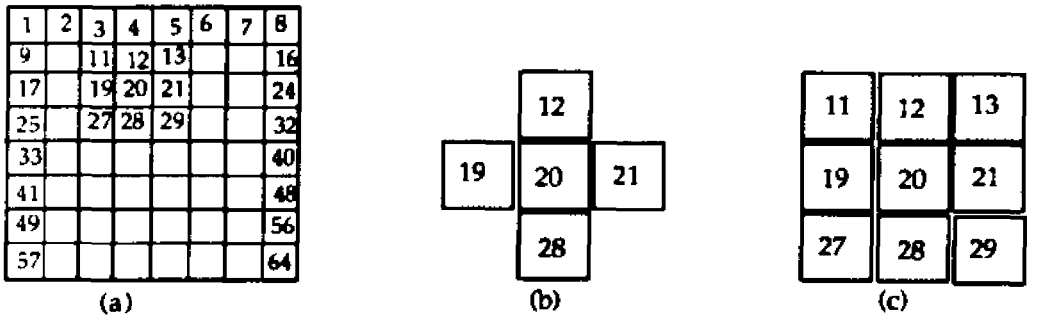


Figure 6.1 Examples of 4- and 8-connectivities

In figure 6.1(a) we show a binary digital image. Consider the pixel number 20. Its neighbors in case of 4-connectivity are 19, 21, 2, and 28 as shown in figure 6.1(b). Similarly, its neighbors in case of 8-connectivity are 19, 21, 2, 28, 11, 27, 13, and 29 as shown in figure 6.1(c).

There are several sequential neighbor finding algorithms available for *region quadtrees*. Samet [Same89a] has described a set of algorithms and analyzed the complexity of each one of these algorithms. The basic idea behind these algorithms is to find the *nearest common ancestor* in the quadtree. We pose a slight variation of the problem as follows. *Given two pixels X* and *Y* can we determine whether they are neighbors in parallel. We now informally outline the important steps in solving this problem. Since we use a pixel quadtree representation, these two pixels correspond to two leaves in the quadtree. These two leaves may then initiate a search (for answer to the problem) simultaneously. In a conventional quadtree based representation, these two leaves communicate to their parents and provide their BFSN's. These two parents

(internal nodes) may then communicate their ancestors this number. This process continues until a common ancestor is found. The common ancestor node then decides whether the two BFSN's it received, have either 4- or 8-connectivity relations satisfied.

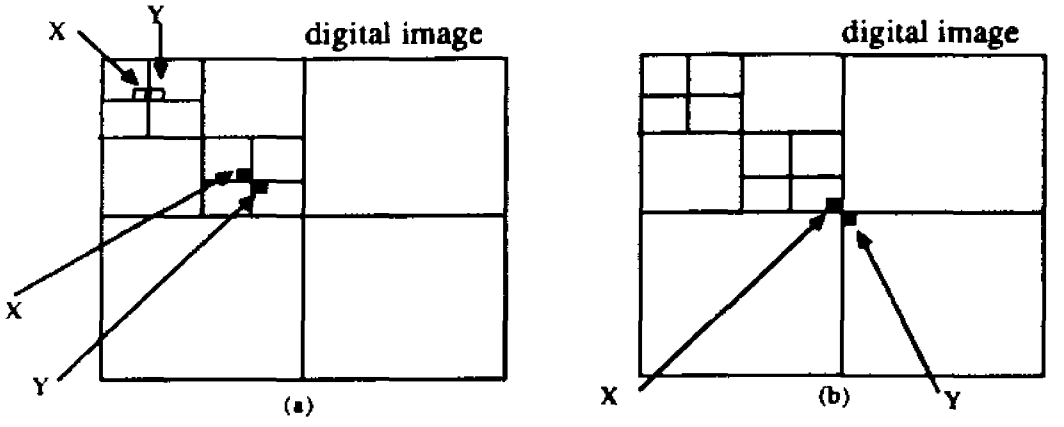


Figure 6.2 Examples of Neighbor-finding

Note that the search process will eventually terminate. In the worst case the root becomes the common ancestor and the algorithm must have traversed the entire quadtree. Figure 6.2(a) shows an example when the algorithm may not traverse the entire quadtree, while figure 6.2(b) shows the one in which the algorithm must traverse the entire quadtree.

We propose a new algorithm to solve this problem of parallel neighbor finding. The basic idea is simple. We first embed the pixel quadtree into the hypercube. The two leaves then broadcasts their BFSNs to their adjacent processors. Since our embedding algorithm places the four children in a *square*, these two leaves or their parents are kept *closer* in the hypercube. This property of the embedding allows a faster search process than the one described before. We formally provide the parallel neighbor finding algorithm, PNF, in figure 6.3.

```
Algorithm PNF;
/*  input:
/*    1. A pixel quadtree of a digital image, CQT(h).
/*    2. Two pixels, X, and Y
/*  output:
/*    Find out whether X and Y are adjacent.

begin
1. Embed the CQT(h) into H(n) using Embed_3D;
2. Send the message to processors where X and Y are stored
   to initiate the search;
3. for all processors do Common;
4. Receive the answer from the processor;
end.

procedure Common
begin
1. Receive the number from the adjacent processor;
2. if 2 different numbers received so far then
   begin
3.    Check for the satisfiablity of neighboring conditions;
4.    Report the answer to the controller (running PNF);
   end
5. else Store the number received ;
end;
```

Figure 6.3 Parallel Neighbor Finding Algorithm

6.3 Computing Digital Distances in Parallel

In many of the machine vision applications, the key to object recognition is the extraction of the geometrical properties of the object, such as area, perimeter, diagonal, and medial axis. All of these features involve computing distances between pixels that constitute the object. Thus there is a strong motivation to compute the distance between two pixels in a digital image. As we mentioned in the introduction, we consider only binary images. This means, we have two type of pixels: *zero pixel* that has a value of 0, and *one pixel* that has a value of 1. In short we call them *z-pixel* and *o-*

pixel.

There are several different notions about distances in a digital image. For example, we have euclidean, manhattan, chess-board distances. Refer to [Rose82a] for a complete description of these distances. In our research we are interested in a variation of these distances, called *digital distances*. The digital distance between two *o-pixels* is given as the number of *o-pixels* between the shortest path between these two while the path must consist of only *o-pixels*. More formally, we define digital distance as follows:

Definition 6.1: The digital distance, *dd* between pixels, x and y is given as:

$dd(x, y) = min(|S|)$

where $S = \{ p \mid p$ is a *z-pixel* and in the path between x and $y \}$

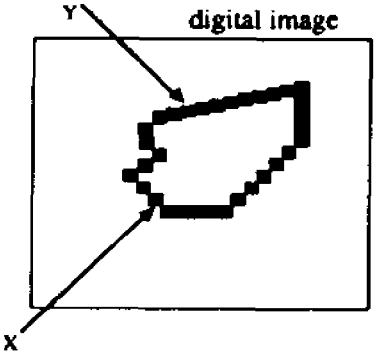


Figure 6.4 Example of Digital Distances

Figure 6.4 shows a binary image. In this image, the digital distance between the pixels x and y is 7.

In this dissertation we present a technique to compute the digital distance between two pixels in a binary image. The basic idea is to make use of the parallel neighbor finding algorithm to compute all the digital distances between the given two pixels and find the minimum of these distances.

CHAPTER SEVEN

CONCLUSIONS

In the foregoing chapters we have shown new techniques for solving problems in the two well known paradigms in computer and robotic vision, namely, scene understanding and parallel image processing. The technique to solve scene understanding is demonstrated with an illustrative example in the area of oceanographic satellite imagery. New parallelization techniques for image processing problems such as neighbor-finding and digital distances have been developed.

In chapter 2, we have described the overall methodology of our technique to automate the interpretation process in oceanographic satellite images. The individual components of our automatic interpretation system is explained with an illustrative example. We have designed a hybrid architecture for labeling the features in oceanographic images. The architecture blends the conventional relaxation-based technique with an expert system for interpretation. The feedback from the expert system to the relaxation labeling reduces the ambiguities introduced in the labeling process. As a part of our on-going research in this area, we have studied interpretation techniques in some of the related areas such as remote sensing, medical image processing, and astronomy, and shown that our technique is unique and powerful. In chapter 3, we have given a detailed description of our knowledge-based feature labeling architecture.

In the second part of the dissertation, we have presented new techniques for parallelization. In chapter 4, we have described a new, generalized technique for

developing parallel algorithms using standard interconnection networks such as hypercubes. We have emphasized the fact that there is no systematic way of designing parallel algorithms. In our new technique, we choose the best PRAM algorithm available for a specific problem; transform the algorithm to run on the given architecture. We have illustrated this with the help of a sorting example.

In chapter 5, we first have developed the motivation for storing and manipulating digital images using spatial data structures such as quadtrees and octrees. We then, show how these structures may be embedded into a hypercube-based architecture efficiently. We presented two algorithms to embed quadtrees into hypercubes, and explained the advantages and disadvantages of each one of them. As a bye-product, we have characterized the fault tolerant aspects of our embedding scheme. This also has led to the development of reconfiguration schemes, and are explained in section 5.2.4. The theoretical results obtained in chapter 5, are then applied to some of the problems in chapter 6.

Our research in these two paradigms has raised the following interesting questions or problems.

(1) The knowledge-based feature labeling algorithm may be made more consistent by fusing the information obtained from various sources such as altimetry, and bathymetry. An outline of this approach is already given in chapter 3.

(2) It may be interesting to design an expert system for the oceanographic feature interpretation problem by incorporating the recent research such as belief intervals.

(3) Our quadtree and octree embedding results have a lot of theoretical implications. It may be interesting to determine the lower bound on the maximum dilation after embedding a complete m-ary tree into its nearest hypercube.

(4) We have developed reconfiguration algorithm for a class of failures, called homogeneous faults. Solving the problem of reconfiguration for a different (and more) types of failures will make the system more fault tolerant.

(5) Our research in the area of fault tolerance has paved the way for a new class of algorithms called "Reconfigurable Embedding Algorithms". The basic idea is to provide an embedding in such a way that after recovering from faults, a constant dilation is maintained in the resultant system.

(6) Our new parallelization technique may be extended in two directions: choosing different computational data structures, and choosing different interconnection networks. Clearly there is lot of work to be done in this area.

(7) Is is possible to embed an arbitrary quadtree (such as a region quadtree) so that many of the existing quadtree algorithms may be parallelized?

(8) Our new parallelization technique may be applied to a class of problems in low and intermediate level vision.

References

[Akl89] Akl, S.G., *The Design and Analysis of Parallel Algorithms*, Prentice-Hall Inc., 1989.

[Anut73] Anuta, P.E., and Bauer, M., "An Analysis of Temporal Data for Crop Species Classification and Urban Change Detection," *LARS Irformation Note 110873, Laboratory for Applications of Remote Sensing, Purdue University*, 1973.

[Batc68] Batcher, K.E., "Sorting Networks and Their Applications," *Proceedings of the Spring Joint Computer Conference*, vol. 32, pp. 307-314, 1968.

[Bhat84] Bhatt, S.N., and Leiserson, C.E., *How to Assemble Tree Machines*, 2, pp. 95-114, JAI Press Inc.,, 1984.

[Bhat85] Bhatt, S.N., and Ipsen, E.C.F., "How to Embed Trees in Hypercubes," *RR-443, Dept. of Computer Science, Yale University*, 1985.

[Bila89] Bilardi, G., and Nicolau, A., "Adaptive Bitonic Sorting: An Optimal Parallel Algorithm for Shared Memory Machines," *SIAM Journal of Computing*, vol. 18(2), pp. 216-228, April 1989.

[Bokh81] Bokhari, S.H., "On Mapping Problem," *IEEE Transactions on Computers*, vol. C-30(3), pp. 207-214, March 1981.

[Broo81] Brooks, R.A., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*, vol. 17, pp. 285-349, 1981.

[Buch84] Buchanan, B.G., and Shortliffe, E.H., in *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*,

1984.

[Camb90] Cambridge, V.J., "Edge/Region Correlation in Image Segmentation and Analysis," *Proceedings of the Workshop on Automated Image Interpretation at NOARL*, October 1990.

[Cant90] Cantrell, M.S.L., and Holyer, R.J., "An Automated Technique for Locating the Gulf Stream in Altimeter Profiles," *Proc. 5th Conf. Satellite Meteorology and Oceanography*, pp. 139-144, 1990.

[Cart85] Carter, E.F., "The Structure of the Gulf Stream as Derived from an EOF Analysis," *Gulf Stream Workshop Proceedings, University of Rhode Island*, 1985.

[Cayu90] Cayula, J.F., and Cornillon, P., "Edge Detection Applied to Sea Surface Temperature Fields," *Proc. SPIE Tech. Symp. Optical Eng. and Photonics in Aerospace Sensing, Conf. Digital Image Process. and Visual Commun. Tech. in Earth and Atmos. Sci.*,, 1990.

[Cayu91] Cayula, J.F., Cornillon, P., Holyer, R., and Peckinpaugh, S., "Comparative Study of Two Recent Edge-Detection Algorithms Designed to Process Sea-Surface Temperature Fields," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 29(1), pp. 175-177, January 1991.

[Chen88a] Chen, J., Simonett, D.S., and Sun, G., "Computer-Aided Interpretation of Forest Radar Images," *International Conference on Pattern Recognition*, vol. I, pp. 1245-1249, 1988.

[Chen88b] Chen, S.K., Liang, C.T., and Tsai, W.T., "An Efficient Multi-

Dimensional Grids Reconfiguration Algorithm on Hypercubes," *International Conference on Parallel Processing*, pp. 368-373, 1988.

[Clan84] Clancey, W.J., and Shortliffe, E.H., in *Readings in Medical Artificial Intelligence: The First Decade*, 1984.

[Cley88] Cleynenbreugel, J.V., Fierens,. F., Suetens, P., and Oosterlinck, A., "Knowledge-Based Improvement of Automatic Image Interpretation for Restricted Scenes: Two Case Studies," *Journal of Image and Vision Computing*, vol. 6(4), pp. 238-246, November 1988.

[Corn87] Cornillon, P., Gilman, C., Stramma, L., Brown, O., Evans, R., and Brown, J., "Processing and Analysis of Large Volumes of Satellite-Derived Thermal Infrared Data," *Journal of Geophysical Research*, vol. 92(C12), pp. 12933-13002, 1987.

[Coul83] Coulter, "Application of the Bayes Decision Rule for Automatic Water Mass Classification from Satellite Infrared Data," *17th International Symposium on Remote Sensing of Environment*, vol. II, pp. 589-597, 1983.

[Cros88] Cross, A.M., "Detection of Circular Geological Features Using the Hough Transform," *Internation Journal of Remote Sensing*, vol. 9(9), pp. 1519-1528, 1988.

[Davi78] Davis, L.S., and Rosenfeld, A. , "Noise cleaning by iterated local averaging," *IEEE Transactions on System, Man and Cybernetics*, vol. 8, pp. 705-710, 1978.

[Davi83] Davis, L.S., Wang, C.Y., and Xie, H.C. , "An experiment in multispectral, multitemporal crop classification using relaxation techniques," *Computer Vision, Graphics and Image Processing*, vol. 23 , pp. 227-235, 1983.

[Desp78] Despain, A., and Patterson, D., "X-tree: A Structured Multiprocessor Computer Architecture," *5th Symposium on Computer Architecture*, pp. 144-151, April 1978.

[Dey90] Dey, S., and Srimani, P.K., "A New Parallel Sorting Algorithm and its Efficient VLSI Implementation," *THE COMPUTER JOURNAL*, vol. 33(3), pp. 241-246, 1990.

[Dutt88] Dutt, S., and Hayes, J.P., "Design and Reconfiguration Strategies for Near-Optimal k-Fault-Tolerant Tree Architectures," *IEEE-Conference on Fault Tolerance*, pp. 328-333, 1988.

[Eklu80] Eklundh, J.O., Yamamoto, H., and Rosenfeld, A., "A relaxation method in multispectral pixel classification," *IEEE-Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 72-75, 1980.

[Fu87] Fu, K.S., Gonzalez, R.C., and Lee, C.S.G, *Robotics : Control, Sensing, Vision and Intelligence*, McGraw-Hill International Editions, 1987.

[Garg82] Gargantini, I., "An Efficient Way to Represent Quadtrees," *Communications of ACM*, vol. 25, pp. 905-910, 1982.

[Gers77] Gerson, D.J., and Gaborski, P., "Pattern Analysis for Automatic Location of Ocean Fronts in Digital Satellite Imagery," *Naval Oceanographic*

Office Technical Report, vol. 3700-65-77, October 1977.

[Gers82] Gerson, D.J., Khedouri, E., and Gaborski, P., "Detecting the Gulf Stream from Digital Infrared Data Pattern Recognition," *The Belle W. Baruch Library in Marine Science*, vol. 12, pp. 19-39, 1982.

[Gold81] Goldberg, M., "Digital Image Processing of Remotely Sensed Imagery," in *Digital Image Processing, J.C. Simson and R.M. Haralick (eds.)*, pp. 383-437, 1981.

[Good87] Goodenough, D.G., Goldberg, M., Plunkett, P., and Zelek, J., "An Expert System for Remote Sensing," *IEEE-Transactions on Geoscience and Remote Sensing*, vol. 25(3)], pp. 349-359, May 1987.

[Hadd90] Haddon, J,F., and Boyce, J.F., "Image Segmentation by Unifying Region and Boundary Information," *IEEE-Transactions on Pattern Analysis and Machine Intelligence*, vol. 12(10), pp. 929-948, October 1990.

[Hanc90] Hancock, E.R., and Kittler, J., "Edge-Labeling Using Dictionary-Based Relaxation," *IEEE-Transactions on Pattern Analysis and Machine Intelligence*, vol. 12(2), pp. 165-181, February 1990.

[Hara82] Haralick, R.M., "Pattern Recognition of Remotely Sensed Data," in *Pictorial Data Analysis, ed R.M. Haralick*, pp. 351-367, 1982.

[Hard85] Hardy, D.D., "Today the Earth is 100 000 000 000 000 Bits," *International Conference on Advanced Technology for Monitoring and Processing Global Environment Data*, pp. 1-4, London, UK, 1985.

[Haye83] Hayes-Roth, F.A., Waterman, D.A., and Lenat, D.B., in *Building Expert*

Systems, 1983.

[Hegd91] Hegde, V.G., Krishnakumar, N., and Iyengar, S.S., "A New Technique for Designing Parallel Algorithms for Standard Interconnection Networks : A Parallel Sorting Illustration," *Robotics Research Laboratory - Technical Report*, May 1991.

[Hirs78] Hirschberg, D.S., "Fast Parallel Sorting Algorithms," *Communications of ACM*, vol. 21(8), pp. 657-666, 1978.

[Ho89] Ho, C.T., and Johnsson, S.L., "Dilation d Embedding of a Hyper-Pyramid into A Hypercube," *Proceedings of the Supercomputing '89*, pp. 294-303, November 1989.

[Holy87] Holyer, R.J., and Peckinpaugh, S.H., "Edge Detection Applied to Satellite Imagery of the Oceans," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 25, pp. 349-356, 1987.

[Horo81] Horowitz, E., and Zorat, A., "The Binary Tree as an Interconnection Network: Application to Multiprocessor Systems and VLSI," *IEEE Transactions on Computers*, pp. 247-253, April 1981.

[Hwan85] Hwang, K., and Briggs, F.A., *Computer Architecture and Parallel Processing*, McGraw-Hill International Editions, 1985.

[Jano85] Janowitz, M.F., "Automatic Detection of Gulf Stream Rings," *Office of Naval Research Technical Report*, vol. J8501, June 1985.

[Jone84] Jones, L., and Iyengar, S.S., "Space and Time Efficient Virtual Quadtrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*,

vol. 6, pp. 244-247, 1984.

[Karp88] Karp, R.M., and Ramachandran, V., "A Survey of Parallel Algorithms for Shared-Memory Machines," *Rept. No. UCB/CSD 88/408, Computer Science Division, Berkeley, California*, 1988.

[Kimm] Kimme, C., Ballard, D., and Sklansky, J., "Finding Circles by an Array of Accumulators," *Communications of ACM*, vol. 18(2), pp. 120-122.

[Kitt85] Kittler, J., and Illingworth, J., "Relaxation labeling algorithms - a review," *Journal of Image and Vision Computing*, pp. 206-216, 1985.

[Klee86] Kleer, J., "An Assumption-based TMS," *Artificial Intelligence*, vol. 28, pp. 127-162, 1986.

[Klin76] Klinger, A., and Dyer, C.R., "Experiments in Picture Representation Using Regular Decomposition," *Computer Graphics Image Processing*, vol. 5, pp. 68-105, 1976.

[Kris89a] Krishnakumar, N., Iyengar, S.S., Holyer, R., and Lybanon, M., "An Expert System for Interpreting Mesoscale Features in Oceanographic Satellite Images," *SPIE-Applications of Artificial Intelligence*, pp. 181-194, April 1989.

[Kris89b] Krishnakumar, N., Iyengar, S.S., Holyer, R., and Lybanon, M., "A Technique for Feature Labeling in Infrared Oceanographic Images," *Fifth International Conference on IIPS*, pp. 368-375, February 1989.

[Kris90a] Krishnakumar, N., Iyengar, S.S., Holyer, R., and Lybanon, M., "A Technique for Feature Labeling in Infrared Oceanographic Images," *Journal*

of Image and Vision Computing, pp. 112-119, May 1990.

[Kris90b] Krishnakumar, N., Iyengar, S.S., Holyer, R., and Lybanon, M., "An Expert System for Interpreting Mesoscale Features in Oceanographic Satellite Images," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 4(3), September 1990.

[Kris91] Krishnakumar, N., Hegde, V.G., and Iyengar, S.S., "Fault Tolerant Based Embeddings of Quadtrees into Hypercubes," *Proceedings of International Conference on Parallel Processing*, August 1991.

[Kurt90] Kurtz, M.J., Mussio, P., and Ossorio, P.G., "A Cognitive System for Astronomical Image Interpretation," *Pattern Recognition Letters*, vol. 11, pp. 507-515, 1990.

[Lea] Lea, S.M., and Kellar, L.A., "An Algorithm to Smooth and Find Objects in Astronomical Images," *Astronomical Journal*, vol. 97(4), pp. 1238-1246, color plates pp. 1269-1276, 1989; erratum (figures reversed), loc. cit., vol. 98, p. 736..

[Lee87] Lee, S.Y., and Aggarwal, J.K., "A Mapping Strategy for Parallel Processing," *IEEE Transactions on Computers*, vol. C-36(4), pp. 433-442, April 1987.

[Levi83] Levine, M.D., Noble, P.B., and Youssef, Y.M., "Understanding Blood Cell Motion," *Computer Vision, Graphics and Image Processing*, vol. 21, pp. 58-84, 1983.

[Lyba86] Lybanon, M., Mckendrick, J.D., Blake, R.E., Cockett, J.R.B., and Thoma-

son, M.G., "A Prototype Knowledge-Based System to Aid the Oceanographic Image Analyst," *SPIE-Applications of Artificial Intelligence*, vol. 635, pp. 203-206., 1986.

[Mala84] Malagnini, M.L., Pucillo, M., and Santin, P., "A Project for Faint Object Discrimination in Astronomy," in *Digital Image Analysis, Ed. S. Levialdi*, pp. 343-354, 1984.

[Mcke83] Mckeown, D.M., "MAPS: The Organization of A Spatial Database System using Imagery, Terrain, and Map Data," *DARPA Image Understanding Workshop*, pp. 105-127, June 1983.

[McKe85] McKeown, D.M., Harvey, W.A., and McDermott, J., "Rule-Based Interpretation of Aerial Imagery," *IEEE-Transactions on Pattern Analysis and Machine Intelligence*, vol. 7(5), pp. 570-585, Sept 1985.

[Meag82] Meagher, D., "Geometric Modeling using Octree Encoding," *Computer Graphics and Image Processing*, vol. 19, pp. 129-147, 1982.

[Moit87] Moitra, A., and Iyengar, S.S., "Parallel Algorithms for a Class of Computational Problems - A Survey," *Advances in Computers*, vol. 26, pp. 94-153, 1987.

[Moli87] Molinelli, E.J., and Flanigan, M.J., "Optimized CEOF Interpolation of the Gulf Stream," *Tech. Report #TR-392395, NOARL, Code 321, MISS 39529.*, 1987.

[Naga79] Nagao, M., Matsuyama, T., and Mori, H., "Structural Analysis of Complex Aerial Photographs," *16th International Joint Conference on*

Artificial Intelligence, pp. 610-616, August 1979.

[Naga80] Nagao, M., and Matsuyama, T., *A Structural Analysis of Complex Aerial Photographs*, New York:Plenum, 1980.

[Nass81] Nassimi, D., and Sahni, S., "Data Broadcasting in SIMD Computers," *IEEE Transactions on Computers*, vol. C-30, pp. 101-107, 1981.

[Nich87] Nichol, D.G., "Autonomous Extraction of Eddy-like Structure from Infrared Images of the Ocean," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 25, pp. 28-34, 1987.

[Pint87] Pinter, S.S., Wolfstahl, Y., "On Mapping Processes to Processors in Distributed Systems," *International Journal of Parallel Programming*, vol. 16(1), pp. 1-15, 1987.

[Pras89] Prasanna Kumar, V.K., and Krishnan, V., "Efficient Parallel Algorithms for Image Template Matching on Hypercube SIMD Machines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11(6), pp. 665-669, June 1989.

[Prep85] Preparata, F.P., and Shamos, M.I., *Computational Geometry*, Springer-Verlag, New York, 1985.

[Quin88] Quinn, M.J., *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill International Editions, 1988.

[Rank90] Ranka, S., and Sahni, S., *Hypercube Algorithms : With Applications to Image Processing and Pattern Recognition*, Springer-Verlag, New York, 1990.

[Requ80] Requicha, A., "Representations for Rigid Solids: Theory, Methods, and Systems," *Computing Surveys*, vol. 12(4), pp. 437-464, 1980.

[Requ83] Requicha, A., and Voelcker, H., "Solid Modeling: Current Status and Research Directions," *IEEE Computer Graphics and Applications*, pp. 25-37, October 1983.

[Rose80] Rosenberg, A.L., "Issues in the Study of Graph Embeddings," *Proceedings in Workshop on Graph-theoretic Concepts in Computer Science*, 1980.

[Rose76] Rosenfeld, A., Hummel, R.A., and Zucker, S.W., "Scene labeling by relaxation operations," *IEEE Transactions on System, Man and Cybernetics*, pp. 420-434, 1976.

[Rose82] Rosenfeld, A., and Kak, A.C., *Digital Picture Processing*, 1, Academic Press, 1982.

[Same81a] Samet, H., "Connected Component Labeling Using Quadtrees," *Journal of ACM*, vol. 28, pp. 487-501, 1981.

[Same81b] Samet, H., "Computing Perimeters of Images Represented by Quadtrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, pp. 683-687, 1981.

[Same82] Samet, H., "Neighbor Finding Techniques for Images Represented by Quadtrees," *Computer Graphics Image Processing*, vol. 18, pp. 37-57, 1982.

[Same83] Samet, H., "A Quadtree Medial Axis Transform," *Communications of*

ACM, vol. 27, pp. 680-693, 1983.

[Same84] Samet, H., "The Quadtree and Related Hierarchical Data Structures," *ACM Computing Surveys*, vol. 16, pp. 187-260, 1984.

[Same89] Samet, H., *Applications of Spatial Data Structures : Computer Graphics, Image Processing, and GIS*, Addison-Wesley, 1989.

[Scha77] Schachter, B.J., Lev, A., Zucker, S.W., and Rosenfeld, A., "An application of relaxation methods to edge reinforcement ," *IEEE Transactions on System, Man and Cybernectics*, vol. 7 , pp. 813-816, 1977.

[Sedm84] Sedmak, G., "Current Problems in Astronomical Image Processing," in *Digital Image Analysis, Ed. S. Levialdi*, pp. 102-133, 1984.

[Serr] Serra, J., *Image Analysis and Mathematical Morphology*, pp. 34-62, Academic Press, New York.

[Stan86] Stansfield, S.A., "ANGY: A Rule-Based Expert System for Automatic Segmentation of Coronary Vessels from Digital Subtracted Angiograms," *IEEE-Transactions on Pattern Analysis and Machine Intelligence*, vol. 8(2), pp. 188-199, March 1986.

[Tail86] Tailor, A., Cross, A., Hogg, D., and Mason, D., "Knowledge-Based Interpretation of Remotely Sensed Images," *Journal of Image and Vision Computing*, vol. 4(2), pp. 67-83, May 1986.

[Todd82] Todd-Pokropek, A., "Medical Image Processing," in *Pictorial Data Analysis, ed R.M. Haralick*, pp. 295-320, 1982.

[Tous86] Toussaint, G.T., "New Results in Computational Geometry Relevant to Pattern Recognition in Practice," *Pattern Recognition in Practice II*, pp. 135-146, Elsevier Science Publishers B.V., North-Holland, 1986.

[Vali75] Valiant, L., "Parallelism in Comparison Problems," *SIAM Journal of Computing*, vol. 4(3), pp. 348-355, 1975.

[Venk90] Venkateswar, V., and Chellappa, R., "A Framework for Interpretation of Aerial Images," *IEEE-International Conference on Pattern Recognition*, vol. I, pp. 204-206, 1990.

[Wang87] Wang, H.Q., Ritchings, R.T., and Colchester, A.C.F., "Image Understanding System for Carotid Angiograms," *Journal of Image and Vision Computing*, vol. 5(2), pp. 79-84, May 1987.

[Wu89] Wu, Q., Suetens, P., and Oosterlinck, A., "On Knowledge-based Improvement of Biomedical Pattern Recognition - A Case Study," *IEEE-Conference on Artificial Intelligence Applications*, vol. I, pp. 239-244, 1989.

[Yen91] Yen, I.L., Bastani, F.B., and Leiss, E.L., "An Inherently Fault-Tolerant Sorting," *Fifth International Parallel Processing Symposium*, April 30 - May 2, 1991.

[Zuck77] Zucker, S.W., Hummel, R.A., and Rosenfeld, A., "An Application of Relaxation Labeling to Line and Curve Enhancement," *IEEE Transactions on Computer*, vol. 26, pp. 399-403, 1977.

VITA

Krishnakumar Narayanan was born in Dharmapuri, Tamilnadu, India on July 26, 1961. He obtained his undergraduate degrees in applied sciences and electrical engineering from Madurai University, Madurai, India in 1981 and Indian Institute of Science, Bangalore, India in 1984 respectively. He completed the master's degree program in systems science at Louisiana State University, Baton Rouge, Louisiana in 1991.

His research interests include digital image processing, computer vision, artificial intelligence, parallel processing, fault tolerance, and graph embedding.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: NARAYANAN KRISHNAKUMAR

Major Field: COMPUTER SCIENCE

Title of Dissertation: NEW TECHNIQUES IN SCENE UNDERSTANDING AND PARALLEL IMAGE PROCESSING

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

06/26/91