1994

A Relaxation Scheme for Mesh Locality in Computer Vision.

Weian Deng
Louisiana State University and Agricultural & Mechanical College

A RELAXATION SCHEME FOR MESH LOCALITY IN COMPUTER VISION

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
Weian Deng
B.E., Beijing University of Aeronautics and Astronautics, 1984
M.E., Zhongshan University, 1987
December 1994

Dedicated to my parents

Acknowledgments

First and foremost, I would like to convey my sincere gratitude to my advisor, Professor S. Sitharama Iyengar, for his support, encouragement and his thoughtful criticisms and suggestions during the entire course of the research. His devotion to research will always be an inspiration to my future research. I will always be grateful for this.

It has been an enlightening experience, both scientific and personnel, to collaborate and learn from my minor advisor, Professor Suresh Rai. His problem solving approach, and his thorough knowledge in reliability and testability are truly inspiring.

Much appreciation is extended to my advisory committee, Professor Donald Kraft, Professor Bush Jones, Professor Doris Carver, Professor Jiahua Chen, and Professor Arlo Landolt, for their encouragement, intellectual support, invaluable advice, insight and patience.

I thank Professor Iyengar and the computer science department at Louisiana State University for continually supporting me as a graduate assistant. I also extend my thanks to Mr. John Benton at the Dept. of Army for providing partial financial support for this research and his useful technical suggestions. I thank Dr. Nathan Brener for his support, suggestions, and discussions in this research.

I also express my gratitude to Professor John Tyler and Professor SiQing Zheng for their guidance and help during the period of this work.

I am very grateful to Dr. Wu Wang whose encouragement and support helped me start graduate studies at LSU. I would like to thank my good friend Hla Min for

helping me proof read this document. I also thank my friends Rundong Li, Xingping Wang, Jigang Liu, Deky Gouw, Maung Maung Htay, Fenglien Lee, Todd Larsen, Daryl Thomas, Krishnamurthy Sankar who have made my stay at LSU a pleasant and a remembering one.

My love goes to my parents for the love and attention they have given to me in my life, which have been my inspiration throughout. My love also goes to my brothers Jindong and Xuejun.

And last but not the least, my wife YUYING gets my love and appreciation for her understanding, continued patience, and enthusiastic support without which I could have never reached this stage in my life.

Table of Contents

List of Tables

List of Figures

Abstract

Parallel processing has been considered as the key to build computer systems of the future and has become a mainstream subject in Computer Science. Computer Vision applications are computationally intensive that require parallel approaches to exploit the intrinsic parallelism. This research addresses this problem for low-level and intermediate-level vision problems. The contributions of this dissertation are a unified scheme based on probabilistic relaxation labeling that captures localities of image data and the ability of using this scheme to develop efficient parallel algorithms for Computer Vision problems.

We begin with investigating the problem of skeletonization. The technique of pattern match that exhausts all the possible interaction patterns between a pixel and its neighboring pixels captures the locality of this problem, and leads to an efficient One-pass Parallel Asymmetric Thinning Algorithm ($OPATA_8$). The use of 8-distance in this algorithm, or chessboard distance, not only improves the quality of the resulting skeletons, but also improves the efficiency of the computation. This new algorithm plays an important role in a hierarchical route planning system to extract high level topological information of cross-country mobility maps which greatly speeds up the route searching over large areas.

We generalize the neighborhood interaction description method to include more complicated applications such as edge detection and image restoration. The proposed probabilistic relaxation labeling scheme exploit parallelism by discovering local interactions in neighboring areas and by describing them effectively. The proposed scheme consists of a transformation function and a dictionary construction

method. The non-linear transformation function is derived from Markov Random Field theory. It efficiently combines evidences from neighborhood interactions. The dictionary construction method provides an efficient way to encode these localities.

A case study applies the scheme to the problem of edge detection. The relaxation step of this edge-detection algorithm greatly reduces noise effects, gets better edge localization such as line ends and corners, and plays a crucial rule in refining edge outputs. The experiments on both synthetic and natural images show that our algorithm converges quickly, and is robust in noisy environment.

Chapter 1

Introduction

Vision, as our most powerful sense, supplies us with a tremendous amount of information and enables us to interact intelligently with our environment, without direct physical contact. The positions and identities of objects and the relationships among them are obtained through this sense. Considerable disadvantages would result if we were deprived of vision. Since digital computers became generally available, continuous attempts have been made to provide computers with a sense of vision. These attempts lead to the establishment of an important research area — Computer Vision, which is the study of how to make computers capable of seeing things. This young and changing area is considered one of the most active research areas in computer science.

Significant progress has been made in computer vision. A successful example is its industrial applications, where the visual environment can be controlled and the vision tasks are clear-cut, such as to direct a robot arm to pick parts from a conveyor belt [39]. Optical Character Recognition (OCR) is another encouraging application that has progressed to a point where reasons for using OCR become readily apparent [69]. Robust OCR systems that can read printed materials have become realistic.

However, vision is our most complicated sense. Until now, we have only fragments of knowledge about how biological vision systems work. But one facet is clear that biological vision systems are complex. This is why many attempts to provide

computers with a sense of vision have ended with failure, especially in those areas in which ill-defined information from images needs to be extracted that even people find hard to interpret. In the current stage, no "universal" vision system exists.

Besides the theoretical reason, practically, images are made of huge amounts of data (of the order of 10^6 to 10^9 bits per image) and many operations per pixel must be performed to achieve useful transformation for computer vision tasks [8]. Computer vision tasks are computationally intensive that a parallel approach is mandatory if the system is subjected to real time constraints such as in robot vision and automatic inspection.

This dissertation focuses on the practical aspect of lower and intermediate level vision tasks such as skeletonization, segmentation, edge extraction and image restoration. The program of how to perform these tasks faster using parallel technique will be carefully examined. The remainder of this chapter overviews the tasks of computer visiondiscusses some important models of parallel computing, and discusses the need to exploit intrinsic parallelisms in vision tasks.

1.1 An Overview of Computer Vision

Computer Vision is an image understanding task that automatically builds from images of the two- or three-dimensional world, not only the descriptions of an image itself, but also the descriptions of the three dimensional scene that it depicts. These descriptions must capture the characteristics of the objects being imaged that are useful in carrying out some specific tasks. Thus, a computer vision system is always a part of a larger system that interacts with the environment, and can be considered

an element of a feedback loop that is related to sensing, while other elements of the larger system dedicated to decision making and implementation of these decisions.

The input to a vision system is an image (or several images) which is a two dimensional array of data resulting from sampling the projected instantiation of a local variable, the scene brightness function, obtained via sensing devices. The function values are either brightness values, vectors of brightness values in different spectral bands, or range data, *etc.*

The process of computer vision may be divided into six principal phases: image acquisition, preprocessing, segmentation, representation and description, recognition, and interpretation [25].

- **Image acquisition** (or sensing) is to acquire a digital image. An image sensor and the capability to digitize the sensor signals are required.

- **Preprocessing** is the process to improve images in the ways that increase the chances for success of the entire system. Typically, these include contrast enhancement, noise removal, histogramming, and thresholding. It is an explicit transformation from an input image to an output image.

- **Segmentation** divides the input image into segments, which is one of the most difficult tasks in digital image processing. The quality of segmentation is critical to the success of a vision system. For example, in OCR, the key role of segmentation is to extract individual characters and words from the background. The output of this phase is a type of raw description of the contents of input images.

- **Representation and description** is the phase that establishes a list of properties of the image segments extracted in the segmentation phase. Represen-

tation used in computer vision typically involves relational structures such as directed graphs where edges represent "relations". Each node is usually associated with a list of property values [59]. Description (or feature selection) extracts properties that provide quantitative information of interest or properties needed to distinguish one class of objects from another.

- **Recognition** is a matching process that assigns a label to an object based on the information provided by its descriptors. There are many potential methods available for the task of recognition. The most common way is to select the best match according to some form of "distance measure". Although this method is able to find an exact counterpart of the template, it is not very effective to deal with rotated, scaled or perspective transformed templates. A matching technique capable of solving these problems is called relaxation labeling. It involves a search through a space of possible distortions and transformations of the original template.

- **Interpretation** is the final phase of the whole process. It assigns meaning to an ensemble of recognized objects. This is still an essentially unachieved long term goal of computer vision, and is an important area of research in the AI field. The tasks that can be performed by current computer vision systems are so limited compared to human vision that it is not only a quantitative difference. There are fundamental shortcomings in our theoretical understanding of the visual process.

These six phases can be further grouped into three levels according to the sophistication of abstractions in their image descriptions. Figure 1.1 depicts these three levels of processes and their associated phases.

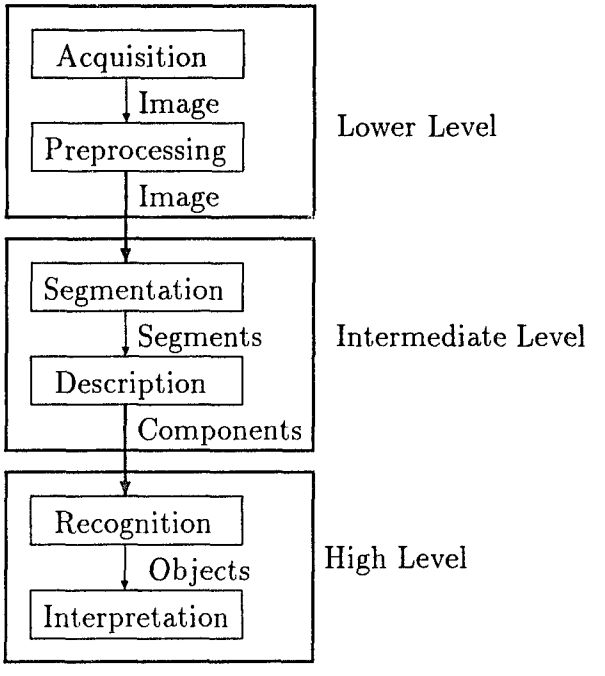


Figure 1.1: Computer vision system.

- **Lower level vision** carries out the task of transforming an input image into a more desirable output image through the above mentioned sensing and preprocessing processes. It uses no knowledge of the scene domain.

- **Intermediate level vision** consists of the segmentation phase, and the description phase. It produces from an image proper descriptions of distinguished components of the image which will be used in high level image understanding missions. Associated with it are those processes that extract properties, label segments, and characterize components.

- **High level vision** attempts to emulate cognition, capturing the meaning of the objects (recognition) and the meaning of the scene (understanding). It manipulates objects to obtain meanings.

This dissertation research focuses on the intermediate level vision problems. In this level, the main problem is labeling, *i.e.*, how to label each pixel with proper property values. Segmentation can be considered as a process to group pixels together according to their property label. Property extraction and component characterization obviously fall into this category. Skeletonization and edge detection are two of the typical tasks of intermediate level vision.

1.1.1 Skeletonization

A common approach to describe the shape of an image is to reduce it into a graph [25]. This graph is obtained by approximating the skeletons (or the medial axises) of image shapes. The skeletonization is an important step for many computer vision tasks such as OCR (in which skeletons are used to extract vector representations of the input characters) [3], route planning (where the skeleton of a map speeds up the search) [4], fingerprint recognition [55] and biomedical imaging [21]. The number of pixels to be searched during high-level vision process, hence its complexity, can be greatly reduced by shrinking an image into a unitary skeleton. Skeletonization problem can be solved by medial axis transformation (MAT) proposed by Blum [6]. For a region R with border B, the distance from a pixel p in R to the border B is defined as the distance from p to the nearest neighbor in B. A pixel belongs to the medial axis of R, if p has more than one nearest neighbor. However, the direct implementation of this method is unacceptable computationally because it potentially calculates distance from every interior pixel to every boundary pixel. Many algorithms have been developed to improve computational efficiency. Techniques that mark the medial axis pixels in their neighbor context is a favorable solution and will be discussed in detail in the next chapter.

1.1.2 Edge Detection

Segmentations generally are performed based on two basic properties of gray intensity: similarity and discontinuity. The basic approach for the first category is based on thresholding, region growing, and region splitting and merging. For the second category, the basic idea is to divide an image by detecting abrupt changes in gray intensity. These abrupt changes are generally isolated noises or edges between two regions with distinct gray intensities.

Edge detection has attracted a great deal of research effort because the resulting descriptions significantly affect the quality of higher level processes. Four approaches have been used in edge detection: template matching, edge fitting, statistical edge detection, and gradient and difference based filtering [83].

- **Template matching** has a template base of ideal edges setting at all orientations. All regions in an image with the same size are matched against these templates. Corresponding templates are located by evaluating the degree of matches.

- **Edge fitting** uses a mathematical model with a small number of well defined parameters to describe ideal edges. The detection process evaluates how closely the model fits the neighborhood of every pixel and obtains the values for the parameters of the model so that a "distance measurement" can be minimized.

- **Statistical edge detection** considers a small region (or window) in an image and views an edge as a boundary between two nonhomogenous subregion. Statistical hypothesis testing is then used to choose from the two hypotheses that:

H0: The image values on both sides of a line are homogenous;

H1: The image values on both sides of a line are nonhomogenous.

- **Gradient based filtering** is the most widely used approach. It treats an image as an intensity function that is continuous and differentiable. Edges correspond to rapid changes in intensity. Therefore, the gradient function of the image which represents the speed of change, will have maximum values in the positions where edges exist. Edges are detected by locating all maxima points of the gradient function. An alternative method is to use second order derivatives, the gradient of a gradient. The second order derivative of an image has zero-responses at edge points, and responds to the two sides of the edge with different signs (positive *versus* negative). This is called zero-crossing. Second order derivatives give more accurate localization. However, derivatives can amplify any error component. Thus, second order derivatives tend to be much noiser than gradient operators.

The approach taken in this research is to employ neighbor context to enhance edges and to suppress noise. Each pixel in an image is given a label ("edge" or "non-edge"). Labels for the neighborhood pixels are taken into account in the process of label assignment.

In this section, we outlined the general procedure of computer vision. Most of the mentioned processes are computationally intensive due to the amount of data to be processed and the time-consuming nature of the algorithms. A typical size of digital images is 1024×1024 pixels, and each image needs to go through several layers of time consuming processing, from numeric operation to symbolic calculation. Thus,

the throughput required for computational demand is enormous. This is even more in the case of a real-time vision system. A search for parallelism is inevitable.

1.2 Parallel Processing

Despite the rapid progress of state of the art computer components, the performance increase rate has always lagged behind the increasing demand of computer users. Parallel processing came into being when designers looked towards replicating the current technology rather than advancing the technology itself. Instead of a single function unit, a number of such processing units (PEs) could be incorporated into the same computer to operate concurrently, so that the performance of the system can be increased. Parallel processing is a kind of information processing that emphasizes the concurrent manipulation of data elements belonging to one or more processes, solving a single problem. A parallel computer is a computer designed for the purpose of parallel processing [63]. The primary sources of references for the following material are from Offen [59] and Quinn [63].

A variety of mechanisms may be used to achieve parallel computing. The three major architecture configurations are pipeline vector computers, array processors and multiprocessor systems.

Pipeline vector computers exploit temporal parallelism. Successive computations on vector operands are executed in an overlapped fashion. A computer's computational structure is segmented into consecutive units, and program processes are decomposed into temporally overlapping subprocesses that have to pass sequentially through each unit. For example, the execution of an instruction by a computer can be divided into 4 consecutive steps: instruction fetch (IF), instruction decoding

(ID), operand fetch (OF) and execution (EX). In a pipeline computer, these four steps are performed by four different function units that are sequentially linked. When vectors are applied, these four function units perform their own jobs on different vector elements at the same time. Therefore, the computer can produce a new result for every function unit cycle. Figure 1.2 shows the dynamics of a pipeline.

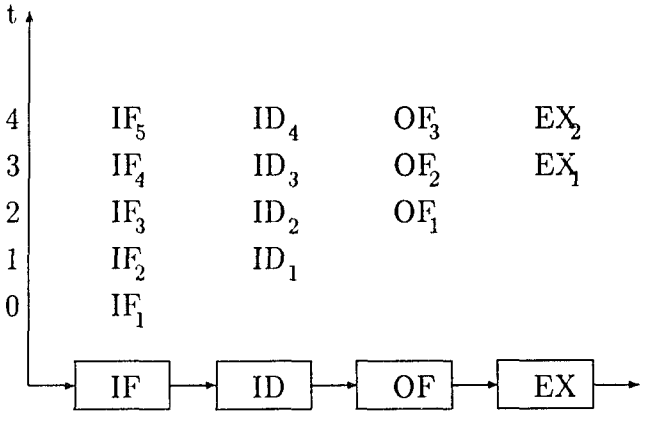


Figure 1.2: Dynamics of a pipeline computer.

Array processors use a number of processing units (PEs) to perform vector or matrix operations concurrently, and to exploit the inherent spatial parallelism. An array processor (or SIMD-array processor) is a synchronous parallel computer with multiple parallel operated PEs under the control of a single central control unit. Each PE has its own Arithmetic Logic Unit (ALU) and its own memory. This memory can store data only. Instructions are stored in the main memory which is controlled by the control unit. Communications between PEs are implemented through an Inter-PE connection network (See Figure 1.3). All the PEs execute the same instruction at the same time in a "lockstep" mode. The single control unit supplies the sequence of PE instructions and PE addresses. Instructions are provided to activate only those PEs that satisfy a program-specified condition and to deactivate those PEs that don't satisfy the condition.

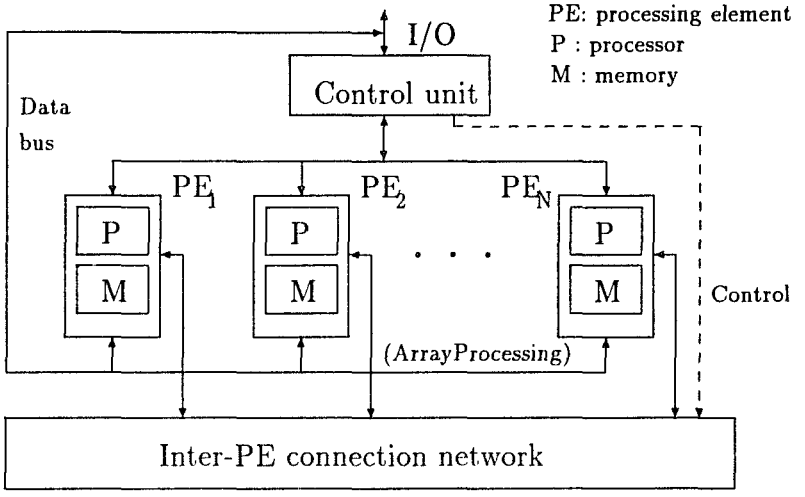


Figure 1.3: Array processors.

The difference between a **multiprocessor system** (MIMD) and a array processor is that a multiprocessor system has no central control unit to synchronize the PEs (See Figure 1.4). It exploits asynchronous parallelism. PEs are more independent. They have their own instruction decoder and their memory can store both data and instructions. In other words, each PE is a unique computer by itself. However, the PEs do share some resources such as memory and peripherals and they interact during the course of their execution. Inter-PE communications are implemented in two ways:

- **Inter connection network**: Communications between PEs are implemented by message passing between PEs through a network. This type of system is called loosely-coupled.

- **Shared memory**: It has a common address space with which all PEs can have access. Processors communicate by reading from and writing to the same address in the shared memory. This type of system is called tightly coupled.

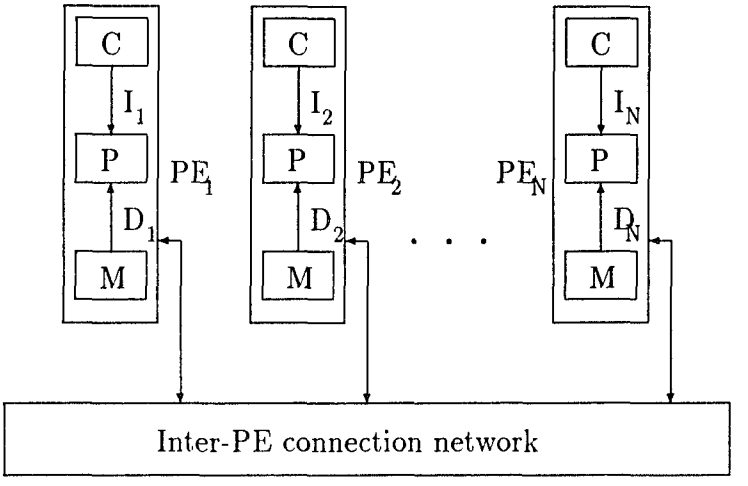


Figure 1.4: A multiprocessor system.

Multiprocessor systems introduce a new philosophy of parallel programming in which a problem is decomposed into a number of distinct tasks which are distributed among the processors for concurrent processing. They improve throughput, reliability, flexibility and availability of computer systems. However, they are relatively sophisticated.

In both array processor and loosely-coupled multiprocessor systems, inter-PE communications are implemented through an inter-PE network. The time spent on PE interactions is not ignorable. Thus, how to design a network that minimizes both the complexity of the network and the time for inter-PE communication is an important research issue. Many processor organizations (methods of connecting PEs) have been proposed. Butterfly, mesh, and pyramid are three of the popular organizations that are used in computer vision.

Butterfly

A butterfly network consists of $(k+1)2^k$ nodes divided into $k+1$ rows (or ranks). Each rank contains $n = 2^k$ nodes each (Figure 1.5). The ranks are labeled 0 to k.

The rank 0 and k are sometimes identical to form cycles, leaving each node with four connections.

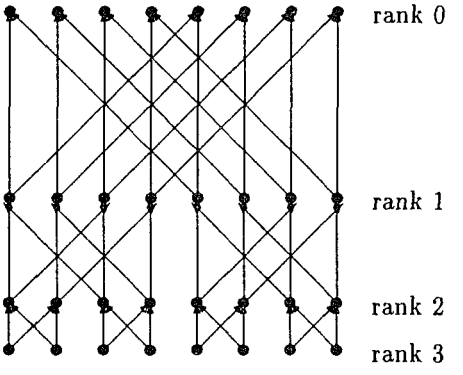


Figure 1.5: A butterfly network.

Let (i, j) denote the jth node on the ith rank, $0 \leq i \leq k$ and $0 \leq j < n$. Node (i, j) on rank $i > 0$ is connected to two nodes on rank $i - 1$; node $(i - 1, j)$ and node $(i - 1, m)$, m is the integer obtained by inverting the jth most significant bit in the binary representation of j. If node (i, j) is connected to node $(i - 1, m)$, then node (i, m) is connected to node $(i - 1, j)$. This type of "butterfly" pattern gives the network its name. The widths of butterfly wings increase exponentially when the ranks decrease. Butterfly networks are the best networks to implement the Fast Fourier Transformation which is a widely used technique in image processing.

Mesh

A mesh is a network of PEs that are arranged into a q-dimensional lattice. Communications are allowed only between those q neighboring nodes. Figure 1.6 is an example of a two-dimensional mesh. Mesh networks do have the disadvantage that data routing requirements often prevent the development of $O(log^k n)$ parallel algorithms. Routing data from one side to the other side in an n PEs network requires $O(\sqrt{n})$ time. However, for problems that have "mess-local" property, data

are never transmitted more than a fixed distance from the starting location [77]. So, mesh connected network is the best choice. Many lower level and intermediate level computer vision problems are mess-local.

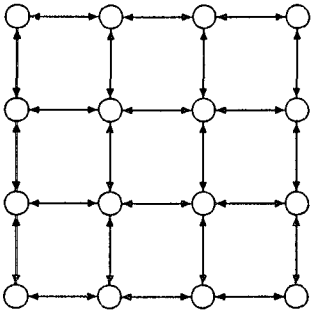


Figure 1.6: A mesh connected network.

Pyramid

Primary designed for computer vision, a pyramid network of size p is a complete 4-ary rooted tree of height $log_4 p$ with additional interprocessor links to connect the processors in each level into a two dimensional mesh. A pyramid of size p has a total of $\frac{4}{3}p - \frac{1}{3}$ PEs and p PEs at its base, a two-dimensional mesh network. Figure 1.7 shows a pyramid of size 16. Each interior PE connects to nine other PEs: one parent, four mesh neighbors, and four children.

The hardware cost of a pyramid machine is more expensive than a mess machine. It has one third more PEs and more than double the connections. However, it allows fast communication between any two PEs, specifically in $O(\log n)$, and is perfectly designed for adopting a bottom-up (data driven) image analysis, top-down (model driven) image analysis or a hierarchical processing mode [81].

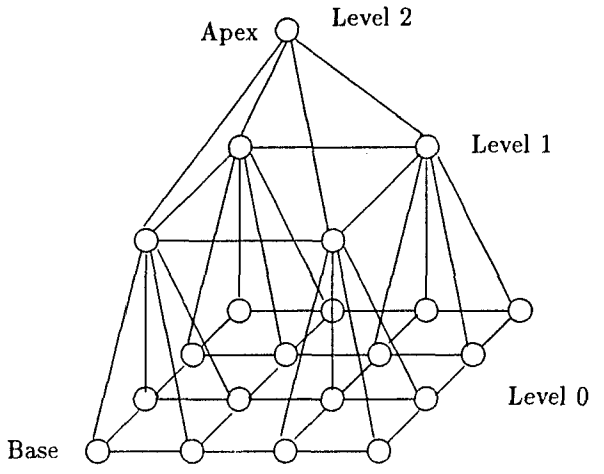


Figure 1.7: A pyramid connected network.

1.3 Parallelism in Computer Vision

There are two fundamental issues involved in designing parallel algorithms for a given application: selecting an appropriate architecture and exploiting the inherent parallelism.

For computer vision application, a pyramid is a flexible organization especially for high level vision tasks. Each PE in this network can access all image data quickly in $O(\log n)$ time. However, pyramid computers are costly. Most intermediate level and lower level vision tasks operate on the input image. An image is a two dimensional array which can be easily stored in an array processor.

Many of these vision tasks, such as noise removal and image smoothing, can be implemented by identical local operators applying to all image pixels. These tasks are suitable to be implemented on mesh-connected SIMD machines which have local interconnections. Algorithms designed on this type of organization have become increasingly significant in computer vision because of three reasons [59]:

1. The requirement for real-time processing often precludes the use of serial organizations.

2. The advent of VLSI has made it feasible to build large parallel processing networks, particularly ones with a uniform structure, with only local neighborhood connections, and with uniform simple type of PEs.

3. Studies of the structure of biological vision systems have shown the primate visual cortex to contain neural networks that can be modeled as parallel architectures with the above properties of locality, uniformity and simplicity of individual processors.

However, not all computer vision tasks have locality. The higher the level they belong, the less locality they have. Then comes a question whether these tasks can be reformulated, so that they are suitable for a mesh-connected SIMD machine.

To exploit parallelism, two approaches have been recommended: functional parallelism and data parallelism.

Functional parallelization methods look into the functional structure of the particular given problem, trying to partition it into many relatively independent functional units which can be executed concurrently. Different function units may perform different jobs. The problems with these methods are that they are problem dependent and cannot be easily generalized.

Data parallelism, on the other end, may be appled to variety of problems. It tries to formulate a given problem in a way that the solution is obtained by the cooperation of a set of data-driven function units that are identical to all the data of the type. For example, in computer vision, many processes are performed identically

to all pixels in an image. Data parallelism is more suitable for developing parallel hardware by current VLSI techniques.

1.4 Contributions

In this dissertation, we exploit parallelism for computer vision problems that have locality characteristics. These problems can be divided into two categories: those that are explicitly localized and the others whose localities have to be extracted from global interactions. Both cases are studied in this research. Figure 1.8 depicts the contributions pictorially.

An example of explicitly localized problem, skeletonization, is studied in detail and a new parallel thinning algorithm is proposed. This new algorithm ($OPATA_8$) uses one-pass technique, asymmetric pattern match method, and decision tree structure. It implements 8-distance skeletonization, runs faster, and obtains skeletons with nice topological structure, unitary 8-connected, and good noise resistance.

This parallel thinning algorithm is an important component in a route planning system — PIMTAS, a co-operative project between LSU and Topographical Engineering Center (TEC) of U.S.Army. The distinguished feature of PIMTAS is its hierarchical structure. Route planning is implemented in three steps: extract the graph structure of a map, plan grid level paths for adjacent graph nodes, and search for paths from source point to target point using graph level planner. This unique structure gives the system the flexibility to find optimal paths and the possibility to achieve real-time response for large maps. AI techniques are used to implement the planning and skeletonization technique is employed to extract graph structures.

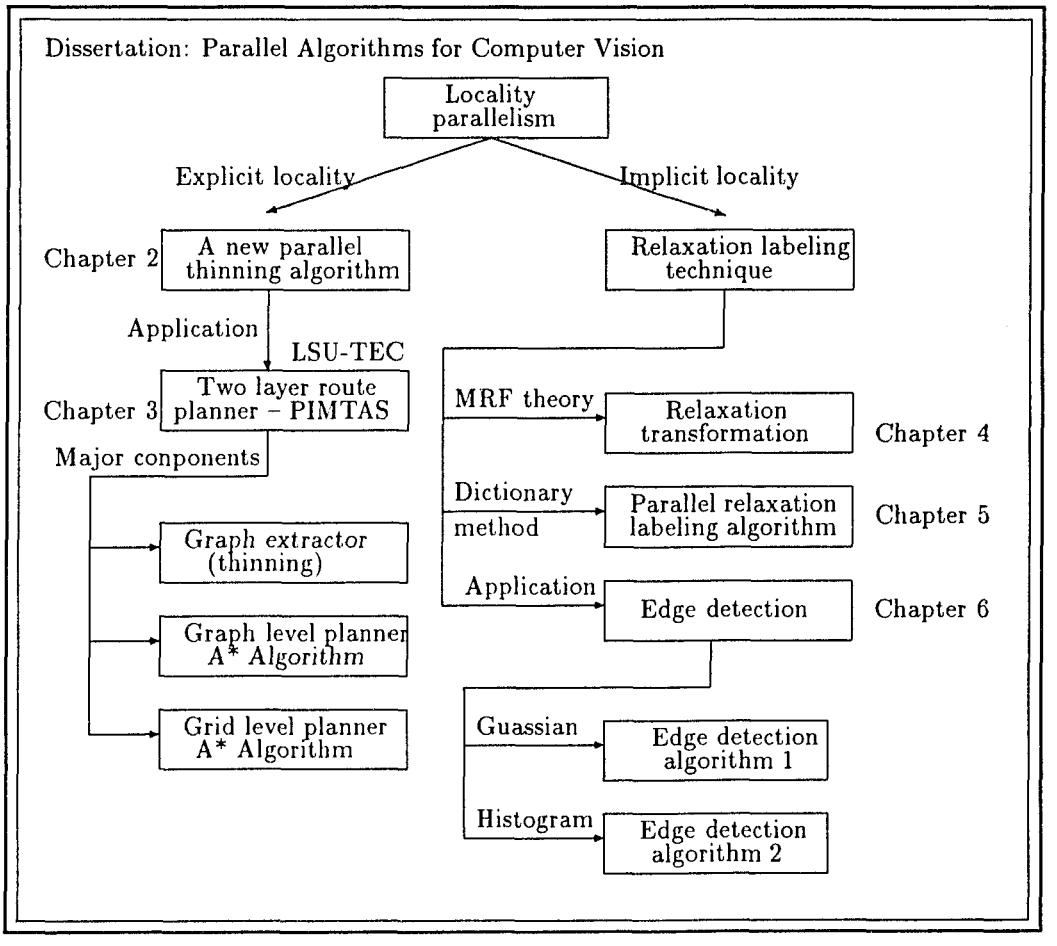


Figure 1.8: The organization of the paper.

For implicit locality in Computer Vision, we proposed a probabilistic relaxation labeling scheme to exploit parallelism from local interactions in neighboring areas and to represent them effectively. The proposed scheme consists of a transformation rule and a dictionary construction method. The non-linear transformation rule is derived from Markov Random Field theory. It efficiently combines evidences from neighborhood interactions. The dictionary construction method provides an efficient way to encode these localities. The algorithm developed using this scheme has the characteristics of mesh local and can be efficiently implemented by mesh connected array processors.

Edge detection is an important process in computer vision applications. Edge maps are sufficient to conduct higher level processes such as motion analysis and object recognition. A variety of edge detectors have been proposed. Most of them perform reasonably well for simple noise free images, but tend to fail for noisy images. When edge detection is treated as a relaxation labeling problem that each pixel has a label set of several "edge labels" and a "non-edge label", the correlation of a pixel's label and its neighboring pixels is an efficient tool to improve edge detection. In the last part of this dissertation, we apply our probabilistic relaxation labeling scheme to edge detection to greatly reduce noise effects and to obtain better edge localization such as line ends and corners. Two edge detection algorithms have been developed which are based on the Gaussian distribution assumption and the histogram of intensity changes, respectively. Our study shows that though applying contextual information efficiently through the update procedure can eliminate ambiguities from initial label assignment errors, there is no guarantee that all the errors can be corrected if there are too many of them, which is the case in Gaussian distribution

based algorithm. The experiments on both synthetic images and natural images show that our algorithms converge quickly, and are robust in noisy environment.

1.5 Outline of the Dissertation

One-pass parallel thinning algorithm is the focus of Chapter 2. In this chapter, we first introduce the problem of skeletonization, the existing approaches, and the advantages and disadvantages of one-pass parallel algorithms. We then develop a new one-pass asymmetric parallel thinning algorithm that implements 8-distance skeletonization. We compare the results of our algorithm with two important existing algorithms and show that our technique yields better results in a shorter time.

In Chapter 3, we describe the Predictive Intelligent Military Tactical Analysis System (PIMTAS). We discuss the underlying hierarchical structure of the system, detail the description of the various modules in PIMTAS, and study the existing approaches for route planning. We also examine the implementation issues of thinning algorithm for this application and show the speedups for different approaches.

Our probabilistic relaxation labeling scheme as a tool to exploit mesh-local parallelism is introduced in Chapter 4. A survey on relaxation labeling is given. After defining the problem of relaxation, we present the theory of Markov Random Field (MRF) and use it to derive the transformation $\mathcal{T}$ for our scheme. The properties of this scheme are then examined.

Chapter 5 develops an efficient relaxation labeling algorithm. Maximum Entropy Estimation technique is first used to estimate the joint conditional probabilities for transformation $\mathcal{T}$. The method to find Influential Configuration Set (ICS) are

presented along with the method to assign the configurations probabilities. The construction of dictionary of ICS leads to our new probabilistic relaxation algorithm.

In Chapter 6, a case study that applies the relaxation labeling method to edge detection is presented in detail. The problem of initial label assignments is discussed. Two edge detection algorithms are developed — one of them based on Gaussian model and the other on histogram. A detailed performance comparison with two existing edge detection algorithms is also included.

We summarize the results in Chapter 7. Applications of the techniques and algorithms developed in this dissertation are outlined. The need for further research in the area of mesh-local parallelism is discussed.

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

Chapter 2

Parallel Image Skeletonization

In this chapter, the problem of binary image skeletonization is studied. Skeletonization is widely used in many image processing applications, such as optical character recognition, chromosome analysis and military route planning[4]. It provides a convenient and condensed representation of image object information. Skeletons of objects can preserve topological information of the original objects[4] and reduce storage requirements [50].

The algorithm that will be discussed is an good example of parallelizing an image processing task by localization. It is also a critical part of a mobility route planning system — PIMTAS (Predictive Intelligent Military Tactical Analysis System). The new one-pass parallel thinning algorithm inherits the asymmetrical feature of Wu and Tsai's algorithm $OPATA_4$[86]. However, the use of 8-distance, or chessboard distance, to approximate Euclidean distance not only improves the quality of the resulting skeletons, but also speeds up the thinning procedure. Our algorithm has been implemented sequentially and has been compared to both Wu and Tsai's $OPATA_4$ and Zhang and Suen's TPTA[91].

Section 2.2 discusses the advantages and disadvantages of both sequential and parallel skeletonization algorithms. The ideas in parallel thinning algorithms, especially, in Wu and Tsai's asymmetric one-pass algorithm $OPATA_4$ are reviewed in Section 2.3. Based on the observations on $OPATA_4$, Section 2.4 presents in

detail our new parallel algorithm $OPATA_8$. The experiments and the comparisons of $OPATA_8$ with other thinning algorithms will be discussed in Section 2.5.

2.1 Preliminaries

A binary image is defined as a matrix P where each element(pixel) is either 1(black) or 0(white). Objects in images consist of black pixels.

Neighbors: For a pixel p in image P, the **8-neighbors** of p are defined to be the eight pixels adjacent to p (p_0, p_1, ..., p_7 in Figure 2.1(a)), and denoted by $\mathcal{N}_8(p)$. Also, p_0, p_2, p_4, and p_6 are referred to as the set of **4-neighbors** of p, $\mathcal{N}_4(p)$. p_8 and p_9 are the two pixels that will be used to introduce asymmetry.

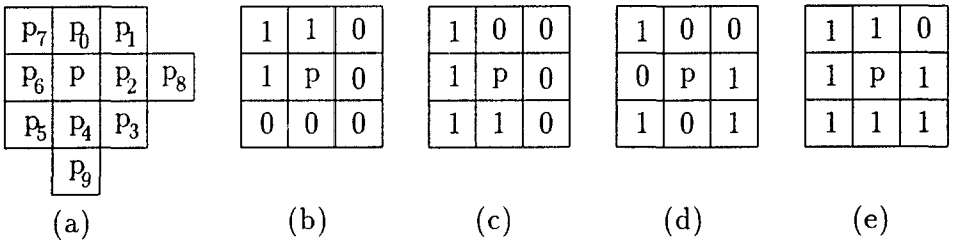


Figure 2.1: 4(8)-neighbors of a pixel

Distance: Between two pixels $p(x_p, y_p)$ and $q(x_q, y_q)$, where x_p, y_p, x_q, and y_q are coordinates, the **8-distance**, or the chessboard distance, is defined as

$$d_8(p,q) = max(\mid x_p - x_q \mid, \mid y_p - y_q \mid)$$

and the **4-distance**, or the city block distance, is

$$d_4(p,q) = \mid x_p - x_q \mid + \mid y_p - y_q \mid$$

Connectedness: A skeleton is considered to be **4(8)-connected** if between any two black pixels p_0 and p_n there exists a path $p_0p_1 \ldots p_{i-1}p_ip_{i+1} \ldots p_n$ such that p_{i-1} is a 4(8)-neighbor of p_i for $1 \leq i \leq n$.

Neighbor sequence: A sequence of 8-neighbor pixels of p $(p_i, p_{i+1}, \ldots, p_{i+n})$ is called **neighbor sequence** of p if $p_i, p_{i+1}, \ldots, p_{i+n}$ are 4-connected in a clockwise order around pixel p and they are all black pixels. For example, Figures 2.1(b) and 2.1(c) contain neighbor sequences $p_6p_7p_0$ and $p_4p_5p_6p_7$, respectively, while in Figure 2.1(d), $p_2p_3p_5$ is not a neighbor sequence because p_3 and p_5 are separated by a white pixel p_4.

Simple path: A path $\mathcal{P}$ is called a **simple path** if the removal of any pixel from the path except end pixels will violate the 4(8)-connectedness of the path when 4-distance(8-distance) is used.

Edge pixel: An **edge pixel** is a black pixel that one of its 4-neighbors is white pixel.

Convex corner pixel: a **convex corner pixel** is a black pixel such that two of its 4-neighbors p_i and p_{i+2} are white pixels. Figure 2.1(b) is an example. A convex corner pixel is an edge pixel.

Concave corner pixel: A **concave corner pixel** is a black pixel such that only one of its diagonal 8-neighbors is white (all other 8-neighbors are black). Figure 2.1(e) is an example.

End pixel: An **end pixel** is a black pixel such that only one of its 4(8)-neighbors is a black pixel when 4-distance (8-distance) is used.

Contour: A **contour** is the set of edge pixels.

Interior pixel: An **interior pixel** is a black pixel that does not belong to any contours.

Interior: The **interior** is the set of interior pixels of an object.

Thin object: A **thin object** is an object that has no interior pixels. A thin object is not necessary a simple path. For example, lines that are two pixels wide are not simple paths.

2.2 Sequential *versus* Parallel

The skeletonization problem has been extensively studied in the last twenty years. Many methods have been proposed in the literature. These methods can be classified into two categories: distance transform methods [1, 2, 79, 89] and parallel thinning methods [9, 38, 54, 76, 86, 91].

Distance transformation was proposed for binary pictures in the Euclidean plane by Blum[5]. The idea is that a fire line, which propagates with constant speed from the contour of an object to its inside, will meet at quench points that form the skeleton. Many discrete approximations of this fire propagation technique have been reported. Generally, these approximations were implemented by explicitly tracing the object's boundary pixels to avoid going back to the area that had already burned [2, 79]. Xia[89] further refined the technique to construct the next fire front during the tracing of the current fire contour. These algorithms perform efficiently in a sequential machine, and the skeleton obtained can be used to recover the original objects. However, these algorithms have some serious drawbacks. The tracing technique has sequential feature and cannot be implemented in parallel. Due to the properties of discrete space, they can not guarantee that the topology of the

objects will be preserved. In addition, the skeletons obtained are sensitive to local variations and noise [89].

The other type of algorithm is called parallel thinning algorithm which implements the fire line implicitly. They are developed by investigating the locality of skeletonization. Whether a pixel belongs to the contour of an image object can be determined locally by examining the pixel values of its neighborhood. Thus local neighborhood patterns are used as sufficient conditions to determine whether a pixel is a contour pixel that can be removed. This type of algorithm iteratively deletes contour pixels that are removable. The patterns selected should guarantee the connectivity of the resulting skeleton.

There are many algorithms of this type that are based on the same principle. The differences are usually the sets of patterns they use. The advantage of this type of algorithm is that one can custom design the patterns to delete certain end points and thereby omit certain details in order to make the skeleton easier to interpret. This type of algorithm can capture topological information on objects.

Some people have argued that the majority of parallel thinning algorithms are time-consuming compared to distance transformation methods. While it is true that parallel thinning algorithms have a complexity proportional to the size of images and the maximal thickness of objects in the images if implemented sequentially [89], this is misleading because these algorithms are constructed to take advantage of parallel. It is primarily the sequential counterparts that can be time-consuming. These countparts can be made more efficient by adopting the same contour tracing technique employed by distance transformation approaches. However, the drawback of this type of skeletonization is that it may not preserve as much detail as distance transformation methods. Therefore, the resulting skeletons may not be recoverable.

Here, we focus on parallel thinning algorithms. For more detailed discussions of distance transformation methods, one can refer to Xia[89].

2.3 Parallel Thinning Algorithms

Skeletonization is usually performed by iteratively removing edge pixels along the contour of image objects. The pixels removed must satisfy the following three criteria:

1) An end pixel is never deleted

2) Connectedness is not violated

3) Excessive erosion should not occur.

These criteria can be imposed to a pixel in most cases by checking its 8-neighbors. Most of the proposed parallel thinning algorithms differ only in the way that they conduct the test to meet these criteria[56]. For the thinning algorithms before 1984, Naccache and Shinghal[56] have given a detail review and comparison. In the past ten years, the most widely cited algorithm is that of Zhang and Suen[91]. This algorithm is called a two pass algorithm because in each iteration, there are two passes, or sub-iterations. In the first pass, the conditions to remove pixel p are the followings:

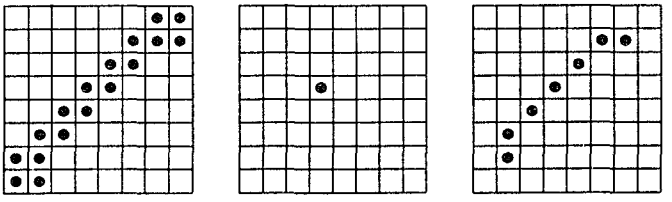
1) It has a neighbor sequence of length between 2 and 6

2) It is a north-west edge pixel or a south-east convex corner pixel.

The second pass deletes the south-east edge pixels and the north-west convex corner pixels that satisfy the condition 1 above. The use of a two pass iteration imposes a

bias in which north-west edge pixels and south-east convex corner pixels are preferred over other south-east edge pixels. This type of bias avoids excessive erosions.

Although Zhang and Suen's algorithm is good at connectivity and contour noise immunity, there are some disadvantages that need to be addressed. First, the algorithm still suffers from excessive erosions in some extreme conditions, such as two-pixel-wide diagonal lines (See Figure 2.2(a)). As shown in Figure 2.2(b), when Zhang and Suen's algorithm is used to thin the image in Figure 2.2(a), the original topological structure is totally destroyed. Second, though two-pass algorithms run faster than four-pass algorithms by reducing the time for scanning the image, the amount of time needed to scan white pixels (which is a waste) is still significant. When we applied Zhang and Suen's algorithm to thin mobility maps in our route planner application, we noticed that, in some cases, half of the running time was spent on scanning white pixels. Furthermore, the resulting skeletons from Zhang and Suen's algorithm are not of unitary thickness.



(a) is the input image; (b) is the degraded result from Zhang and Suen's TPTA; and (c) is from OPATAs.

Figure 2.2: The erosion problem.

Several attempts have been made to overcome these drawbacks. Lü and Wang[51] restricted neighbor sequences to a length greater than three instead of greater than two. The improved algorithm preserved structures better, but degraded the noise

suppression feature of Zhang and Suen's TPTA. Holt *et al* [38] modified Zhang and Suen's algorithm from two-pass to one-pass. In general, the number of overall passes needed to thin an image was reduced. However, they used a neighbor region of 5×5 that contains 25 pixels, which is much bigger than a 3×3 neighbor region, and as a result, the algorithm turned out to be consistently slower than Zhang and Suen's algorithm. Another problem is that Holt's algorithm didn't guarantee the preservation of original patterns[54]. Mendel[54] modified both Zhang and Suen's and Holt's algorithms to preserve the original pattern better in approximately the same amount of running time.

2.3.1 One-pass Parallel Thinning Algorithm

Recently, a lot of effort has been devoted to developing one-pass parallel thinning algorithms. The paper by Holt et al was one of the attempts. Chin and Wan[9] came up with a more efficient one-pass algorithm that mainly used 3×3 operators. The algorithm used eight 3×3 thinning patterns to remove edge pixels, and two restoring patterns (a 1×4 and a 4×1) to preserve continuity. Eight more patterns were employed later to trim noise effects. However, a major problem with this algorithm is that it generates biased skeletons. Convex corners are removed faster than concave corners, as shown in Figure 2.7(b). Linear objects with a sharp turn generate skeletons that do not run along the Euclidean medial axis at the turn.

Wu and Tsai[86] designed a new set of matching patterns that eliminated the need to distinguish between thinning patterns, restoring patterns, and trimming patterns. The set of fourteen patterns are shown in Figure 2.3(cited from [86]). In these patterns, x indicates that the pixel can either be 0 or 1, y indicates that in this pattern at least one of the y's is 0, and c indicates a current contour pixel that may

be removed. Each black pixel whose neighbor area matches one of these patterns will be removed in the current pass.

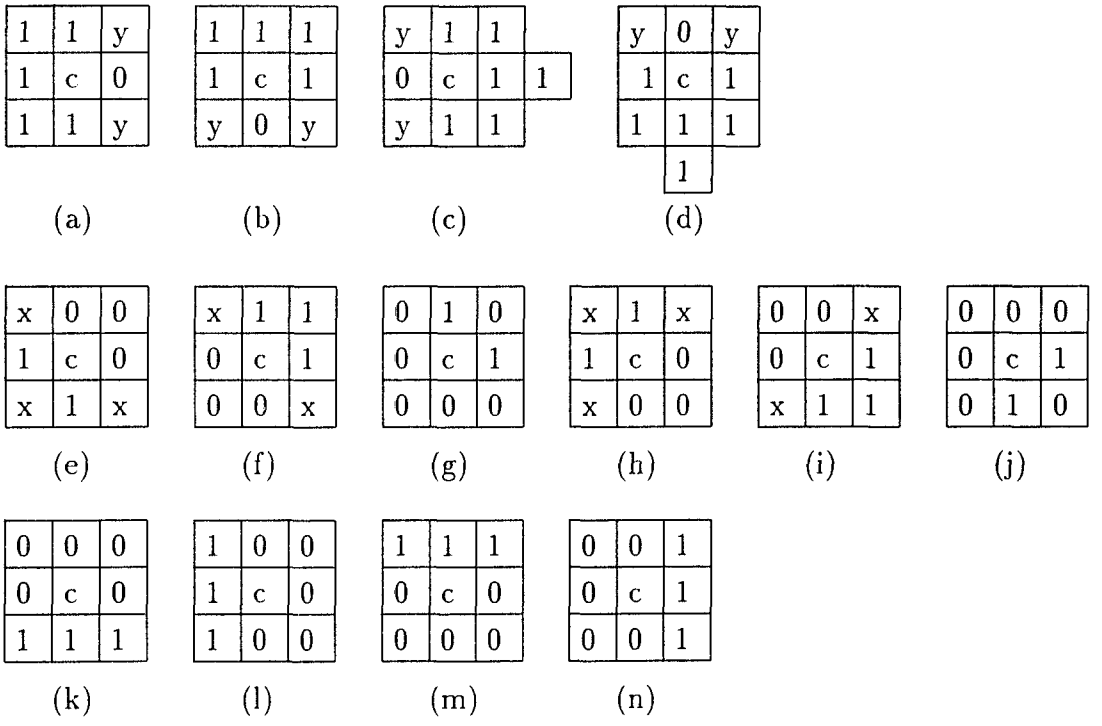


Figure 2.3: Wu and Tsai's fourteen thinning patterns.

This set of patterns was derived from the idea of asymmetry. When an object is thinned to a thin object (two pixels wide, generally), the pixels on one side of the object are removed according to preset preferences, while the pixels on the other side are retained. For example, patterns (a) and (c) are used to thin vertical lines. When a vertical line is not a thin object, both patterns will be used to thin the contour on both sides of the object, with pattern (a) thinning the right side and pattern (c) the left side. When the object becomes a thin object, pattern (a) will continue to thin the contour pixels on the right side, while pattern (c) will leave the contour pixels on the left side intact. In the same way, patterns (b) and (d)

will preferentially remove contour pixels on the top over those on the bottom for horizontal lines. Patterns (e) and (f) deal with diagonal contours that go from the top left corner to the bottom right corner, where pixels on top right side are thinned while those on the bottom left side remain. Pattern (g) is a complement pattern for pattern (f), since the condition in pattern (f), that the top right pixel p_1 must be black, excludes pattern (g). Patterns (h), (i), and (j), in a similar way, thin the diagonal contour that goes from the top right corner to the bottom left corner. As an example, in Figure 2.2, patterns (e), (f), (h) and (i) are used to get the resulting skeleton in Figure 2.2(c) from the input image in Figure 2.2(a). Patterns (k), (j), (l) and (n) were designed to remove noise.

The algorithm presented by Wu and Suen is a pattern match algorithm. In parallel, all black pixels are checked against the fourteen patterns. When any pattern matchs a pixel's neighbor setting, the pixel is whitened out (changed from value 1 to 0). This procedure continues until no more pixels can be thinned.

The advantages of Wu and Tsai's algorithm are its quickness and uniqueness. To my knowledge, it is one of the fastest parallel algorithms currently in use. The algorithm is unique. It treats all fourteen patterns in the same way. It is also noise insensitive. It produces perfect 8-connected skeletons, and the resulting skeletons are quite isotropic in terms of city-block distance. However, this algorithm inherits the major problem from which Chin's algorithm suffers, namely that the resulting skeletons are biased, cutting corners. As a result, these skeletons do not preserve the topological structures of original objects as well as those from Zhang and Suen's algorithm.

2.4 An 8-Distance Parallel Thinning Algorithm $OPATA_8$

Both Chin's algorithm and Wu and Tsai's algorithm suffer from the different thinning speed at convex corners and concave corners. Their results in deterioration of the skeleton's topological features. The problem comes from their discrete approximation of the Euclidean distance. Both algorithms implemented the city-block distance(4-distance) approximation d_4. The city-block distance from a convex corner pixel to a background area is always 1. Thus, the pixel is always considered to be an edge pixel and may be thinned in the current pass. A concave corner pixel, on the other hand, has a city-block distance of 2 from the background area, and can become an edge pixel only in the next pass. Figure 2.4(b) is an example where a convex corner is removed in the first pass and a concave corner is removed in the areas pass. Figure 2.4(a) is the input image with five convex corners and one concave corner. In Figures 2.4(b) and 2.4(c), a number in a pixel square indicates the pass in which the pixel is thinned.

This problem can be solved if we adopt the chessboard distance(8-distance). Both convex corner pixels and concave corner pixels have a chessboard distance of 1, and are thinned at the same speed. For example, in Figure 2.4(b), all edge pixels and the concave pixel have d_8 equal to 1, and are removed in one pass. The result is a prefect skeleton (Figure 2.4(c)).

The patterns used by Wu and Tsai's $OPATA_4$ remove edge pixels including convex corner pixels. However, concave corner pixels are not edge pixels and thus cannot be removed by those 14 patterns. In order to modify $OPATA_4$ to develop an algorithm that has the chessboard distance approximation, we need to find a way

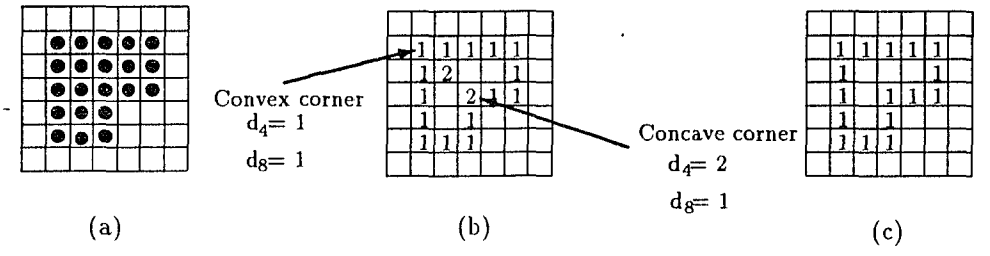


Figure 2.4: The d_4 and d_8 approximations of the Euclidean distance.

to recognize concave corner pixels and find the condition to preserve connectedness when these pixels are removed.

2.4.1 Locating a Concave Corner

In order to recognize a concave corner pixel, 3×3 patterns are not the correct choice. The 3×3 patterns that could be used to locate a concave corner are those four patterns obtained from the rotation of Figure 2.5(a). However, these patterns invite some problems. These four new patterns enhance noise if they are combined with the fourteen existing patterns. Figure 2.5 is an example. In Figure 2.5(c), pixels with black dots are part of a rectangular object with a white noise pixel at the center of its bottom. Applying the fourteen patterns in Figure 2.3 along with the four 3×3 patterns from the rotation of Figure 2.5(a) once, one obtains the situation shown in Figure 2.5(d), where two white noise pixels are introduced from the original white noise pixel. The problem is that the pattern shown in Figure 2.5(b) is not a member of the thinning pattern set. However, this pattern cannot be added to the thinning pattern set because it always preserves the kind of white noise shown in Figure 2.5(c). Thus, 3×3 patterns fail to provide enough information to locate a concave corner.

5×5 patterns could be used in this case since they provide more information. However, they require to check a much larger neighboring area.

Each concave corner pixel has two adjacent edge pixels. For example, in Figure 2.5(a), concave corner pixel c has two neighboring edge pixels: p_2 and p_4. When these two pixels are checked against the thinning patterns and found to be edge pixels, it is possible to identify concave corner pixels such as pixel c in Figure 2.5(a), by performing a few additional checks. For example, if a pixel and its 3×3 neighbors are found to match the pattern in Figure 2.3(a) and if pixel p_1 is found to be a black pixel, then pixel p_0 will be a candidate for a concave corner pixel. Similarly, if pixel p_3 is a black pixel, then p_4 is another candidate. Since a concave corner pixel has exactly two adjacent edge pixels, a pixel that is marked twice as a concave corner candidate is a concave corner pixel. For example, in Figure 2.5(a), the concave pixel c is found to be a concave corner candidate when either pixel p_2 is checked against the pattern in Figure 2.3(c) (or 2.3(h)) and when pixel p_4 is checked against the pattern in Figure 2.3(a) (or 2.3(h)). Therefore, pixel c is a concave corner pixel. This method provides a means to locate concave corner pixels.

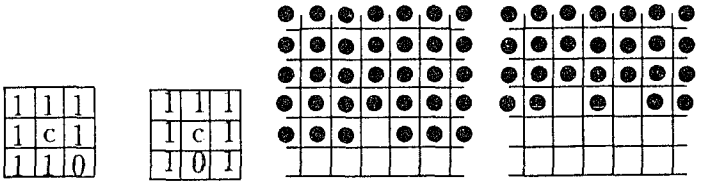


Figure 2.5: 3×3 patterns are not suitable for locating concave corner pixel.

Of all the 14 patterns used in Wu and Tsai's algorithm, only patterns (a) to (f), (h) and (i) are related to the detection of concave corners. When a pixel is found to match one of these patterns, at most two more checks are needed to decide whether

one of its 4-neighbors is a concave corner candidate. The detection of concave corner pixels for pattern (a) has been discussed in the proceeding paragraph. A similar technique can be applied to patterns (b), (c), (d), (f), and (i). Patterns (e) and (h) require a different treatment. In pattern (e), for example, p_5 is an unspecified pixel that can be either 0 or 1. In addition to the condition $P_4 = 1$ (or $p_6 = 1$), to make pixel p_4 (or p_6) a concave corner, p_5 must also be 1 so that the object is at least two pixels wide. Thus for pattern (e), two checks are needed to determine a concave corner. To summarize, the following is the check list for these eight patterns:

Pattern (a): if pixel $p_1 = 1$ then p_0 is a concave corner pixel.
if pixel $p_3 = 1$ then p_4 is a concave corner pixel.

Pattern (b): if pixel $p_3 = 1$ then p_2 is a concave corner pixel.
if pixel $p_5 = 1$ then p_6 is a concave corner pixel.

Pattern (c): if pixel $p_5 = 1$ then p_4 is a concave corner pixel.
if pixel $p_7 = 1$ then p_2 is a concave corner pixel.

Pattern (d): if pixel $p_1 = 1$ then p_2 is a concave corner pixel.
if pixel $p_7 = 1$ then p_6 is a concave corner pixel.

Pattern (e): if pixel $p_7 = 1$ and $p_5 = 1$ then p_6 is a concave corner pixel.
if pixel $p_5 = 1$ and $p_3 = 1$ then p_4 is a concave corner pixel.

Pattern (f): if pixel $p_7 = 1$ then p_0 is a concave corner pixel.
if pixel $p_5 = 1$ then p_2 is a concave corner pixel.

Pattern (h): if pixel $p_7 = 1$ and $p_1 = 1$ then p_0 is a concave corner pixel.
if pixel $p_7 = 1$ and $p_5 = 1$ then p_6 is a concave corner pixel.

Pattern (i): if pixel $p_1 = 1$ then p_2 is a concave corner pixel.
if pixel $p_5 = 1$ then p_4 is a concave corner pixel.

To locate a concave corner pixel, a marker is introduced for every pixel. As discussed above, each concave pixel c is visited by both of its neighboring edge pixels (Figure 2.5(a) for example). When the corner is first found to be a concave corner candidate (suppose by pixel p_2 in this example), the marker is set. Thus when the second neighboring edge pixel (pixel p_4 again finds that pixel c is a concave corner candidate while the marker has already been set, the corner will be designated as a concave corner pixel.

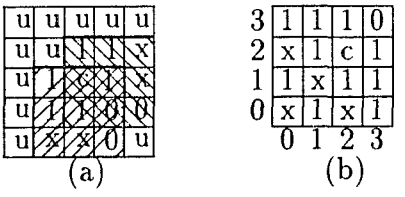
2.4.2 Preserving Connectedness

Not all the concave corner pixels detected are eventually thinned in a pass, as this would lead to discontinuity due to the symmetric feature of one-pass algorithms. To preserve connectedness, an asymmetric treatment for convex corners over concave corners is needed. For example, Figure 2.6(a) is the general situation after pixel c is recognized as a concave corner pixel. The pixels marked by 0, 1 and x are the neighboring pixels that have always been checked when two of pixel c's 4-neighbor edge pixels are processed. x means that the pixel is either 0 or 1 according to a particular situation. The 3×3 area with right-downward lines is checked when the pixel to the right of pixel c is processed. At the time we process the pixel below pixel c, the area with left-downward lines is checked. The pixels marked u have not been checked and are still unknown. If all of these u pixels are white pixels, the removal of pixel c will cut the object into two halves which is not desirable. In general, if an object is a thin object (less than two pixels wide), the removal of concave corner pixels would introduce discontinuity.

One method to avoid this problem is to stop thinning concave corner pixels when that portion of an object is thin. To determine whether the corner area is thin or

not, all of the pixels marked u in Figure 2.6(a), except the one at the bottom-right corner, need to be checked. But that requires ten more checks which is quite expensive. Since our goal is to preserve connectedness in an efficient manner, we further relax the condition to check only four of them. Pattern (o) (Figure 2.6(b)) is the new pattern we add to the thinning pattern set to guarantee that the object is more than two pixels wide around the concave corner pixel when this pixel is thinned. In pattern (o), x indicates that the pixel is unspecified, *i.e.*, it can have a value of either 0 or 1. The reason we designate pixels (0, 0), (0, 2), (1, 1), and (2, 0) as unspecified is that once pixels (0, 1), (0, 3), (1, 0) and (3, 0) are all black pixels, pixels (1, 2) and (2, 1) will not be thinned in this pass no matter what value pixels (0, 0), (0, 2), (1, 1), and (2, 0) have. Connectedness at this point is reserved by pixels (1, 2) and (2, 1) even when concave corner pixel c is removed.

Therefore, when a pixel is found to be a concave corner pixel, four more check (for pixels (0, 3), (0, 1), (1, 0) and (3, 0)) are needed to guarantee connectedness. If all of these four pixels are black pixels, the concave corner pixels will be removed.



(a): An example of concave corner pixel detection. (b): Pattern (o) — the pattern for removing concave corners.

Figure 2.6: Concave corner removal.

2.4.3 The New Parallel Algorithm

Based on the above discussion, our one-pass parallel asymmetric thinning algorithm ($OPATA_8$) is constructed in the following way. In each pass, Wu and Tsai's fourteen thinning patterns are employed to thin convex corner pixels and other edge pixels. When a pixel matches pattern (a) - (f), (g) or (i) (and thus is going to be thinned) the identification procedure described in section 4.1 is applied to locate concave corner pixels among its four 4-connected neighbors. If a concave corner pixel is found, it will be checked against pattern (o) to insure connectedness. When a match is found, the concave corner pixel will be thinned. To locate concave corners, a marker matrix is needed as an intermediate data structure. Each pixel in an image has its corresponding marker. A pseudo-code description of the algorithm is given below.

Algorithm $OPATA_8$:

Input: A binary image I_0.

Output: The skeleton S of I_0 after thinning.

1. $i = 0$;
2. **do** {
3. $flag = false$; $i = i + 1$;
4. **for all** (pixel p in I_{i-1}) {
5. $I_i(p) = I_{i-1}(p)$;
6. **if** ($p = 1$ && p's neighbors match one of
the patterns from (a) to (n)) {

7. $p = 0$; $flag = true$;
8. **for** (p_c is a concave corner candidate next to p)
9. **if** (p_c is not marked) mark p_c;
10. **else**
11. **if** (p_c's neighbors match pattern (o)) $p_c = 0$;

}

}

12. } **while** (flag);
13. $S = I_i$;

In the algorithm, I_i, $i = 1, 2, 3, \ldots$, is an intermediate result after the ith pass. Step 4 processes all pixels simultaneously. Step 5 copies the intermediate thinned image from the $i - 1$ pass. The search for edge pixels that can be thinned is done in step 6. Step 8 uses the condition listed in section 2.4.1 to determine whether any of p's 4-neighbors is a concave corner pixel candidate and then processes each one that is found. Step 11 checks the neighbors of p_c against pattern (o). The algorithm stops when no more pixels can be thinned.

Theorem 1: *Algorithm $OPATA_8$ will always terminate.*

Proof: Let $B(I)$ be the number of black pixels in image I. Since edge pixels and concave corner pixels are removed from image I_{i-1} to get image I_i, so

$$B(I_{i-1}) \geq B(I_i)$$

for $i = 1, 2, 3, \ldots$, and the $B(I_i)$s keep decreasing. Because an image has a finite number of black pixels that can be thined, eventually, when

$$B(I_{i-1}) = B(I_i)$$

the algorithm will stop. ∎

Theorem 2: *Algorithm $OPATA_8$ obtains the 8-distance skeleton of an image when the algorithm terminates.*

Proof: To prove the theorem, the only thing we need to verify is an invariant before step 5 of the algorithm that, the skeleton of image I_i is the same as that of image I_{i-1}.

Because when Algorithm $OPATA_8$ terminates which is proved by Theorem 1, all the contour pixels of the image objects will be removed and what remains are the skeletons of the objects.

To prove the invariant, we should prove that, steps 6 to 12 remove the contours of objects in image I_i (all the pixels that have 8-distance of 1 from the background) except those that are one pixel wide.

This can be proved as follows:

An 8-distance contour of an object is its 4-distance contour plus all its concave corner pixels;

- Patterns a to n (From Wu) exhaust all cases when p is a 4-distance contour pixel that is not a end pixels. They guarantee that all those 4-distance contour pixels will be removed;

- Steps 8 and 9 correctly locate all concave corner pixels. This has been discussed in detail in section 2.4.1;
- Step 11 preserves the connectedness of the skeletons by stopping the removal of a concave corner pixel when the corresponding segment of an object is less than three pixels wide.

Therefore, all the 8-distance contour pixels in I_i that don't affect connectedness have been removed after step 12. Hence the proof. ■

Theorem 3: *The number of passes taken by Algorithm OPATA$_8$ is less than, or equal to that by Algorithm OPATA$_4$.*

Proof: For any two pixels p and q in an image,

$$d_8(p,q) \leq d_4(p,q)$$

the 8-distance from an object pixel to the background is at most as big as its 4-distance, and most of the time the 8-distance is smaller.

Because the number of passes for a one-pass algorithm to converge is equal to the maximum distance of any pixel of the objects in the image from the background, a one-pass thinning algorithm that implements 8-distance, such as OPATA$_8$, will need an equivalent, if not smaller, number of passes to converge.

■

This fact is verified in the experiment discussed below in Section 3. Also, an 8-distance thinning algorithm preserves the topological features of objects better. However, since the removal of concave corner requires extra effort (at most two

more checks to locate the corners and four more checks to insure connectedness), each pass of $OPATA_8$ is expected to take slightly more time than a corresponding pass of $OPATA_4$.

When implemented in parallel, the markers may be accessed concurrently by a pixel's four 4-connected neighbors. This causes the problem of read/write conflict. However, this problem can be solved. If the algorithm is implemented on SIMD (Single Instruction Multiple Data) architectures, then, assuming that each PE (Processing Unit) processes a pixel, the conflict problem is avoided by letting all the PEs check their 8-neighbors in the same order, *i.e.*, $p_0, p_1, p_2, p_3, p_4, p_5, p_6$, then p_7. Hence no concurrent access will occur. On MIMD (Multiple Instruction Multiple Data) platforms, each marker can be treated as a shared piece of data and be accessed exclusively. The order, in which the pixel's four 4-connected neighbors check and change the marker, has no effect on the result.

2.5 Experiments

We implemented the proposed algorithm ($OPATA_8$) along with Zhang and Suen's two-pass algorithm (TPTA) and Wu and Tsai's one-pass algorithm ($OPATA_4$). All three algorithms were programmed using the Decision Tree method. Some interesting comparisons have been done. The results confirm the improvement of our new algorithm over the two existing algorithms in both the quality of the results and the speed. In this section, we will show the quality of the skeletons and compare the speeds of these three methods.

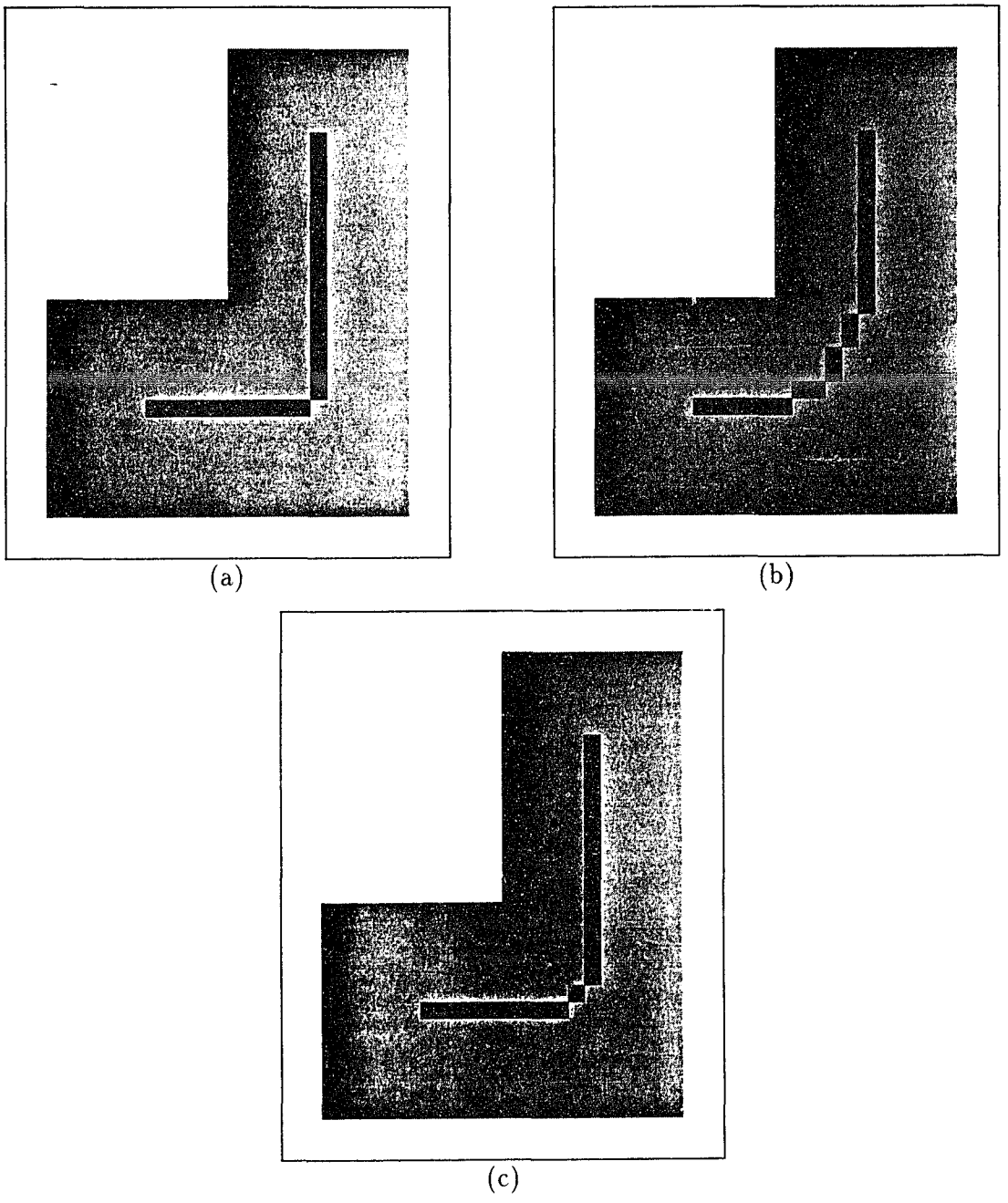
The comparisons of the results from the three algorithms for several binary images are shown below. All the input images are displayed as shadow areas. The

corresponding skeletons are shown in black color. In all of the following figures, figures (a), (b), and (c) are the result from our algorithm $OPATA_8$, Wu and Tsai's $OPATA_4$, and Zhang and Suen's TPTA, respectively. The comparisons focus on four aspects of the algorithms:

1. The preservation of topological structures of original objects;
2. Erosions of the resulting skeletons;
3. Noise resistance;
4. 8-connectedness of the resulting skeletons.

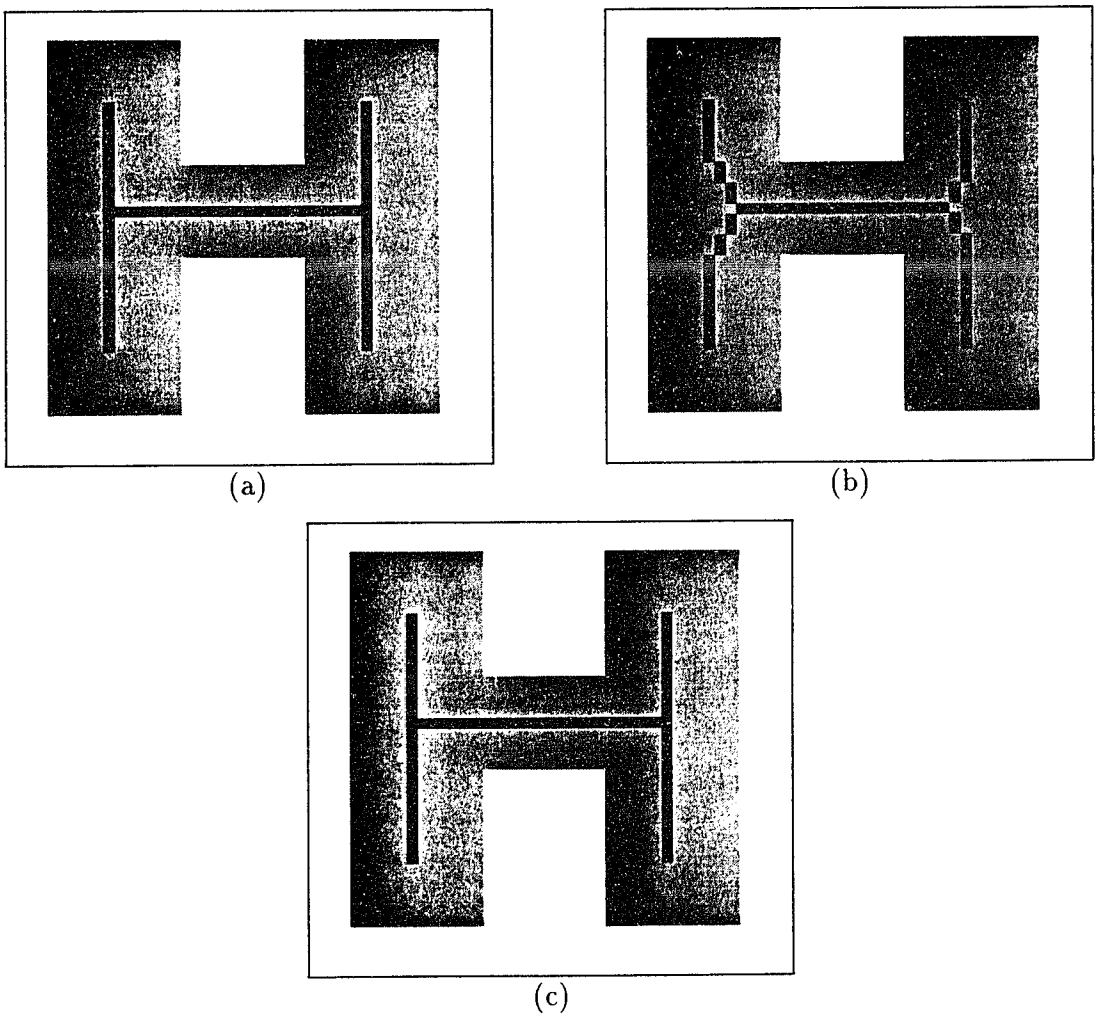
Figure 2.7 shows the significant difference between 4-distance and 8-distance based thinning algorithms in preserving the original structure of a corner. The input image has a dimension of 24×50. $OPATA_8$, which is an 8-distance based algorithm, produces a skeleton that perfectly preserves the corner feature. The output from $OPATA_4$ has a significant deterioration at the corner because it implements 4-distance. The small deterioration of TPTA in this case comes from its special treatment for corners. The skeletons of the letter "H" (Figure 2.8) provide another excellent example where $OPATA_8$ and TPTA get the same results that well capture the topological structure of the letter H, while the skeleton from $OPATA_4$ misses at corners.

TPTA has been found to have serious erosion problems. $OPATA_4$ overcomes this problem by asymmetric patterns. $OPATA_8$ successfully inherits this useful feature. Figure 2.9 is a typical case. In Figure 2.9(c) which is the result from TPTA, the skeleton of the letter X deteriorates to a three pixel long bar. In contrast, the result from $OPATA_4$ (Figure 2.9(b)) and $OPATA_8$ (Figure 2.9(a)), which are the same in



(a), (b), and (c) are the results of $OPATA_8$, $OPATA_4$, and TPTA respectively.

Figure 2.7: The difference between 8-distance and 4-distance thinning.



(a), (b), and (c) are the results of $OPATA_8$, $OPATA_4$, and TPTA respectively.

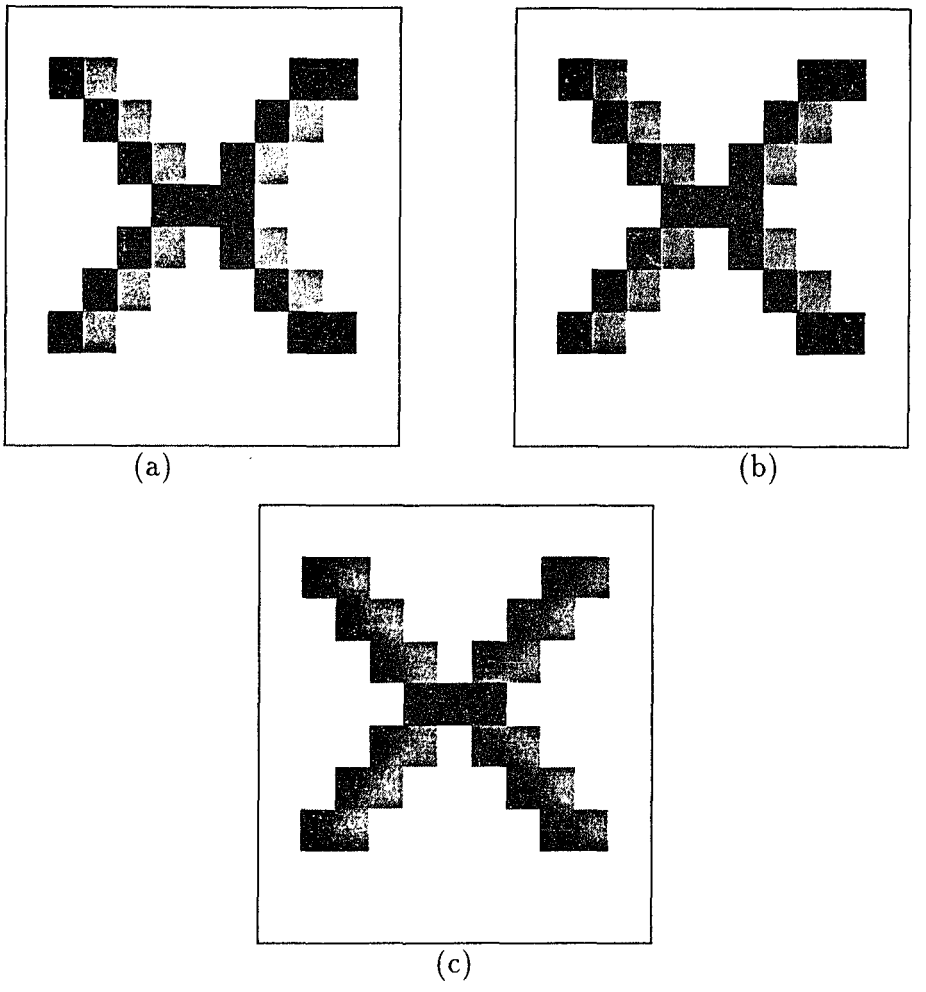
Figure 2.8: The letter H.

this case, preserve the original structure. Figure 2.2 is another example of both $OPATA_4$ and $OPATA_8$'s ability to resist erosions.

Figure 2.10 is a bar 15 pixels wide and 46 pixels long. Noises are added to it along its edges and at its corners. The results indicate that all three algorithms are good at resisting noise on the edges. For the noise at the corners, TPTA has the best performance, which suppresses all but one kind of corner noise (at top left corner). $OPATA_4$ and $OPATA_8$ cannot suppress corner noise. However, there is a tradeoff between good erosion resistance and good noise suppression. One way to reduce corner noise for OPATAs is to have a preprocessing stage to smooth corner noise.

Figures 2.11 and Figure 2.12 are two more examples that show the high quality of the skeleton produced by $OPATA_8$. In both examples, $OPATA_8$ and $OPATA_4$ get perfect 8-connected skeletons that are unitary connected. However, the results of TPTA are not unitary, but rather the pixels are 4-connected in some places. In Figure 2.12, one can see that the noise suppression of TPTA is better than that of $OPATA_8$, while result of $OPATA_4$ is even noisier than the result of $OPATA_8$.

We implemented all three algorithm in C on a DECstation 5000/240. Table 2.1 gives their time for the examples discussed earlier. For a given method and a given image, the running time shown in the table, which is given in milliseconds, is the average time over 100 repeated attempts. The number of passes is the number of times the program scans over the image in the thinning process. The results show that $OPATA_8$ and $OPATA_4$ are significantly faster than TPTA in all cases. When $OPATA_8$ requires fewer passes than $OPATA_4$ does, $OPATA_8$ is faster. When both require the same number of passes, $OPATA_4$ is slightly faster since the search for concave corners takes extra time.



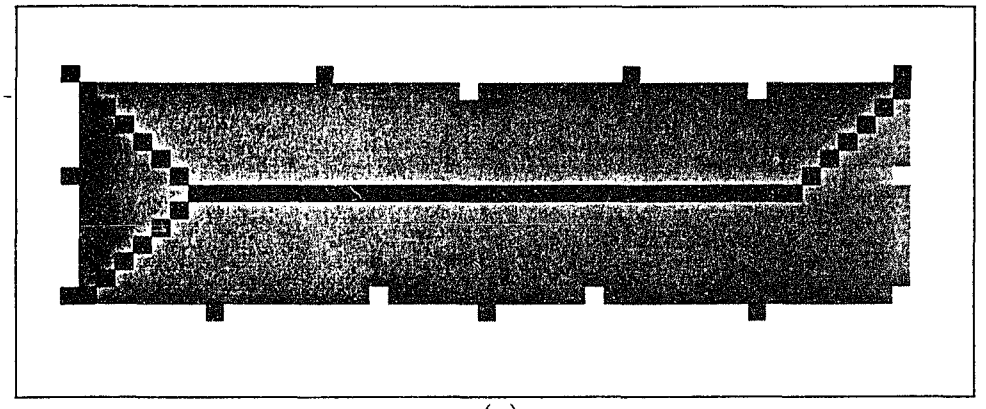
(a), (b), and (c) are the results of $OPATA_8$, $OPATA_4$, and TPTA respectively.

Figure 2.9: Another example of erosion

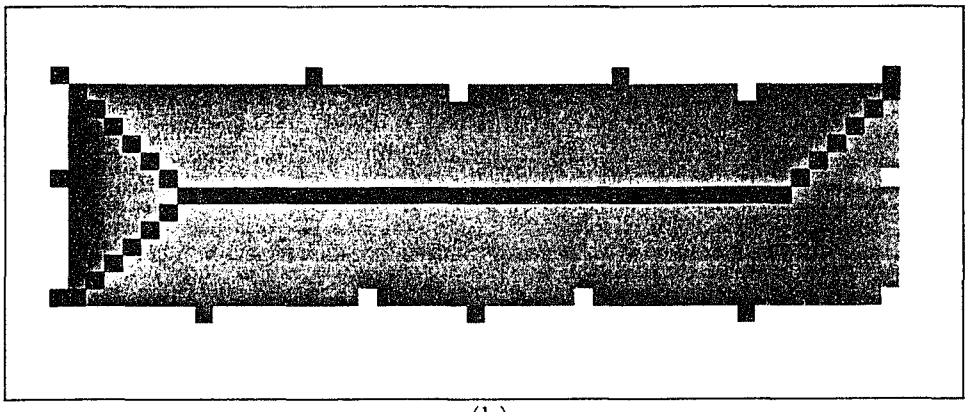
Table 2.1: A comparison of running times.

The last 3 lines give running time/passes.

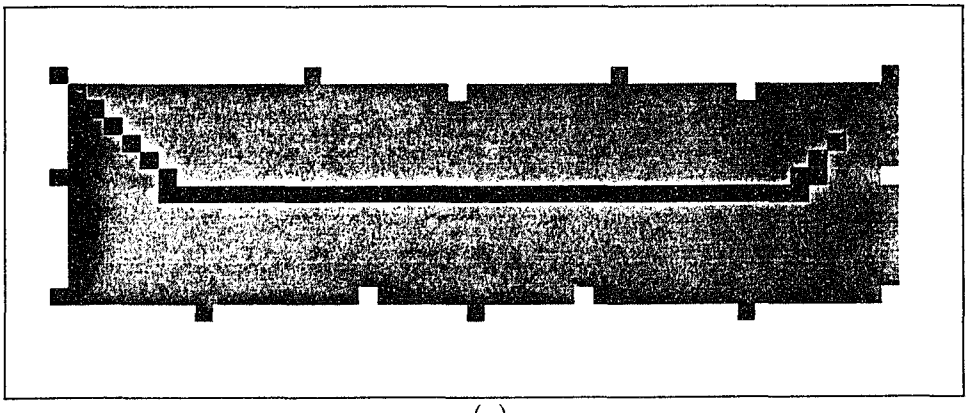
Time/Passes	Corner	H	Erosion	Noise	Chinese char	man
Size	24×30	40×40	8×9	50×30	40×50	55×65
TPTA	15.08/14	34.41/14	0.98/8	21.01/14	13.91/6	47.97/12
$OPATA_4$	11.64/9	27.30/9	0.35/2	15.58/8	10.04/4	35.54/8
$OPATA_8$	10.59/7	25.08/7	0.35/2	16.17/8	10.09/4	33.94/6



(a)



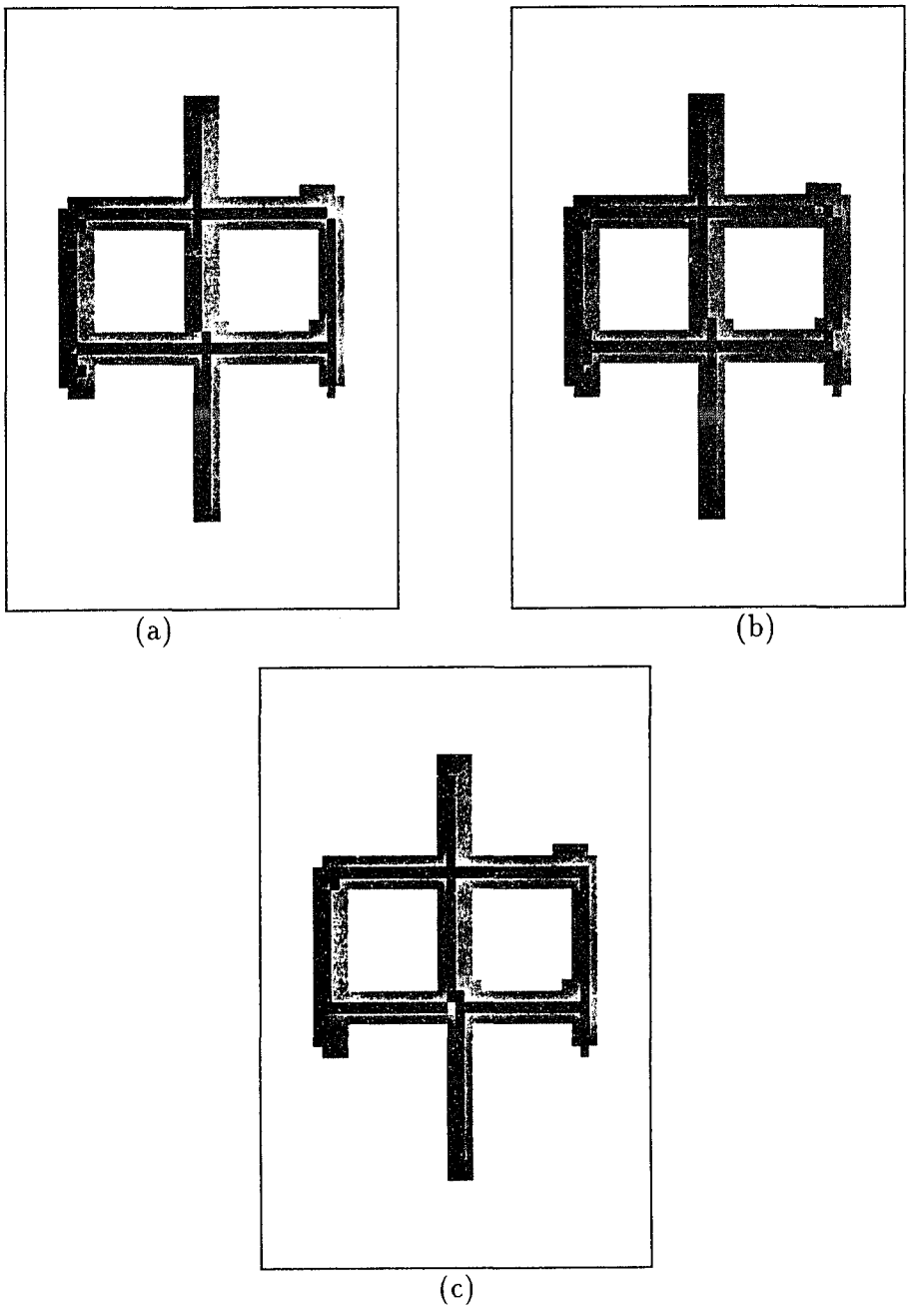
(b)



(c)

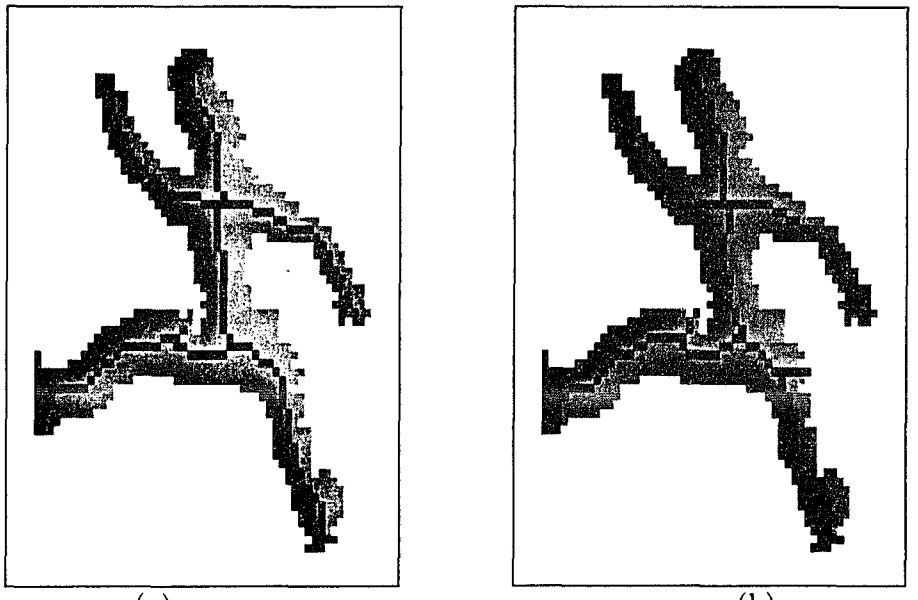
(a), (b), and (c) are the results of $OPATA_8$, $OPATA_4$, and TPTA respectively.

Figure 2.10: Noise suppression.



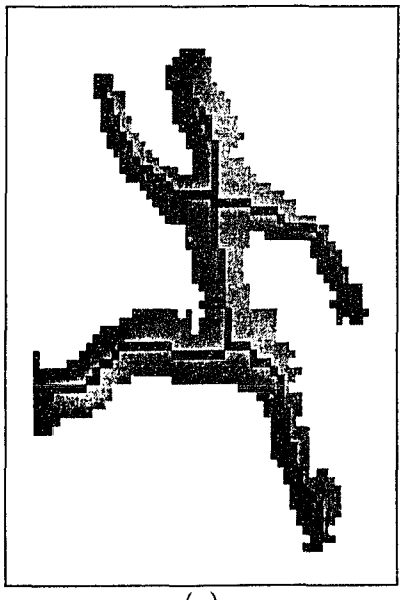
(a), (b), and (c) are the results of $OPATA_8$, $OPATA_4$, and TPTA respectively.

Figure 2.11: A Chinese character.



(a)

(b)



(c)

(a), (b), and (c) are the results of $OPATA_8$, $OPATA_4$, and TPTA respectively.

Figure 2.12: A walking man.

Chapter 3

PIMTAS

In the last chapter, we have proposed a new one-pass parallel asymmetric thinning algorithm called $OPATA_8$. Because of the implementation of the chessboard distance, the skeletons from this algorithm preserve topological information of input images much better than algorithm $OPATA_4$. In addition, $OPATA_8$ is faster than many other thinning algorithms through reducing both the number of passes and the actual running time. This proposed algorithm also has good noise resistance and obtains unitary 8-connected skeleton outputs. Recently, this algorithm has been successfully used in PIMTAS — Predictive Intelligent Military Tactical Analysis System. This section will introduce PIMTAS and the issue of applying skeletonization algorithms to this system.

3.1 Introduction

PIMTAS is a two-level hierarchical automated military route planning system, which will be applicable to both terrain vehicles (tanks, armored personnel carriers, autonomous land vehicles, etc.) and combat helicopters. The system generates optimum paths quickly. This speed is crucial in combat situations where a few seconds can make the difference between the survival or destruction of the military vehicle. The system also provides considerable flexibility in regard to cost factors associated with the path (e.g., distance or time of travel, threats, altitude, weather, etc.). By

selecting a particular set of weights, the military planner can put any desired degree of emphasis on each of these cost factors.

In developing a computer model for route planning over a large area, a prime concern is to avoid what is known as "combinatorial explosion". For example, a four-level deep full binary tree has 15 branch points or nodes while a 20-level- deep full binary tree has more than a million nodes. If the map data consist of a grid at 10 meter spacings, an unrestricted search algorithm would generate a search tree with literally millions of nodes in order to plan a route for a distance of less than 200 meters.

There are several methods that can be used to reduce the search space. They can be divided into two classes: (1) preprocessing methods and (2) methods for searching more intelligently. The preprocessing methods are concerned with the data representations of maps. Maps are normally stored in one of two forms in a computer, raster and vector. The raster is simply the ordered rows of grids mentioned above. The vector representation makes use of the fact that typically a large area of the map has identical mobility factors. This area can be at least approximately bounded by a polygon of contiguous vectors. The area, or "region" as it is generally called, can then be represented in the computer by a list of the vectors of which the polygon is composed. In contrast to the region representation, each grid in a raster is surrounded by either four or eight neighboring grids, depending on the definition of neighbor that is used. Thus for a given node, the number of neighbors and the distance to each neighbor are always constant. With a polygon representation, the number of neighbors will vary, and the distance to each neighbor is no longer well defined since the shape of the regions may be irregular. A method intermediate in storage efficiency uses a quadtree representation [67] in which the map of a square

area is divided into four quadrants. Each quadrants is subdivided if the quadrant is not completely contained within a uniform region. The process terminates when no regions remain to be subdivided. The distance from the center of any quadrant to the center of an adjacent quadrant (which may have been subdivided a different number of times and therefore is a different size) is easily computed.

There is one additional representation method which can result in a large reduction in storage requirements. Any one of a number of thinning algorithms can be applied to the mobility map in order to generate a skeletal structure that will correspond to a line drawn along the center of mobility corridors or avenues of approach. Branches in the skeletal structure are caused by obstacles that force a traveler to fork either to the left or to the right in order to go around the obstacle. Since the thinning algorithms require a binary image, the first step is to generate a mobility map containing only two levels of mobility: GO or NO-GO. The skeleton can be considered as a graph-theoretic structure to which graph theory can be applied. A graph-searching algorithm can be used to find the optimum path from the graph. However, the lines of the graph generated by the thinner may not be the optimum path between those two nodes. The solution is to use a grid level route planner to generate the precise route between nodes and a graph level route planner to efficiently plan a route over a larger area but without detail about the precise path that is followed. The graph level planner uses the traversal times generated by the grid level planner. This is the approach that was selected for use on this project. Intelligent search methods can be applied at both levels of route planning.

3.2 System Organization

Input data to this system are cross country mobility maps containing data on topography, vegetation, soil types, transportation, trafficability, *etc.*, plus information about the disposition of enemy and friendly forces and the location of weather fronts. For a given vehicle type, this input data is used to specify "GO" and "NO-GO" regions on the map. A raster-to-graph converter then converts the map to a graph with a manageable number of nodes. Next a grid level planner uses a detailed grid of points to generate the optimum (minimum cost) path between each pair of adjacent nodes on the graph. The path costs so generated are then used as weights by the graph level planner which determines several optimum non-competing routes for traveling between any two specified points on the map. Thus the grid level planner performs detailed planning over small and medium sized area but is subject to combinatorial explosion when a search over a wider area is required. The graph level planner provides the capability to efficiently plan a route over a larger area but without detail about the precise path followed. The combination of these two planners produces a powerful two-level hierarchical system that enables fast and efficient route planning over the large areas that are typical of modern battle zones, while at the same time retaining the accuracy and detail needed for combat effectiveness and vehicle survivability. The major advantages of this system are the speed of the grid level planner, which generates optimum routes in a small sized area typically in a second and the fast thinning and node-merging modules employed by the graph level planner.

The overall system is shown in Figure 3.1. It consists of a cross country mobility information base, an image skeletonization module, a graph conversion module, a node merger module, a grid-level planner, and a graph-level planner.

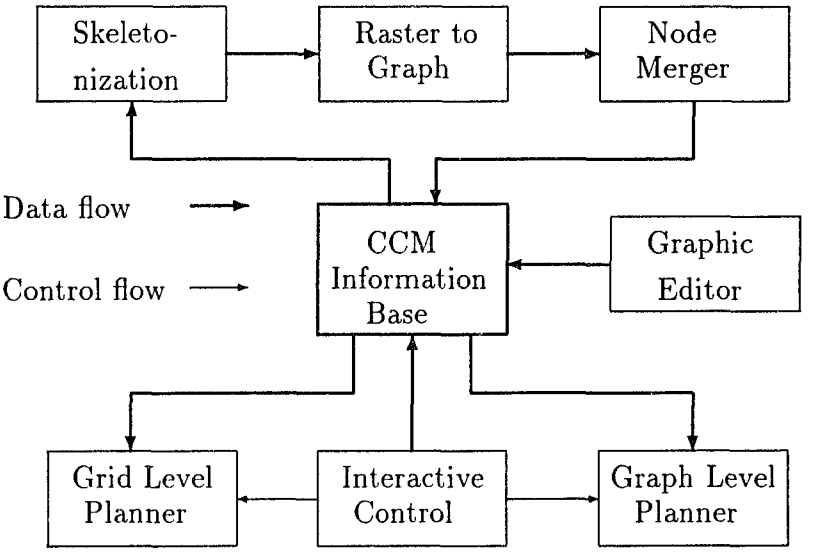


Figure 3.1: The organization of PIMTAS.

Skeletonization

Our new parallel thinning algorithm is used to extract the skeletons of GO regions.

Raster to Graph

This module accesses the skeleton of the GO regions and travels trough the skeleton to locate junctures, to convert them to graph nodes, and to connect these nodes with edges according to the skeletons. The output of this module is a graph.

Node Merger

This module scans all detected edges and eliminates closely-spaced nodes. When four or more paths of the CCM map converge to a single junction, the area of the intersection is frequently quite large compared to the width of the paths. The

thinner often generates adjacent nodes which need to be merged into a single node. The node-merger module merges nodes if the separation of the nodes is less than a threshold value.

Graphics Editor

A route planner program must provide the user a tool to manipulate the generated graph. Changes may be necessary due to new information obtained from aerial photographs, reconnaissance, or other data available to the user. The module will allow users to add nodes and edges to the graph by tracing the edge-paths with a mouse pointer. Spline curves are fitted to the traced lines and the data structures required by the graph level route planner are generated. When a path is being traced, if there is a sharp turn in the traced path, then a separate spline curve will be used on each side of the turn. A discontinuity in slope can thereby be accommodated. The route planner can be run using data generated solely by the mouse-tracker. Functions contained in the node-merger module will also be used in the graphics editor.

Grid Level Route Planner

As stated above, the weights used in the graph level route planner will be determined by the grid level route planner, which generates optimum paths quickly over a small sized area and provides considerable flexibility in regard to cost factors associated with the path. The search algorithm used by this module is based on A^*.

Graph Level Route Planner

An algorithm essentially equivalent to the A^* algorithm developed by Hart, Nilsson, and Raphael [35] is used to search the graph structure for an ordered listing of noncompetitive optimum routes between two given nodes. Each route is represented by a linked list of nodes, with the first node being the start and the final node being

the destination node. The route with the lowest cost is listed first. Since the routes are noncompetitive, no two routes can share a path and thus they cannot share two successive nodes in their respective paths.

CCM Information Base

Each grid point has an altitude penalty which depends on its height above some reference level, a threat penalty which depends on its proximity to enemy threats (missiles, anti-aircraft guns, etc.), and a weather penalty which depends on its location relative to weather fronts. Grid points that are too close to the top of a hill, a threat, or a weather front are labeled as avoided points which are not allowed to go. In the 3-dimensional version of the program, vertical grid points are added to the square grid, and the altitude, threat, and weather penalties vary in both the vertical and horizontal directions.

Interactive Control

This module provides the military planner (e.g., a tank commander or helicopter pilot) a way to choose how much emphasis he wishes to place on each of the above factors. For example, if the planner wants to reach the goal quickly and is willing to use a dangerous route to do so, he would place a large weight on time and a small weight on threats. PIMTAS will then generate a direct path that may go close to enemy threats. On the other hand, if time is not a major factor and the planner's primary concern is safety, he would place a large weight on threats and a small weight on time, in which case PIMTAS would generate a path that stays as far away from threats as possible and consequently may be much longer.

3.3 Capabilities and Limitations of Current Approaches

In the current literature, there is no automated planning system that can handle unexpected events in rapidly changing battlefield environment. More specifically, the Tactical Movement Analyzer (TMA) system [48], developed at Jet Propulsion Lab, finds a path for a land based military unit, such as an infantry unit, armored unit, tank unit, *etc.*, that desires to move from a starting point to a goal in military operations. The TMA considers a number of factors that affect the travel time of the military unit, called a "mover", including terrain features, roads, railroad tracks, rivers, bridges, vegetation, weather, *etc.*, as well as the physical attributes of the mover (maximum speed, ability to move over rough terrain, *etc.*) and then determines the path that will require the least amount of time. Thus the only factor used by TMA to determine an optimum path is time of travel. Terrain features, such as hills, and weather conditions are considered only to the extent that affect the travel time. Enemy threats that may be located in the combat theater are not considered by the current version of TMA.

While several other grid level route planners are currently available, *e.g.*, the system of Sudkamp, Lizza, and Wagner [78] (SLW) and the ARM system of Systems Control Technology [52] (SCT), these do not have the speed or flexibility needed for the applications proposed here. The SLW planner, for example, does not employ an underestimator function in its version of the A^* algorithm, and as a result, it is too time-consuming to be useful in a rapidly changing battlefield situation. The SCT planner, which was developed for Cruise Missiles, requires extensive processing times (on the order of several hours) and hence is also inappropriate for the rapid response terrain vehicle/helicopter route planner. In addition, both the SLW and

SCT systems lack the flexibility to put different amounts of emphasis on the various cost factors associated with the path.

3.4 An Application of Skeletonization

The efficiency and quality of the thinning algorithm is crucial to the success of PIMTAS. Zhang's TPTA was first used in this system. It produced good quality skeletons at a slow speed. In an application like PIMTAS that requires quick response, time is an important factor. So, we looked for fast thinning algorithms. First, we used a better implementation method to speed up TPTA. Original TPTA algorithm was not written in an efficient way. To count the number of black pixels in 3×3 neighbor takes seven additions. To find the number of 01 pattern in the sequence of $p_0p_1p_2p_3p_4p_5p_6p_7$, eight comparisons, eight assignments, and eight additions are required. Finally, to verify the four conditions set by TPTA, four more comparisons are needed. This takes a total of 35 operations to thin a single pixel.

3×3 matrix has 512 different combinations. After carefully examining the conditions to thin a pixel, we found that only 40 out of 512 patterns are patterns that are corresponded to the conditions set by TPTA that the pixel should be thinned. So, one way to improve the efficiency is to use pattern match. A nine dimensional array is constructed to store the patterns using $p, p_0, \ldots, p_7$ as indices. The array elements that correspond to a thinning pattern have a value of 1 and the rest have a value of 0. However, to calculate the index of the array seven multiplications and seven additions are still needed. To check whether the pattern is a thinning pattern takes one comparison. Totally, fifteen operations are required to thin a pixel with this pattern match method.

Using better data structure can further speed up the thinning process. We reconstruct the pattern table using a decision tree — an efficient data structure. The decision tree has a height of at most nine. Once the tree is built, at most nine comparisons are needed to retrieve a pattern. Thus, at most nine operations are needed to thin a pixel.

All the three algorithms discussed in the last chapter have been implemented using decision tree method. Fig 3.2 is a decision tree for $OPATA_4$. The tree is searched when the current pixel p has a value of 1. The decision tree for TPTA is more complicated.

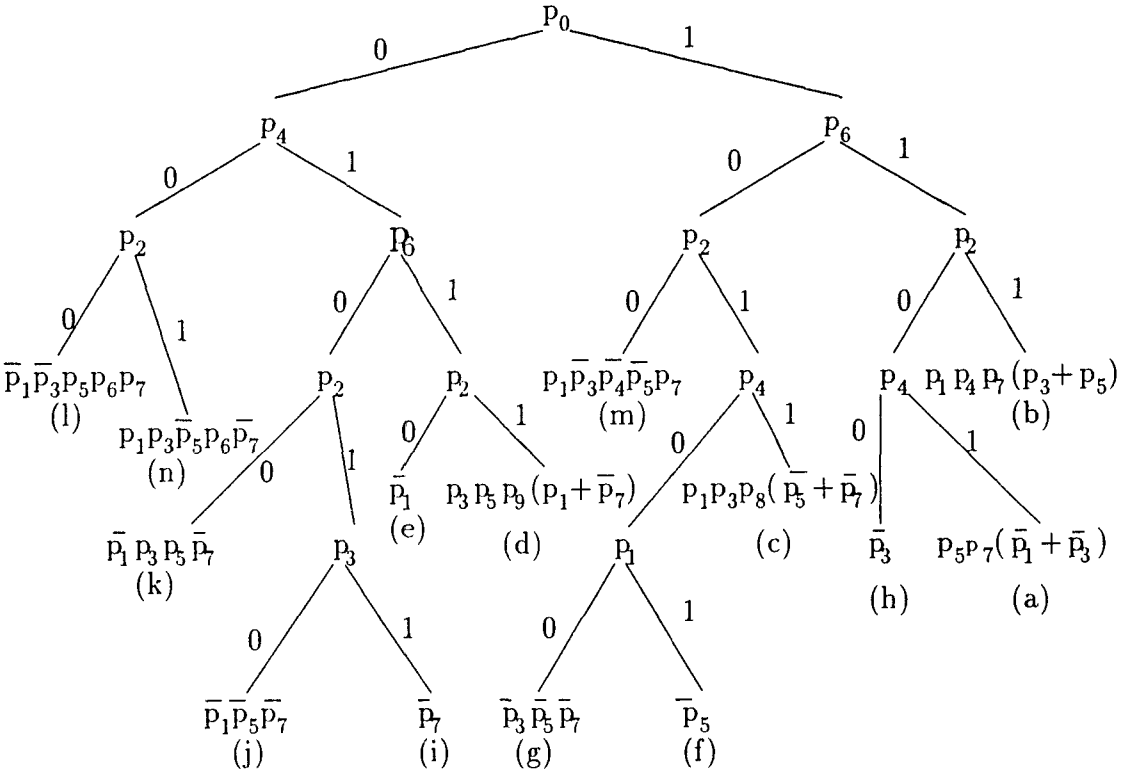


Figure 3.2: Decision tree for $OPATA_4$.

For a mobility map shown in Fig map(a), different colors indicate different kind of mobilities. the map is 237 × 224 in dimension. After extracting the GO/NO-GO binary map Fig 3.3(b), thinning algorithms are used to obtain the skeleton. Fig 3.3(c), Fig 3.3(d) and Fig 3.3(e) are the skeleton from TPTA, $OPATA_4$ and $OPATA_8$ respectively. Table 3.1 shows the time and speedup for TPTA, TPTA's decision tree implementation, $OPATA_4$, and $OPATA_8$. Using decision tree alone, TPTA is speeded up by a factor of 1.65. And our new algorithm $OPATA_8$ has a speedup of 2.59. The route planner is implemented in Common Lisp on DECstation 5000/240.

Table 3.1: The speedup of our new thinning approach.

	TPTA	TPTA(decision)	$OPATA_4$	$OPATA_8$
Time(seconds)	108.67	59.5	48.55	41.98
Speedup	1.0	1.65	2.24	2.59

In this chapter, we presented an advanced movement analysis system that based on hierarchical structure. The system is being developed to produce quick responses to many factors in a battlefield situation. We discussed in detail the use of thinning algorithms in this system. Implementation issues of thinning algorithms are investigated. This research shows that searching for a better implementation of an existing algorithm, using a better data structure, and looking for a more efficient algorithm can all speedup an existing application.

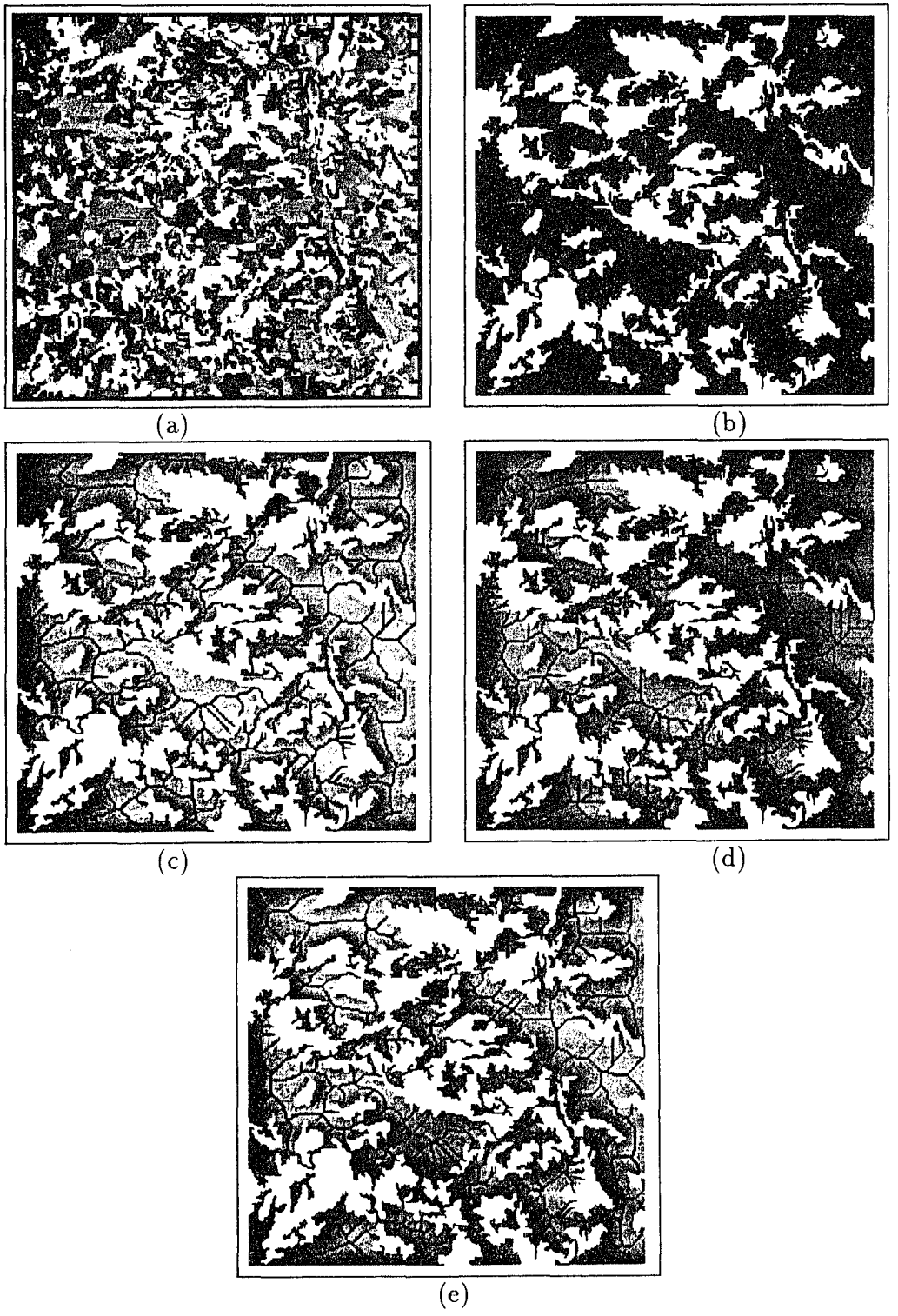


Figure 3.3: Thinning the mobility map.

Chapter 4

A Unified Locality Model for Computer Vision

Chapter 2 presented an example of computer vision processes that have explicit locality. For many computer vision tasks, locality is not so obvious. However, many of them have one thing in common — the interpretation of sensory data, the need to classify perceptual objects (Here, an object is an abstract concept. It can be a pixel, a segment, or an image of a real object). For example, in recognition phase, the separated segments are objects. The goal is to assign them with labels such as chairs, desks, and book shelves. Another example is edge detection. Pixels in the image, which are objects, need to be labeled as edge pixels or non-edge pixels.

In general, suppose we analyze a picture or a scene, which we would like to understand. Also suppose we have a set of objects that may exist in the scene but have not been identified unambiguously. The observed information of the objects and the knowledge of relationships between the objects can both be used to reduce, or even eliminate the ambiguity so as to find the correct labels for the objects [65]. This is the problem of labeling.

It has been found that object specific information(or observed measurement) is not sufficient to obtain unambiguous identifications of objects. However, in reality, objects are inevitably interconnected. The constraints imposed by the mutual relationships between individual objects are useful to reduce ambiguity [46].

Relaxation labeling refers to a family of contextual labeling algorithms that aim at the global interpretation of objects by recursively updating symbolic label

(or meaning) assignments [49]. It was originally developed to assign numeric or symbolic labels to objects in the presence of ambiguities. The goal is to use prior knowledge about the problem domain, in the form of the likelihood of co-occurrence of particular label assignments, to resolve ambiguities in a manner consistent with the initial data, and to do so by means of a series of simple, local computations which could be implemented in local parallel structures [59].

In this chapter, we will investigate the problem of probability relaxation labeling. The aim of our study is to develop a unique relaxation scheme which can lead to efficient parallel algorithms for many image applications. we will briefly review the related techniques developed in this area especially in the direction of probability relaxation labeling and the major problems in this area. Then, we propose our new scheme.

4.1 Related Works

Relaxation, as a numerical method, was first introduced by R. V. Southwell [24] in early 40s. It is used to solve large systems of linear equations, and to get finite difference approximations to partial differential equations. It was found that relaxation had two distinguish features. Though based on the same idea, the way for different users to apply the method can be very diverse. But their answers can all be as correct as required. However, the convergent rates may vary significantly. The other feature is its robustness to calculation error. Even when an error occurs during the calculation, recalculation is not necessary. The error can be eliminated by continuing the relaxation ([72, 73]).

The pioneering work in relaxation labeling was done by Waltz [84]. He considered the problem of labeling edges in line drawings. In order to maintain consistency, only unambiguous identification of line segments are allowed in his formulation. This was done by using the idea of relaxation that sequentially filtering out inconsistent identification pairs on connected segments [36].

Rosenfeld *et al* [65] extended Waltz's work to a more general technique, which they called discrete relaxation labeling. They showed that Waltz's filtering has locality and can be performed in parallel and implemented as a network of processors each of which works for an object. They further argued that it was better to formulate the problem in a continuous domain instead of discrete relaxation that enforces unambiguous labeling. Nonlinear probability relaxation labeling was then proposed that defined nonlinear rules to iteratively update label weights so as to improve labeling consistency.

The general idea of probabilistic relaxation labeling is as follows. Suppose a set of objects $V = \{1, 2, \ldots, n\}$ are classified into m categories $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$, each category j is represented by a label λ_j, $1 \leq j \leq m$. Suppose further that the category assignments are correlated, and the correlations are described by graph $G = (V, E)$, where an edge in G represents a direct interaction between two objects. Let $P^{(0)}\{x_i = \lambda_j\}$ be an initial estimate of the probability that object i belongs to category λ_j. For each object i, $P^{(0)}\{x_i = \lambda_j\}$ should satisfy $0 \leq P^{(0)}\{x_i = \lambda_j\} \leq 1$ for $1 \leq j \leq m$ and $\sum_{j=1}^{m} P^{(0)}\{x_i = \lambda_j\} = 1$. This initial estimate is calculated from observation vector $\vec{y}_i$ of each object i and $Y = \{\vec{y}_i \mid i \in V\}$. The goal of probabilistic relaxation labeling is to find a classification for the objects that is compatible with the initial estimate $P^{(0)}\{x_i = \lambda_j\}$ and the correlation described by

graph G. The assignment of a label λ_j to object i is based on *a posteriori* probability $P\{x_i = \lambda_j \mid Y\}$ of assigning label λ_j to object i under observation set Y.

Since the emerge of relaxation labeling technique, various schemes have been developed. These different approaches can be classified into two categories.

One is **Discrete Relaxation Labeling** in which each label of an object is either possible or impossible ([22, 27, 28, 65]). The relaxation procedure consists of an iteratively pruning, in parallel over all objects, of labels which are incompatible with the labels of their neighboring objects. The developments of efficient algorithms both sequential and parallel are relatively easy ([26, 66]).

The other category is called **Probabilistic Relaxation Labeling**. A probability is given to each label assignment for each object. The main issue in the development of a probabilistic relaxation labeling algorithm is to construct a transformation rule over probability vector space that recursively refines initial probabilities. A variety of different transformation rules were proposed. They formed the family of probabilistic relaxation labeling algorithms ([40, 44, 45, 65]).

To develop probability relaxation labeling schemes, four approaches have been used: heuristics, optimization, probability theory and Markov random field theory.

Rosenfeld's probability labeling schemes were developed based on heuristics. That is the reason that the supporting function defined for their nonlinear relaxation labeling induces the problem of bias and convergence. Hummel and Zucker [40] overcame these problems by reformulating relaxation labeling as an optimization process with some objective functions which quantify labeling consistency. They formally characterized the goal of relaxation labeling, developed a projected gradient update scheme, and derived the property of local convergence for their transformation rule. This transformation rule has theoretical basis and hence permits an analytic proof

of convergency. However, the transformation function is not easy to be implemented efficiently.

Kittler and Hancock ([27, 45]), on the other hand, argued that the probabilistic relaxation labeling problem was overspecified in [65] with the definition of the supporting function. They developed an evidence combining formula in the framework of probability theory. Based on a few reasonable assumptions, they derived a non-linear probabilistic relaxation transformation rule that is similar to Rosenfeld's [65]. Because no heuristics was used, the problems of bias, choice of compatibility coefficient, choice of the supporting function, and interpretation of the computed probabilities were easily solved. One of the difficulties besetting this scheme was its potential computational intensity. The number of possible global label configurations is exponential to the number of objects to be labeled. This problem was solved by the observation that, far from being exponential, in many realistic applications, the number of permissible configurations is relatively small because the applications are highly structured. Therefore, exhausted compilation of permissible configurations was made to form dictionaries. In this way, the efficiency of probabilistic relaxation labeling was significantly improved. Kittler and Hancock ([27, 28, 29, 45]) successively used dictionary method to develop their probabilistic relaxation labeling algorithm. Their experimental results [27] show that their scheme performs better than some well-known edge detection algorithms like Canny's, Hilditch's, and Spacek's.

Markov Random Field (MRF) theory that was developed as a class of parametric model for spatial data, provides an important theoretical foundation. The locality of MRF well captures the localization characterization of probabilistic relaxation labeling. A lot of research has been conducted in this direction. Most of them

attempted to find global configuration with MAP (Maximum *a Posteriori* probability) using discrete relaxation ([22, 45, 13]).

Among them, Geman and Geman's paper [22] has been considered as "the invention of refreshing and novel techniques" [12]. Images are considered as instances of a MRF with a Gibbs distribution. An energy function is defined for the MRF that the original image has minimal energy while degraded images have higher energies. Then, a stochastic approach minimizes the energy of the MRF by making local changes randomly based on neighborhood pixel values and a control parameter T (also called temperature). The changes may either increase the energy to avoid local minima or decrease it to advance to global minima. The temperature T decreases from a high initial temperature to zero which makes the process begin with purely random changes and end with gradient descent. This simulated annealing technique results in a highly parallel relaxation algorithm that uses *a posteriori* distribution to yield a MAP estimate to restore images from degrade observations. However, the problem of their method is its convergence rate. Due to the nature of simulated annealing, hundreds of iterations are needed to obtain good restoration. Furthermore, the method is noise sensitive. Good restoration was obtained in low signal to noise ratio.

Pelkowitz [61] developed a probabilistic relaxation algorithm using MRF theory. He applied Maximum Entropy estimate approach and derived a multilinear relaxation update function. The function is data dependent and the final configuration (a fix point in the configuration space) is a function of observations that locally optimizes the *a posteriori* probability.

Relaxation labeling methods have been applied to a variety of image processing problems, such as image restoration [22], edge enhancement [68], edge detection

Table 4.1: Some of the important results in Relaxation Labeling

Investigator(s)	Method	Advantages	Disadvantages
Rosenfeld et al (1976)	Heuristic method; Probability model.	Extend discrete model to probability model.	The problem of bias, consistency, choice of supporting function.
Hummel & Zucker (1983)	Projected gradient updating scheme; Probabilistic relaxation.	A projection operator using projected gradient updating; Proof of consistency.	Lack of efficient implementation.
Gemans (1984)	Markov Random Field; Gibbs Distribution.	Noncausal; Nonlinear; Optimization process.	Slow convergency; Sensitive to noise.
Sammal & Herderson (1987)	Parallel algorithm; Discrete relaxation.	$O(nm)$ parallel implementation.	Discrete model only.
Kittler & Hancock (1990)	Probabilistic relaxation; Dictionary model.	Probability based operator; Fast convergency; Single pixel edges.	Still not very good in noise resistence.
Pelkowitz (1990)	Markov Random Field; Probabilistic relaxation.	Multilinear operator; Local optimization.	Slow convergence rate.

([27, 28, 29, 32, 65]), pixel classification [15], and image segmentation [32]. It can also be used in object recognition and in artificial intelligence. A survey by Kittler and Illingworth [47] on relaxation labeling highlights the importance of this research area and points out the advantages and the possible applications of relaxation labeling. Table 4.1 summarizes some important results in this area.

4.2 Statement of the Problem

The problem of probabilistic relaxation labeling can be described as: given

1. A set of objects(or vertices) $V = \{1, 2, ..., n\}$.

2. A set of labels $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$. More generally, for different object i, there should be a different label set Λ_i. For the sake of simplicity, it is often assumed that $\Lambda_i = \Lambda_j$, for all $i, j \in V$. Labels cannot be measured directly.

3. A graph $G = (V, E)$ that describes neighborhood relations among objects in V, *i.e.* the contextual relation among objects.

4. A set Y of noisy observations,

$$Y = \{\vec{y}_i = h(x_i, u_i) \mid i \in V\}$$

where $\vec{y}_i$ is a vector of observations (or measurements) for object i that depends on the true label assignment x_i and the random noise u_i. Here, x_i and u_i are assumed to be independent.

5. *A priori* probability $P\{\vec{y}_i \mid x_i = \lambda_j\}$ for any $i \in V$ and $j \in \Lambda$.

Find a consistent label assignment for all objects in V by relaxation method so that a unique label λ_j from label set Λ is assigned to each object i. The assignment of label λ_j to object i is based on *a posteriori* probability $P\{x_i = \lambda_j \mid Y\}$, the probability that assigns label λ_j to object i under observation set Y.

For the example of edge detection, the object set V consists of all the pixels in an image. Two labels, "edge" and "non-edge", are in the label set Λ. Graph G has the object set V as its vertex set. Its edge set E consists of edges that connect pixels to their eight neighboring pixels. An example of observations is the grayscale value of a pixel.

In Rosenfeld's non-linear relaxation labeling, the contextual information conveyed by label λ_k of object i about label λ_l of object j is represented by a compatibility coefficient $r(x_i = \lambda_k, x_j = \lambda_l)$. The current sth estimate of the probability that object i is labeled as λ_k is $P^{(s)}(x_i = \lambda_k)$. Then the supporting function describing the support for a label by all the other objects is

$$Q^{(s)}\{x_i = \lambda_k\} = \sum_{j=1}^{n} C_{ij} \sum_{l=1}^{m} r(x_i = \lambda_k, x_j = \lambda_l) P^{(s)}\{x_i = \lambda_l\}$$

Where C_{ij} are weights that $\sum_{j=1}^{n} C_{ij} = 1$. C_{ij} expresses the relative importance of each object's support. The update function is

$$P^{(s+1)}\{x_i = \lambda_k\} = \frac{P^{(s)}\{x_i = \lambda_k\}Q^{(s)}\{x_i = \lambda_k\}}{\sum_{l=1}^{m} P^{(s)}\{x_i = \lambda_l\}Q^{(s)}\{x_i = \lambda_l\}}$$

Defining different supporting functions results in different relaxation schemes. Many schemes have been proposed. However, many issues remain, such as how to define suitable compatibility relation, how to explain the effect of a given scheme, and how to prove convergence properties. For example, Rosenfeld's scheme did introduce bias [60]. To make it unbias, we can either find a new unbias scheme, or derived the condition on compatibility and supporting function which eliminate bias [90, 34].

Besides unbiasness and convergency, another important issue is the convergent rate. Unbiasness and convergency makes a relaxation scheme theoretically sound. However, it is the convergent rate that determines whether a method is practically useful. relaxation algorithms are theoretically computationally intensive and their convergence rates are still not well formulated. Therefore, how to improve their efficiency, especially by certain practical assumptions that are true in most applications, is an important issue.

4.3 Markov Random Field Theory

Markov Random Fields are a natural generalization of one-dimensional Markov processes. Time index is replaced by a two- or higher-dimensional space index. MRFs have many interesting applications. MRF concept is the result of attempts to formulate the Ising model of ferromagnetism using probability theory. It has been

successfully applied to both physical and biological system as models. The use of MRF image models enables us to exploit analogies with statistical mechanics systems.

Let graph $G = (V, E)$ define relations among objects where V is the set of objects and E a set of edges connecting pairs of objects. E defines a neighborhood system $\{\partial i, i \in V\}$ for object set V. The set of neighbors for object i, ∂i, consists of the objects of V that are connected to i by edges in E, *i.e.* $\partial i = \{j \mid (i,j) \in E\}$

Associated with object i is a random variable x_i defined on label set Λ. $X = \{x_1, \ldots, x_n\}$ is the set of random variables for a given problem. $x_i = \lambda_j$ represents the assignment of label λ_j to object i. Set $\omega = \{x_1 = \lambda_{j_1}, x_2 = \lambda_{j_2}, \ldots, x_n = \lambda_{j_n}\}$, called configuration, describes a label assignment for object set V. The set of all configurations is called a configuration space $\Omega = \Lambda^{|X|}$. We use X_i to denote the set of all the random variables associated with objects in V excluding x_i (the random variable for object i), *i.e.* $X_i = \{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. Then ω_i denotes a configuration for X_i, and Ω_i the configuration space of all the configurations ω_i. $\omega_{\partial i}$ represents a configuration for variable set $X_{\partial i} = \{x_j \mid j \in \partial i\}$, *i.e.* a neighborhood configuration for object i. $\omega_{\partial i}(l)$ denotes the label assignment for object l in object i's neighborhood under configuration $\omega_{\partial i}$, *i.e.* the label assignment $x_l = \lambda_{j_l}$ in $\omega_{\partial i}$. $\Omega_{\partial i}$ denotes the configuration space of all the configurations $\omega_{\partial i}$ over ∂i.

ω is regarded as a sample realization of a random field X over label set Λ. This random field X is called a *Markov Random Field* if the measures (or probabilities) P of its configurations ω have the following two properties:

1.

$$P\{\omega\} > 0$$

for all $\omega \in \Omega$, *i.e.* the positivity of P;

2.

$$P\{x_i = \lambda_j \mid \omega_i\} = P\{x_i = \lambda_j \mid \omega_{\partial i}\} \tag{4.1}$$

the locality of Markov Random Field, *i.e.* the probability of assigning label λ_j to object i depends only on object i's neighborhood label configuration.

If the objects of G are a set of $n \times n$ pixel sites, with edges connecting all sites within some given Euclidean distance of each other, and Λ is a set of all gray levels, then Ω is the family of all possible gray scale images, and a MRF defines a stochastic image model that a digitized image is a state of a lattice-like physical system whose dynamics are defined by short range interactions.

The graph G defines relations among objects. The label of an object strongly depends on the label context of its neighboring objects (also called *directly interacting objects*). Other objects that are not adjacent also provide useful contextual information, but in an indirect way. The neighborhood system devices for an object all other objects into two categories: *directly interacting objects* and *indirectly interacting objects*.

4.4 A Unified Relaxation Scheme

The kernel of probabilistic relaxation labeling is a transformation function, also called projection operator, that describes the relation of a label assignment of an

object with the label assignments of its neighboring objects. This function is used to gradually involve more and more contextual information from nearby objects to refine the estimate $P\{x_i = \lambda_j \mid Y\}$ for label assignment $x_i = \lambda_j$.

Given a set of objects V and a set of observations Y over V, if the *a posteriori* probability of assigning label λ_j to object i under observation Y, $P\{x_i = \lambda_j \mid Y\}$, is calculated for any $i \in V$ and $\lambda_j \in \Lambda$. Then, the label assignment with the highest *a posteriori* probability is selected as the label assignment for object i, *i.e.* object i is assigned label λ_j if

$$P\{\ x_i = \lambda_j \mid Y\ \} = max_{k=1}^{m}\ P\{\ x_k = \lambda_{j_k} \mid Y\ \} \tag{4.2}$$

This is called Maximum *a posteriori* probability (MAP).

However, relations among objects in V are usually very complicated and not easy to model. Calculating $P\{x_i = \lambda_j \mid Y\}$ from observation Y directly is very difficult. One way to solve this problem is to give $P\{x_i = \lambda_j \mid Y\}$ a reasonable estimate.

Let's first consider a very simply estimate for $P\{x_i = \lambda_j \mid Y\}$, Assume that a label assignment for an object depends on the observations on that object only, *i.e.* $P\{x_i = \lambda_j \mid Y\} = P\{x_i = \lambda_j \mid \vec{y}_i\}$. When *a priori* probability $P\{x_i = \lambda_j\}$ and conditional probability $P\{\vec{y}_i \mid x_i = \lambda_j\}$ are both known, $P\{\ x_i = \lambda_j \mid Y\ \}$ can be estimated by

$$P\{x_i = \lambda_j \mid Y\} = P\{x_i = \lambda_j \mid \vec{y}_i\} = \frac{P\{\vec{y}_i \mid x_i = \lambda_j\}P\{x_i = \lambda_j\}}{P\{\vec{y}_i\}} \tag{4.3}$$

$$P\{\vec{y}_i\} = \sum_{j=1}^{m} P\{\vec{y}_i \mid x_i = \lambda_j\} P\{x_i = \lambda_j\} \tag{4.4}$$

However, $\vec{y}_i$ contains noise. When the above estimates are used to assign MAP labels, label errors will occur. The assumption that a label assignment only depends on local observations ignores the interactions between objects. These interactions contain the contextual information needed to eliminate noise effects.

Error labels are very likely to be inconsistent within the label context of an object system. Thus, label context can be used to refine the estimate of $P\{x_i = \lambda_j \mid Y\}$. However, the interactions of objects in an object system are very complicated. Capturing all the interactions with a single formulae is impractical. Therefore, to incorporate contextual information into the estimate of $P\{x_i = \lambda_j \mid Y\}$, for given object set V, neighbor system ∂V is used to divide the interactions among objects into two categories. An interaction between an object and its neighbor (its adjacent object) is called a *direct interaction.* Other interactions are called *indirect interactions.* Employing direct interactions only to estimate $P\{x_i = \lambda_j \mid Y\}$ balances the need to use contextual information and the complexity to model object interactions.

In order to formulate direct interactions, we assume that a label assignment of an object depends only on the label configuration of its neighboring objects and its observations $\vec{y}_i$. This makes label assignment X a Markov Random Field. Under this assumption, we have the following lemma.

Lemma 1:

$$P\{\ x_i = \lambda_j \ \mid \ Y\ \} = \sum_{\omega_{\partial i} \in\ \Omega_{\partial i}} W_{\omega_{\partial i}\ \mid Y}\ E(i, j, \omega_{\partial i}) \tag{4.5}$$

where

$$E(i,\ j,\ \omega_{\partial i}) = \frac{P\{x_i = \lambda_j \mid \vec{y}_i\} P\{\omega_{\partial i} \mid x_i = \lambda_j\}}{\sum_{k=1}^{m} P\{x_i = \lambda_k \mid \vec{y}_i\} P\{\omega_{\partial i} \mid x_i = \lambda_k\}} \tag{4.6}$$

$$W_{\omega_{\partial i}|Y} = P\{\omega_{\partial i} \mid Y\} \tag{4.7}$$

Proof: The proof is a direct result from the application of MRF property (Equation 4.1) and the independent assumption on observation noise.

By the theorem of total probabilities,

$$\begin{aligned} P\{\ x_i = \lambda_j \mid Y\ \} &= \sum_{\omega_i \in\ \Omega_i} P\{\ x_i = \lambda_j, \omega_i \mid Y\ \} \\ &= \sum_{\omega_i \in\ \Omega_i} P\{\ x_i = \lambda_j \mid \omega_i, Y\ \} P\{\ \omega_i \mid Y\ \} \\ &= \sum_{\omega_i \in\ \Omega_i} E(i,\ j,\ \omega_i)\ P\{\ \omega_i \mid Y\ \} \end{aligned} \tag{4.8}$$

$E(i,\ j,\ \omega_i)$ denotes $P\{\ x_i = \lambda_j \mid \omega_i, Y\ \}$. Using Bayes' rule and the theorem of total probabilities, we have

$$\begin{aligned} E(i,j,\omega_i) &= P\{\ x_i = \lambda_j \mid \omega_i, Y\ \} \\ &= \frac{P\{\ x_i = \lambda_j, \omega_i, Y\ \}}{P\{\omega_i, Y\ \}} \\ &= \frac{P\{\ x_i = \lambda_j, \omega_i, Y\ \}}{\sum_{k=1}^{m} P\{\ x_i = \lambda_k, \omega_i, Y\ \}} \end{aligned} \tag{4.9}$$

Applying Bayes' rule to $P\{\ x_i = \lambda_j, \omega_i, Y\ \}$, we get

$$\begin{aligned} P\{\ x_i &= \lambda_j, \omega_i, Y\ \} = P\{\ x_i = \lambda_j, \omega_i, \vec{y}_i, Y_i\ \} \\ &= P\{\ \vec{y}_i \mid x_i = \lambda_j, \omega_i, Y_i\ \}\ P\{\ Y_i \mid x_i = \lambda_j, \omega_i\ \}\ P\{\ x_i = \lambda_j, \omega_i\ \} \end{aligned} \tag{4.10}$$

Y_i denotes the set of observations for all objects in V except that for object i. Because $\vec{y}_i$ depends on x_i and noise u_i, and u_i is independent of all x_j for $j \neq i$, so

$$P\{\ \vec{y}_i \mid x_i = \lambda_j, \omega_i, Y_i\ \} = P\{\ \vec{y}_i \mid x_i = \lambda_j\ \} \tag{4.11}$$

$$P\{\ Y_i \mid x_i = \lambda_j, \omega_i\ \} = P\{\ Y_i \mid \omega_i\ \} \tag{4.12}$$

From property 2 of MRF (Equation 4.1), we have

$$P\{\ x_i = \lambda_j, \omega_i\ \} = P\{\ x_i = \lambda_j, \omega_{\partial i}\ \} \frac{P\{\ \omega_i\ \}}{P\{\ \omega_{\partial i}\ \}} \tag{4.13}$$

Substituting Equation 4.10 by Equation 4.11–4.13, we get

$$\begin{aligned}
P\{\ & x_i = \lambda_j, \omega_i, Y\ \} \\
&= P\{\ \vec{y}_i \mid x_i = \lambda_j\ \} P\{\ Y_i \mid \omega_i\ \} P\{\ x_i = \lambda_j, \omega_{\partial i}\ \} \frac{P\{\ \omega_i\ \}}{P\{\ \omega_{\partial i}\ \}} \\
&= \frac{P\{\ \vec{y}_i, x_i = \lambda_j\ \}}{P\{\ x_i = \lambda_j\ \}} P\{\ Y_i \mid \omega_i\ \} P\{\ x_i = \lambda_j, \omega_{\partial i}\ \} \frac{P\{\ \omega_i\ \}}{P\{\ \omega_{\partial i}\ \}} \\
&= \frac{P\{\ x_i = \lambda_j \vec{y}_i\ \}}{P\{\ \vec{y}_i\ \}} P\{\ \vec{y}_i\ \} \frac{P\{\ \omega_{\partial i},\ x_i = \lambda_j\ \}}{P\{\ x_i = \lambda_j\ \}} P\{\ Y_i \mid \omega_i\ \} \frac{P\{\ \omega_i\ \}}{P\{\ \omega_{\partial i}\ \}} \\
&= P\{\ x_i = \lambda_j \mid \vec{y}_i\ \} P\{\ \omega_{\partial i} \mid x_i = \lambda_j\ \} \frac{P\{\ \omega_i\ \} P\{\ \vec{y}_i\ \} P\{\ Y_i \mid \omega_i\ \}}{P\{\ \omega_{\partial i}\ \}}
\end{aligned} \tag{4.14}$$

Substituting Equation 4.9 by Equation 4.14 and deleting common factors from both the numerator and the denominator, we get

$$E(i, j, \omega_i) = \frac{P\{\ x_i = \lambda_j \mid \vec{y}_i\ \} P\{\ \omega_{\partial i} \mid x_i = \lambda_j\ \}}{\sum_{k=1}^{m} P\{\ x_i = \lambda_k \mid \vec{y}_i\ \} P\{\ \omega_{\partial i} \mid x_i = \lambda_k\ \}} \tag{4.15}$$

Let $\omega^c_{\partial i} = \omega_i - \omega_{\partial i}$ be the set of label assignments for the objects that are neither object i or its neighboring objects. Then, Equation 4.8 is rewritten as

$$\begin{aligned} P\{ x_i = \lambda_j \mid Y \} &= \sum_{\omega_{\partial i}} \sum_{\omega^c_{\partial i}} E(i, j, \omega_{\partial i})\, P\{ \omega_{\partial i}, \omega^c_{\partial i}, \mid Y \} \\ &= \sum_{\omega_{\partial i}} E(i, j, \omega_{\partial i}) \sum_{\omega^c_{\partial i}} P\{ \omega_{\partial i}, \omega^c_{\partial i}, \mid Y \} \\ &= \sum_{\omega_{\partial i}} E(i, j, \omega_{\partial i})\, P\{ \omega_{\partial i}, \mid Y \} \\ &= \sum_{\omega_{\partial i}} W_{\omega_{\partial i} \mid Y}\, E(i, j, \omega_{\partial i}) \end{aligned} \tag{4.16}$$

where $W_{\omega_{\partial i} \mid Y}$ stands for $P\{ \omega_{\partial i}, \mid Y \}$.

■

Lemma 1 is the basis to develop our relaxation transformation function. In the lemma, $E(i, j, \omega_{\partial i})$ denotes the support for the assignment of label λ_j to object i by a particular neighborhood label configuration $\omega_{\partial i}$. The lemma states the fact that, *a posteriori* probability $P\{ x_i = \lambda_j \mid Y \}$ is a weighted sum of supports from all the label configurations of object i's neighborhood. The weight factor $W_{\omega_{\partial i}|Y}$ for a neighborhood label configuration is computed by the probability that the configuration occurs under observation Y.

Lemma 1 provides a method to refine the estimate of $P\{ x_i = \lambda_{j_i} \mid Y \}$ in the context of an object's neighborhood. Applying Lemma 1 iteratively further incorporates indirect interactions of objects so that the estimate of $P\{ x_i = \lambda_{j_i} \mid Y \}$ can be recursively refined. This iterative probability update process can be seen as an artifact of filtering observation Y [45], which clearly shows the contextual effect of this relaxation procedure. Let $Y^{(0)}$ be the original observation Y, Equations 4.3 and 4.4 compute initial probability estimates. Applying Lemma 1 to all objects over all

possible label assignments once, we obtain probability estimates $P\{\ x_i = \lambda_{j_i}\ |\ Y^{(0)}\ \}$ under the unfiltered observation $Y^{(0)}$ and a filtered observation $Y^{(1)} = \{\vec{y}_i^{(1)}\ |\ 1 \leq i \leq \bar{n}\}$, where $P\{\ x_i = \lambda_{j_i}\ |\ \vec{y}_i^{(1)}\ \} = P\{\ x_i = \lambda_{j_i}\ |\ Y^{(0)}\ \}$. Applying Lemma 1 again results in probability estimates $P\{\ x_i = \lambda_{j_i}\ |\ Y^{(1)}\ \}$ over filtered observation $Y^{(1)}$. Denote the filtered observation after rth iteration as $Y^{(r)} = \{\vec{y}_i^{(r)}\ |\ 1 \leq i \leq n\}$. The *a posteriori* probability of the assignment of label λ_j to object i after rth iteration is

$$P^{(r)}\{\ x_i = \lambda_j\} = P\{\ x_i = \lambda_j\ |\ Y^{(r)}\} \tag{4.17}$$

Substituting Equation 4.17 into Equations 4.5 - 4.7 results in the final form of our relaxation transformation function, called transformation function $\mathcal{T}$.

$$P^{(r+1)}\{\ x_i = \lambda_j\ \} = \sum_{\omega_{\partial i} \in\ \Omega_{\partial i}} W_{\omega_{\partial i}}^{(r)}\ E^{(r)}(i, j, \omega_{\partial i}) \tag{4.18}$$

Where

$$E^{(r)}(i,\ j,\ \omega_{\partial i}) = \frac{P^{(r)}\{x_i = \lambda_j\ \}P\{\omega_{\partial i}\ |\ x_i = \lambda_j\}}{\sum_{k=1}^{m} P^{(r)}\{x_i = \lambda_k\ \}P\{\omega_{\partial i}\ |\ x_i = \lambda_k\}} \tag{4.19}$$

$$W_{\omega_{\partial i}}^{(r)} = P\{\omega_{\partial i}\ |\ Y\} \tag{4.20}$$

4.5 The Properties of The Scheme

For any object i, Equation 4.19 implies that the support for a label assignment λ_j by a configuration $\omega_{\partial i}$ is within the range of [0, 1] and the supports for all label

assignments under a configuration $\omega_{\partial i}$ sum up to 1, *i.e.*

$$\sum_{k=1}^{m} E(i,k,\omega_{\partial i}) = 1$$

From Equation 4.18, it is found that *a posteriori* probability $P^{(r+1)}\{\ x_i = \lambda_j \mid Y\ \}$ is within the range of $[0,1]$ and the sum of $P^{(r+1)}\{\ x_i = \lambda_j \mid Y\ \}$ for all label assignments of an object is 1.

$$\sum_{k=1}^{m} P^{(r+1)}\{\ x_i = \lambda_k \mid Y\ \} = 1 \tag{4.21}$$

This can be proved by induction on the iteration r in a straight forward way. From Equation 4.3 and Equation 4.4, Equation 4.21 holds for $r = 0$. For $r = l + 1$, from induction hypothesis, $\sum_{j=1}^{m} P^{(r)}\{x_i = \lambda_j\} = 1$ holds for $r = l$. Applying to Equations 4.18 and 4.19, the derivation is straightforward. This guarantees that $\mathcal{T}$ is a transformation from mn dimensional probability space to itself.

Transformation $\mathcal{T}$ are unbias based on the following theorems.

Theorem 4 (No information experiment): *When no information is learned by making observations on objects in the network, i.e.*

$$P\{x_i = \lambda_j \mid \vec{y}_i\} = P\{x_i = \lambda_j\} \tag{4.22}$$

we have

$$P^{(r)}\{x_i = \lambda_j\} = P\{x_i = \lambda_j\} \tag{4.23}$$

Proof: This can be proved by induction on r.

For $r = 0$, From Equation 4.22 we have $P^{(0)}\{x_i = \lambda_j\} = P\{x_i = \lambda_j\}$, Substitute this into Equation 4.19

$$E^{(0)}(i, j, \omega_{\partial i}) = \frac{P\{x_i = \lambda_j, \omega_{\partial i}\}}{P\{\omega_{\partial i}\}} \tag{4.24}$$

Since no information is learned from observation, thus, $\omega_{\partial i}$ is independent from Y.

$$P\{\omega_{\partial i} \mid Y\} = P\{\omega_{\partial i}\} \tag{4.25}$$

Substituting Equations 4.24 and 4.25 into Equation 4.18 results in

$$P^{(1)}\{x_i = \lambda_j\} = P\{x_i = \lambda_j\}$$

Similar approach can be applied to all other r's. ∎

Theorem 5 (Isolated objects): *When an object is independent from its neighboring objects, i.e.*

$$P\{x_i = \lambda_j, \omega\} = P\{\omega\}P\{x_i = \lambda_j,\} \tag{4.26}$$

then

$$P^{(r)}\{x_i = \lambda_j\} = P\{x_i = \lambda_j\}$$

Proof: Substitute Equation 4.26 into Equation 4.19, we have

$$E(i, j, \omega_{\partial i}) = P^{(r)}\{x_i = \lambda_j\}$$

So, from Equation 4.18, we get

$$\begin{aligned} P^{(r+1)}\{x_i = \lambda_j\} &= \sum_{\omega \in \Omega_{\theta_i}} P^{(r)}\{x_i = \lambda_j\} P\{\omega \mid Y\} \\ &= P^{(r)}\{x_i = \lambda_j\} \sum_{\omega \in \Omega_{\theta_i}} P\{\omega \mid Y\} \\ &= P^{(r)}\{x_i = \lambda_j\} \end{aligned}$$

■

Theorem 6 (All objects independent): *If all the objects are independent, i.e.*

$$P\{x_i = \lambda_j, \omega\} = P\{x_i = \lambda_j\} \prod_{x_k \in \omega} P\{x_k = \lambda_{j_k}\} \tag{4.27}$$

then

$$P^{(r)}\{x_i = \lambda_j\} = P\{x_i = \lambda_j\}$$

Proof: Similar to the proof for Property 5. ■

Chapter 5

The Development of a New Relaxation Algorithm

In the last chapter, based on MRF theorem, we derived a relaxation transformation $\mathcal{T}$ which provides a new approach for probabilistic relaxation labeling. The estimate of $P\{x_i = \lambda_j \mid Y\}$ is obtained by adding all the supports from object i's neighborhood configurations. However, to use this transformation, we need to address how to compute the weight $W_{\partial i \mid Y}$. Furthermore, transformation $\mathcal{T}$ is computationally intractable since it the number of neighborhood configurations is exponentially increased. In this chapter, we address these two problems. Maximum Entropy Estimate of $P\{\omega_{\partial i} \mid Y\}$ is used to estimate $W_{\partial i \mid Y}$. The dictionary method makes the new scheme useful for relaxation labeling.

5.1 The Estimation of $W_{\partial i \mid Y}$

Weight $W_{\partial i \mid Y}$ is a joint conditional neighborhood probability $P\{\omega_{\partial i} \mid Y\}$. Because the estimated marginal probability $P\{ x_i = \lambda_k \mid Y \}$ can be calculated from Equation 4.18 and 4.19, and the product of the marginal probabilities $P\{ x_i = \lambda_k \mid Y \}$ for all $x_i = \lambda_k \in \omega_{\partial i}$ has been proved to be the maximum entropy estimate of $P\{\omega_{\partial i} \mid Y\}$ (See [61]), estimating $P\{\omega_{\partial i} \mid Y\}$ from marginal probabilities is a reasonable choice. Therefore, the weight for a neighborhood label configuration is estimated by

$$W_{\omega_{\partial i}|Y} = P\{\omega_{\partial i} \mid Y\} = \prod_{l \in \partial i} P\{ \omega_{\partial i}(l) \mid Y \} \tag{5.1}$$

5.2 The Complexity of Relaxation Labeling

The complexity of transformation rule $\mathcal{T}$ is proportional to $\mid \Omega_{\partial i} \mid$, the number of configurations of an object's neighborhood. In the worst case, a neighborhood consists of all other objects in the system, then the number of neighborhood configurations, $\mid \Omega_{\partial i} \mid = m^n$, is exponential to the number of objects. This time complexity can be reduced considerably in two ways. First, the sizes of neighborhoods are usually much smaller than the size of the object set. *e.g.*, in the case of image processing with a neighborhood operation, neighborhood sizes are usually 3×3 or 5×5. However, even for a 3×3 neighborhood, the total number of configurations is m^9, where m is the number of labels in label set Λ. In real world situation, most applications are well-structured. Most of the configurations are physically impossible, and there are only a small number of *permissible configurations*. *Permissible configurations* are configurations which can occur in a given application in ideal situations. They are also called physically possible configurations. As an example, for a neighborhood of 3×3, with a set of 5 labels (four edge directions and a non-edge), the total number of configurations is $5^9 \approx 2 \times 10^6$. However, only 165 of them are permissible configurations. This suggests that, for an particular application, by constructing a dictionary with all permissible configurations, our scheme can be implemented efficiently.

5.3 Types of Configurations

Besides permissible configurations, there are other configurations that have significant influence on label assignments.

For example, in image edge detection, for a pixel i, we take the eight surrounding pixels as its neighboring pixels (See Figure 5.1a). All the possible one pixel edge patterns across this 3×3 area are permissible configurations. One permissible configuration in this setting is showed in Figure 5.1b. Here, arrows indicate directions of edge pixels and blanks non-edge pixels. The label set contains five labels: four different edge directions $\rightarrow, \uparrow, \leftarrow$ and $\downarrow$, and a non-edge label ϵ. This permissible configuration $\{x_0 = \epsilon, x_1 = \epsilon, x_2 = \epsilon, x_3 = \epsilon, x_4 = \uparrow, x_5 = \epsilon, x_6 = \uparrow, x_7 = \uparrow, x_i = \epsilon\}$ is an entry in the dictionary and has a conditional probability $P\{x_0 = \epsilon, x_1 = \epsilon, x_2 = \epsilon, x_3 = \epsilon, x_4 = \uparrow, x_5 = \epsilon, x_6 = \uparrow, x_7 = \uparrow \mid x_i = \epsilon\}$ associated with it.

Now, let's assign other labels for x_i in the same neighborhood setting, for example, $x_i = \uparrow$ in Figure 5.1c. It is possible that in the previous relaxation iterations, there is a label error for pixel 6, *i.e.* its label should be ϵ not $\uparrow$. Because this error may be corrected later in the relaxation process, the configuration provides a support to assign $\uparrow$ to pixel i as suggested by the configuration in Figure 5.1d. Thus, there are some configurations other than permissible configurations that have contributions to certain label assignments, *i.e.* the configuration in Figure 5.1c. We call them *possible supporting configurations.*

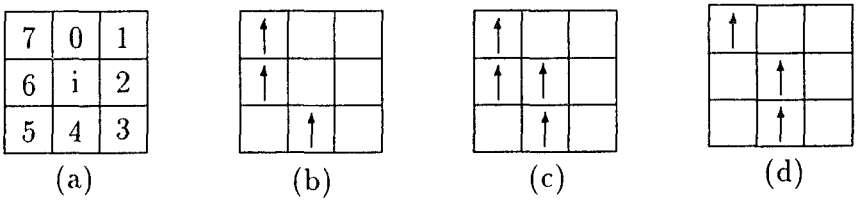


Figure 5.1: Possible supporting configurations.

Therefore, in our scheme, a dictionary contains two kinds of configurations:

1. *Permissible configurations* that occur in ideally labeled situations;

2. The configurations which would lead to some permissible configurations if some of its neighboring objects' labels change while the label for object i remains the same. We call this kind of configurations *possible supporting configurations.*

Permissible configurations and possible supporting configurations together are called Influential Configuration Set (ICS).

5.4 Dictionary Schemes

Kittler and Hancock ([27], [28], [29] & [45]) proposed a dictionary construction method. Each object i has a dictionary D_i that is constructed from all permissible configurations of object i's neighborhood ∂i. Dictionary D_i is further divided into m sections $D_i(\lambda_j)$ according to different label assignments for object i, $x_i = \lambda_j, j = 1, \ldots, m$. Let λ_{jl}^k be the label on object l, $l \neq i$ of the kth configuration in section $D_i(\lambda_j)$. The kth configuration in $D_i(\lambda_j)$ is denoted as

$$I_i^k(\lambda_j) = \{x_i = \lambda_j, x_l = \lambda_{jl}^k, l \in \partial i\} \tag{5.2}$$

Associated with every permissible configuration is a probability $P\{x_i = \lambda_j, x_l = \lambda_{jl}^k, l \in \partial i\}$. For physically impossible configurations, *i.e.*,

$$\{x_i = \lambda_j, x_j = \lambda_{jl}^k, l \in \partial i\} \notin D_i(\lambda_j),$$

their probabilities are set to be zero.

We observe that, when transformation function $\mathcal{T}$ is used, all the probabilities in the formulas are either *a priori* probabilities that remain unchanged during the relaxation process, or *a posteriori* probability estimates $P^{(r)}\{x_i = \lambda_j \mid Y\}$ that

are obtained from the previous iteration. Therefore, it is better to associate with each dictionary item the conditional probability $P\{\ x_j = \lambda_{j_l}^k, l \in\ \partial i\ \mid\ x_i = \lambda_j\ \}$. Transformation function $\mathcal{T}$ is the sum of supports from all configurations of an object's neighborhood. This sum can be further divided into two sums

$$\begin{aligned} P^{(s+1)}\{x_i &= \lambda_j\} \\ &= \sum_{\omega_{\partial i} \in\ \Omega_{\partial i}} W_{\omega_{\partial i}}^{(r)}\ E^{(r)}(i,j,\omega_{\partial i}) \\ &= \sum_{\omega_{\partial i} \in\ \Omega_{\partial i}^D} W_{\omega_{\partial i}}^{(r)}\ E^{(r)}(i,j,\omega_{\partial i}) + \sum_{\omega_{\partial i} \in\ \Omega_{\partial i}^C} W_{\omega_{\partial i}}^{(r)}\ E^{(r)}(i,j,\omega_{\partial i}) \quad (5.3) \end{aligned}$$

where $\Omega_{\partial i}^D$, the "don't care" configuration set [61], contains configurations such that $P\{\omega_{\partial i}\ \mid\ x_i = \lambda_j\} = P\{\omega_{\partial i}\}$ for all label assignment λ_j, *i.e.* under the neighborhood configuration $\omega_{\partial i}$, $P\{x_i = \lambda_j\}$ is independent from its neighbors. $\Omega_{\partial i}^C$, the "care" configuration set, is the complement of $\Omega_{\partial i}^D$, $\Omega_{\partial i} = \Omega_{\partial i}^C + \Omega_{\partial i}^D$. Configurations in $\Omega_{\partial i}^C$ are in favor of changing the probability $P\{x_i = \lambda_j\}$. Equation 5.3 provides an approach to implement the relaxation scheme. However, this implementation has inherent difficulties. Because the number of configurations $\mid \Omega_{\partial i} \mid$ is huge. For example, a $3\ \times 3$ neighborhood with five labels has $\mid \Omega_{\partial i} \mid = 5^9 \approx\ 3 \times 10^6$ configurations. If $\mid \Omega_{\partial i} \mid \approx \mid \Omega_{\partial i}^C \mid$, the computation is very expensive since there are large number of configurations in the calculation. Usually we prefer to have $\mid \Omega_{\partial i} \mid >> \mid \Omega_{\partial i}^C \mid$ *i.e.*, we choose only a few "care" configurations that have significant contributions to label assignments. Then, the expense to calculate $\mathcal{T}$ is acceptable. However, the rate of convergence will be very slow because $\mid \Omega_{\partial i}^D \mid >> \mid \Omega_{\partial i}^C \mid$, *i.e.* the factor for change $\sum_{\omega_{\partial i} \in \Omega_{\partial i}^C}$ is much smaller than the factor for stable $\sum_{\omega_{\partial i} \in \Omega_{\partial i}^D}$.

Another method has been proposed [27] that considers $P\{x_i = \lambda_j, \omega_{\partial i}\}$ as zero for configuration $\omega_{\partial i}$ that has little contribution to label refinement, for example, those in $\Omega^D_{\partial i}$. In this way, we get a small set of configurations that have significant influences to label assignments $\Omega^I_{\partial i}$. Then, we have

$$P^{(r+1)}\{\ x_i = \lambda_j\ \} = \sum_{\omega_{\partial i} \in\ \Omega^I_{\partial i}} W^{(r)}_{\omega_{\partial i}}\ E^{(r)}(i, j, \omega_{\partial i}) \tag{5.4}$$

However the $\Omega^I_{\partial i}$ that consists of permissible configurations only is not suitable for our relaxation scheme. Following is the transformation function that Kittler and Hancock used for their evidence combining process:

$$P^{(r+1)}\{\ x_i = \lambda_j\ \} = \frac{P^{(r)}\{\ x_i = \lambda_j\ \}\ Q^{(r)}\{\ x_i = \lambda_j\ \}}{\sum_{k=1}^{m} P^{(r)}\{\ x_i = \lambda_k\ \}\ Q^{(r)}\{\ x_i = \lambda_k\ \}} \tag{5.5}$$

Where

$$Q^{(r)}\{x_i\ =\ \lambda_j\} = \frac{1}{P\{x_i = \lambda_j\}} \sum_{\omega_{\partial i} \in\ \Omega_{\partial i}} \left[\{ \prod_{(x_k = \lambda_{j_k})\ \in\ \omega_{\partial i}} \frac{P^{(r)}\{x_k = \lambda_{j_k}\}}{P\{x_k = \lambda_{j_k}\}} \} P\{\ x_i = \lambda_j, \omega_{\partial i}\} \right] \tag{5.6}$$

For each object i, Equation 5.6 is used to add together the supports for a label assignment λ_j from all permissible configurations, and hence to obtain supporting function value Q for that label. Normalization is performed by Equation 5.5 after supports for all labels are found. A dictionary of permissible configurations works for this scheme.

In our scheme (transformation function $\mathcal{T}$), normalizations for all possible label assignments are performed for each configuration. Then, the normalized supports for a particular label assignment from all the configurations are added together. Normalization insures proper distributions of supports from a neighborhood setting to all possible label assignments. However, in many applications, most neighborhood settings have no, or only one, permissible configuration. This makes the distribution of supports by Equation 4.19 inefficient, hence deteriorates the power of the relaxation process. For example, in the edge detection problem we mentioned above, with those 165 configurations, there are 149 neighborhood settings. Only 16 of them have two permissible configurations. Others have only one. Thus, the dictionary constructed by K&H's method doesn't work well with our scheme.

A suitable dictionary for our scheme is a dictionary that contains all the configurations for the neighborhood settings that are obtained from permissible configurations and this is our ICS $\Omega^I_{\partial_i}$. In the example of edge detection, the label set has five labels. Therefore, each neighborhood setting contains five different configurations. Totally, the new dictionary, called D'_i, has $5 \times 149 = 745$ configurations in which 165 of them are permissible configurations and the rest are possible supporting configurations.

The dictionaries $D'_i, i = 1, \ldots, n$ are then constructed from all these permissible configurations and possible supporting configurations. Dictionary D'_i is a table with s rows and m columns, where s is the number of neighborhood settings and m is the number of possible labels. $D'_i(\lambda_j)$ denotes a column corresponding to the assignment of label λ_j to object i. Let λ'^k_{jl} be the label on object $l, l \neq i$, of the kth neighborhood configuration in column $D'_i(\lambda_j)$. The kth configuration in $D'_i(\lambda_j)$ is

$$C_i^k(\lambda_j) = \{x_l = \lambda_{j_l}^k, l \in \partial i \mid x_i = \lambda_j\} \tag{5.7}$$

The probability associated with dictionary item $C_i^k(\lambda_j) = \{x_l = \lambda_l^k, l \in \partial i \mid x_i = \lambda_j\}$ is $P\{x_l = \lambda_l^k, l \in \partial i \mid x_i = \lambda_j\}$, the conditional probability that configuration $\{x_i = \lambda_j, x_l = \lambda_l^k, l \in \partial i\}$ occurs when $x_i = \lambda_j$.

The new dictionary scheme can be derived from Kittler and Hancock's scheme. First, we calculate *a priori* probability $P\{x_i = \lambda_j\}$ by adding together the probabilities of all permissible configurations in $D_i(\lambda_j)$.

$$P\{x_i = \lambda_j\} = \sum_{I_i^l(\lambda_j) \in D_i(\lambda_j)} P\{I_i^l(\lambda_j)\} \tag{5.8}$$

Then, the probability for each permissible configuration in $D_i'(\lambda_j)$ is obtained by

$$P\{C_i^k(\lambda_j)\} = \frac{P\{I_i^k(\lambda_j)\}}{P\{x_i = \lambda_j\}} \tag{5.9}$$

Given all the permissible configurations and their probabilities $P\{x_i = \lambda_j, x_l = \lambda_l^k, l \in \partial i\}$, we need to compute probabilities for possible supporting configurations. In the relaxation process, a possible supporting configuration occurs when labeling errors are presented. Thus, it is natural to consider possible supporting configurations as corrupted permissible configurations. In [29], Kittler and Hancock proposed a label error process for discrete relaxation. They derived formulas to estimate the probability for any possible configuration from permissible configurations in their attempt to develop a discrete relaxation algorithm. The idea in this estimation is to add together the likelihoods of the current label configuration with all the permissible configurations. We adopt this method to estimate the probabilities for possible

supporting configurations. By assuming that label errors occur with equal probability p_e, the likelihood of a possible supporting configuration $\{x_i = \lambda_j, \omega_{\partial i}\}$ with a permissible configuration $I_i^k(\lambda_j) \in D_i(\lambda_j)$ is calculated by

$$P\{x_i = \lambda_j, \omega_{\partial i} \mid I_i^k(\lambda_j)\} = (1 - p_e)^{|\partial i| - K(i,k)} p_e^{K(i,k)} \tag{5.10}$$

$| \partial i |$ is the number of neighborhood objects, $K(i,k)$ is the number of labels that are different between a possible supporting configuration $\omega_{\partial i}$ and a given permissible configuration $I_i^k(\lambda_j)$. This likelihood is called *neighborhood transition probability.* The probability of a possible supporting configuration is the summation of all neighborhood transition probabilities over all permissible configurations.

$$P\{x_i = \lambda_j, \omega_{\partial i}\} = \sum_{I_i^l(\lambda_j) \in D_i(\lambda_j)} P\{x_i = \lambda_j, \omega_{\partial i} \mid I_i^k(\lambda_j)\} P\{I_i^l(\lambda_j)\} \tag{5.11}$$

To summarize, the procedure of our dictionary construction is as follows:

1. Find all permissible configurations in the application. Assign each permissible configuration a probability;

2. Find all possible supporting configurations from the set of permissible configurations;

3. Use label error process, *i.e.* Equation 5.10 and Equation 5.11, to calculate probabilities for all possible supporting configurations;

4. Use all permissible configurations and possible supporting configurations to construct the dictionary D_i'.

5.5 Outline Of The Algorithm

Based on the transformation $\mathcal{T}$ developed in the last chapter, the estimation of $W_{\partial i \mid Y}$, and the dictionary of ICS in this chapter. the relaxation labeling algorithm is depicted in Figure 5.2. First, an initial estimate $P^{(0)}\{x_i = \lambda_j\}$ for $P\{x_i = \lambda_j \mid Y\}$ is calculated from observation vector $\vec{y}_i$ for each object i by Equation 4.19 and 4.20. Based on $P^{(0)}\{x_i = \lambda_j\}$, an initial label assignment is selected by maximizing *a posteriori* probability (Equation 4.18). The result is a rough label assignment for the object system V. Then, transformation $\mathcal{T}$ is recursively applied to vector $P^{(r)}$, $r \geq 0$ to obtain the new vector $P^{(r+1)}$ which provides probability estimate of label assignment over larger contextual area. Configurations in ICS and their associated conditional probabilities are obtained by looking up dictionary $D_i(\lambda_j)$ for object i and its assigned label λ_j. In each relaxation iteration, label assignments are updated by MAP rule (Equation 4.18). This procedure ends with refined label assignments.

Because of the locality of the new algorithm. Efficient parallel implementationto on mesh-connected SIMD machine is easy.

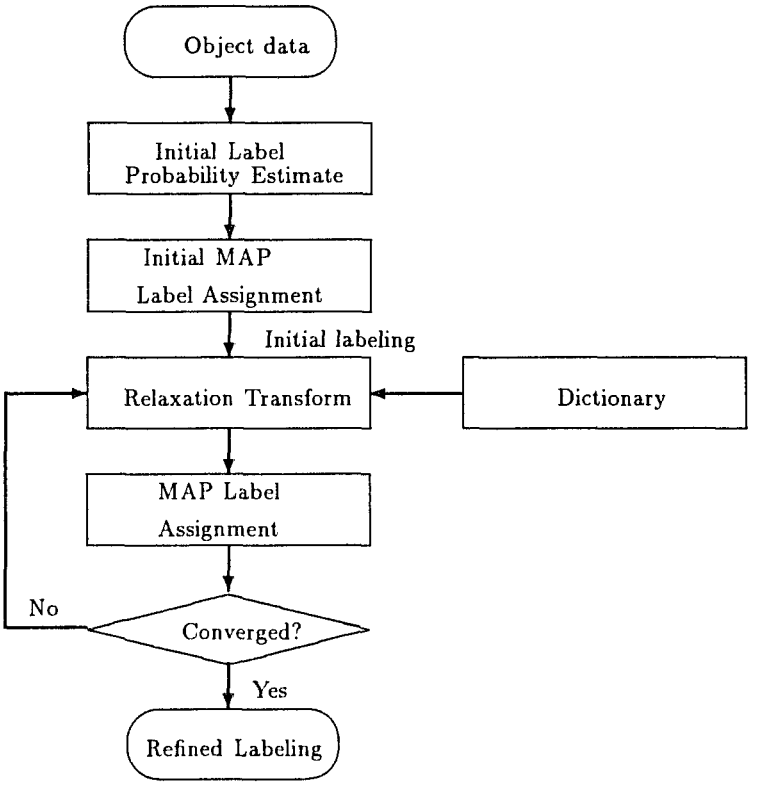


Figure 5.2: The relaxation algorithm.

Chapter 6

Edge Detection: A Case Study

6.1 Introduction

Edge detection plays an important role in computer vision tasks, and has received considerable attention in image processing literature. An edge corresponds to intensity discontinuities in an image. For most machine vision tasks, an edge map is sufficient to conduct

further processes such as motion analysis and object recognition. Edges mainly correspond to boundaries of objects of a scene. They may also correspond to images of shadows or surface marks [57], or the results of noise or blurring. A variety of edge detectors have been proposed. Most of them perform reasonably well for simple noise free images, but tend to fail for noisy images. In our opinion, image smoothing is not the solution. A better way is to make use of edge contextual information.

The ultimate goal of edge detection is to characterize intensity changes of an image in terms of physical processes that originate them [46]. It is commonly believed that, to achieve this goal, at least two stages are required: the characterization of intensity changes, and the use of structural and high-level knowledge to find real boundaries.

Intensity changes are detected by differentials of intensity functions. The local maximum of the first order intensity differential and the zero-crossing of the second order intensity differential are the two commonly used characteristics. The results of

these differentiation operators are rough edge maps that describe intensity changes of an image. Various techniques have been presented in the literature. Robert's operator [64] and Sobel's operator [74] are examples of these simple edge detection operators.

Canny [7] formulated edge detection problem as an optimization problem. He put forward three objective criteria—good detection, good localization, and minimum false alarms— to define an optimal filter. He obtained an optimal 1-D operator for step edge detection and found that this optimal operator can be efficiently approximated by the first derivative of Gaussian function. These three criteria were also used by other authors, notably R. Deriche [19] and Shen & Castan ([70, 71, 46]), to extend the design of optimal filters. The advantage of these two extensions is the recursive feature of the filters. Recursive technique provides an efficient way for image filtering. Both methods use infinite extent filters. Deriche's filter is an infinite extension of Canny's optimal filter and requires five multiplications and five additions for each pixel. Shen and Castan's filter is even more efficient. It is an infinite exponential filter that requires only four multiplications and nine additions for each pixel. As a resemblance to receptive profile of simple cells in mammalian visual systems, Gabor filters have attracted attention recently ([31, 53]). Gabor filters are modulation products of Gaussian and sinusoidal signals. Based on Canny's optimal criteria, Mehrotra et al. [53] discovered the best performance Gabor odd filter and developed an edge detection algorithm based on the filter. Hancock, in his paper [31], used two filters, a modified Gabor odd filter to detect lines, and a modified Gabor even filter to detect step edges. However, all the above mentioned filters, more or less, have the effect of blurring edges, especially edge junctures. A promising approach to solve this problem is to use nonlinear image filters that encourage

intra-region smoothing in preference to inter-region smoothing ([58] and [62]). Perona and Malik [62] proposed anisotropic diffusion for image filtering. The technique is similar to heat flow diffusion phenomenon in physics. Backward flows occur in boundary areas and sharpen edges while regions inside boundaries are smoothed. Nitzberg and Shiota [58] further extended this technique. They used regulation to guarantee that diffusion equations had solutions and that corners and T junctures were enhanced.

Although many improvements have been made on image filters (or intensity differentiation operators), using any image filter alone is not sufficient to obtain good edge detection results, especially in noisy situations. A reason is that most filters use models of a single isolated edge. However, the quality of edge detection should not be determined by small differences in smoothing functions. Postprocessing, therefore, is required to further refine rough edge maps obtained from intensity differentiation. One of the techniques to refine rough edge maps is edge tracing ([87, 82]). Wu, Iyengar and Min [87] investigated edge detection using gradient directional information. In their algorithm, a pixel adjacent to a detected edge pixel, whose magnitude exceeds a given threshold, and whose direction is not perpendicular to that of the edge pixel, is considered as another edge pixel. The algorithm works fine after eliminating many small edges whose length (the number of pixels in an edge) is less than a *ad hoc* threshold value. Ungureanu et al [82] designed another tracing algorithm that used two bar like control windows. These two windows are perpendicular to each other and are used to walk through edges of an image. They further discussed the VLSI implementation of their algorithm that provided realtime edge refinement. The problem with these approaches is that, they are insensitive to

weak edges, and if an edge has a pixel whose magnitude is less than the threshold, they will cut the edge into two smaller edges.

Another technique of edge map refinement is to make use interaction between edges. Chen and Medioni [10] proposed an edge interaction model to capture interactions between edges within a small neighborhood area. Initialized with zero-crossings of the signals convolved with a LOG filter, their method iteratively finds new and more accurate edge location by conveying the information from strongly interacting edges. This method yields good result despite the problems of its over-simplified model, its large mask, and its slow convergence rate. Haralick & Lee [33] and Higgins & Hsu [37] also used structural information of neighborhood area to extract edge pixels.

A prospective technique for postprocessing in edge detection is relaxation labeling. Their ability to convey not only local but also global contextual information from interacting objects makes it a good candidate for edge detection. Kittler and Hancock ([27, 30, 28, 31, 45]) conducted intensive studies on the application of probabilistic relaxation labeling to edge detection. Their approaches employ dictionaries of permissible local edge configurations. A pixels along with its neighborhood is compared with these permissible configurations to estimate the probability that it is assigned certain label. The goal of their algorithms is to find the globally consistent maximum *a posteriori* probability (MAP) estimate to assign a unique label to each pixel. Noise is modeled as a source of inconsistencies. Interactions among label assignments of pixels are used to eliminate these inconsistencies. However, their method produces good results only in lower signal to noise ratio(SNR) situation. Furthermore, relaxation labeling as a general label assignment framework has a higher time complexity, and takes more time than some other techniques such as

the tracing techniques. However, relaxation labeling methods have their advantages. With the ability to link edge segments in local contexts, they produce better edge connectedness. More important, they are easy to be parallelized.

In this chapter, we investigate the problem of using relaxation labeling as a post-processing method in edge detection. We propose two new dictionary based relaxation labeling algorithm that has a better noise-suppression performance than Kittler and Hancock's evidence combining formulas. We first demonstrate the power of the new relaxation labeling method by comparing the results with Kittler and Hancock's algorithm under their assumption that noise is Gaussian distributed. Then, we discuss the issue that initial probability estimate for label assignment is very important to obtain good results for relaxation labeling algorithms, and present a new initial estimation method that based on histograms of image intensity changes. The advantages of the new method are its robustness to noise, its preservation of corners and T-junctures, and its output edge connectedness.

6.2 Relaxation Based Edge Detection

In traditional edge detection, differentials are used. However, differentiations are very sensitive to noise. Although this problem can be eased by smoothing, smoothing can also eliminate edge features and degrade resolution capabilities of edge detectors simultaneously.

Using relaxation labeling as a postprocessing step is a prospective solution. First, a traditional differential operator is employed to obtain an initial edge assignment for every pixel. This edge detector should preserve as many edges as possible. A dictionary of configurations in 3×3 neighborhood of each pixel is then constructed

and used in probabilistic relaxation labeling algorithm to correct the erroneously labeled pixels. The effect of this postprocessing is to remove noise and to acquire the refined one pixel wide edge map of the given image.

To develop an edge detection algorithm from the dictionary based relaxation scheme, two problems need to be addressed:

1. How to calculate initial label probability estimates;
2. How to find the configurations for the dictionary and to calculate the *a priori* conditional probabilities associated with them.

First, we assume that image noise are Gaussian distributed and use the smallest differential operators(1×2 and 2×1 windows). The smallest differential operations are chosen because:

1. This enables us to test the noise insensitivity and the robustness of our relaxation algorithm to the greatest extent;
2. We can compare our algorithm with Kittler and Hancock's algorithm in an identical situation.

Each pixel (u, v) had an observation vector $\vec{y}_{(u,v)}$ with two first order partial differentials c_u and c_v of the observed intensity function $g'(u, v)$,

$$c_u = g'(u+1, v) - g'(u, v)$$

$$c_v = g'(u, v+1) - g'(u, v)$$

The Gaussian noise is assumed to have a zero mean and a standard deviation of σ and is independent of image intensity $g(u, v)$. For non-edge pixel (u, v), pixels

$(u+1,v),(u,v+1)$ and (u,v) belong to the same image segment and should have the same standard deviation σ. Then, *a priori* probability $P\{c_u, c_v \mid x = \epsilon\}$ is calculated by,

$$P\{c_u, c_v \mid x = \epsilon\} = \frac{1}{2\sqrt{3}\pi\sigma^2} e^{\frac{c_u^2 + c_v^2 - c_u c_v}{3\sigma^2}} \tag{6.1}$$

Initial label probabilities are then computed in two steps. First, the *a posteriori* probabilities for non-edge labels, $P\{x = \epsilon \mid c_u, c_v\}$, are calculated from $P\{c_u, c_v \mid x = \epsilon\}$. However, because the mixture density function $P\{c_u, c_v\}$ is unknown, $P\{c_u, c_v\}$ is maximized. With this maximization,

$$P\{x = \epsilon \mid c_u, c_v\} \geq e^{\frac{c_u^2 + c_v^2 - c_u c_v}{3\sigma^2}}.$$

Take the lower limit as the initial label probability estimates for non-edge pixels. The residual is then apportioned among the four edge labels as follows:

$$P_{resid} = 1 - e^{\frac{c_u^2 + c_v^2 - c_u c_v}{3\sigma^2}}$$

Let $\Delta = \mid c_u \mid + \mid c_v \mid$, we have

$$\begin{aligned}
P\{x = \rightarrow \mid c_u, c_v\} &= -\frac{c_v}{\Delta} P_{resid} && \text{if } c_v \leq 0; \\
P\{x = \uparrow \mid c_u, c_v\} &= -\frac{c_u}{\Delta} P_{resid} && \text{if } c_u \leq 0; \\
P\{x = \leftarrow \mid c_u, c_v\} &= \frac{c_v}{\Delta} P_{resid} && \text{if } c_v > 0; \\
P\{x = \downarrow \mid c_u, c_v\} &= \frac{c_u}{\Delta} P_{resid} && \text{if } c_u > 0;
\end{aligned}$$

The above initial label probability estimation method was introduced by Kittler and Hancock[27]. The reason we adopt this method is to make an identical situation where we can compare the performances of both algorithms.

In Chapter 5, we have detailed how to construct a dictionary. To form the dictionary for edge detection, we need to set up the criteria for permissible configurations. A permissible configuration is a configuration in 3×3 lattice with one continuous single pixel wide edge.

Five labels are used to classify pixels. They are ϵ for non-edge pixel, $\uparrow$ for upward edge pixel, $\rightarrow$ for rightward edge pixel, $\downarrow$ for downward edge pixel, and $\leftarrow$ for leftward edge pixel. The criteria to find the permissible configurations are:

1. Edges are all closed;

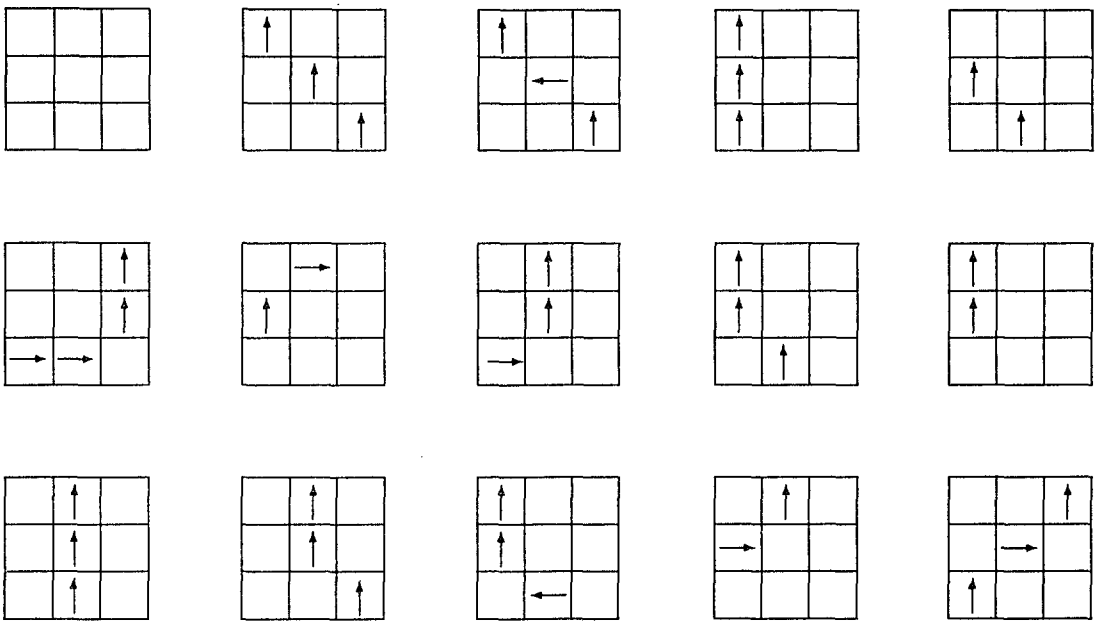
2. Edges are all one pixel wide;

3. Edges are all continuous.

165 permissible configurations are found based on these criteria and can be obtained by manipulating the 15 configurations shown in Fig. 6.1 by reversal, reflection and rotation. 97 of them had label ϵ for the center pixel. And for each of the four edge labels assigned to center pixel, there were 17 permissible configurations. All permissible configurations were considered equally likely Thus, for each permissible configuration ω,

$$P\{x_i = \lambda_j, \omega\} = \frac{1}{165}$$

Out of these 165 configurations, 149 neighborhood settings are found. All of them are from the reversal, reflection and rotation of the 14 neighborhood settings

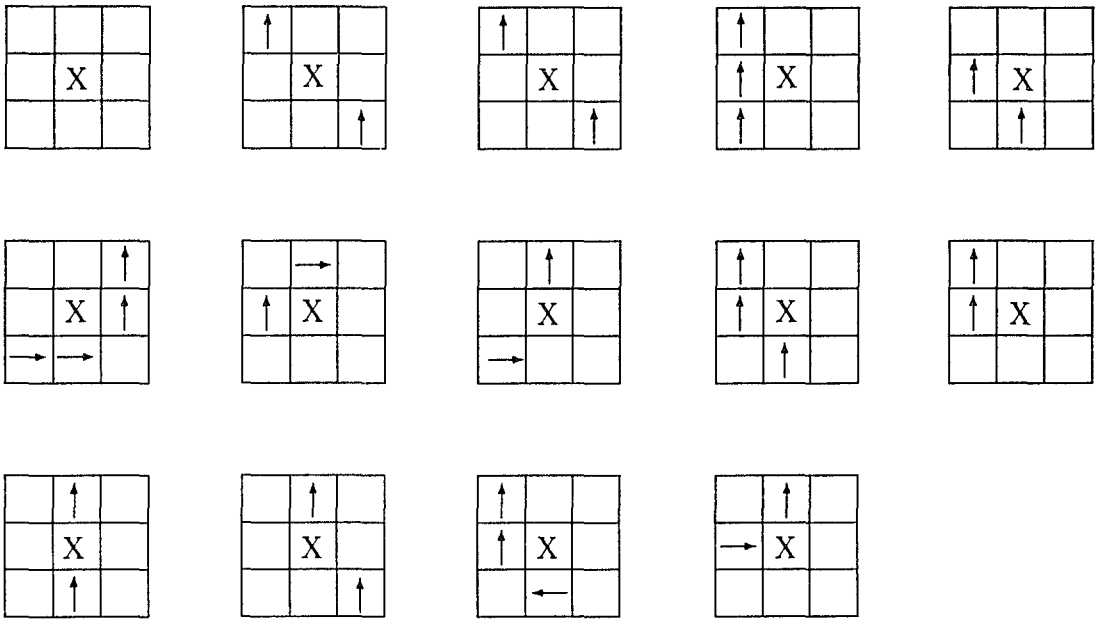


An empty cell indicates a non-edge pixel.

Figure 6.1: Permissible configurations.

in Fig. 6.2. Label "X" in these figures indicates a pixel can take any of the five labels in the label set.

Two kinds of images were used (an artifact image with additional Gaussian noise of different level, and some natural images) to compare the algorithm with K & H's [27]. The results showed that both methods are very good in preserving corner and edge connectivity, but our method has a better noise suppression capability. See [16] for details. However, one of our primary goal, to obtain single pixel edge, was not fully accomplished. For example, Figure 6.11b and Figure 6.11d are the edge outputs obtained from Kittler and Hancock's method and our relaxation method. In both cases, the edges around the fluorescent lamps are not very well constructed.



"X" indicates it can be any of the five labels.

Figure 6.2: Neighborhood settings.

6.3 The Problem of Initial Estimation

Although designing an fast convergent update rule is a major step in developing probabilistic relaxation algorithm, the problem of initial assignment is also crucial. It is true that applying contextual information efficiently through the update procedure can eliminate ambiguities from imprecise initial label assignments. However, if initial assignments contain too many label errors, label contextual information may not be enough to eliminate all of them. Indeed that was the problem of our Gaussianious initial label assignment estimation approach. More precisely, the problem stems from:

1. The assumption that noise is Gaussian distributed may not work in real world applications.

2. It only uses first order differences as the observation vector.

Based on these observations, we propose a new approach to compute initial label assignments. First, histogram $h(l)$ of first order difference of an image is used instead of Gaussian distribution assumption. l denotes the absolute change in gray level. Secondly, second order difference and first order difference are incorporated to obtain a better initial guess of label assignment.

For a particular image, its noise may not be Gaussian distributed. A histogram, in the other hand, is a statistics of a given image and better reflects the intensity distribution of the image. Thus, a histogram provides a better estimation. In our method, the histogram of first order difference of intensity level is used because the probability estimation of initial edge label assignment is based on intensity changes.

Zero-crossings of second order difference of an image has been proved to be a good estimate of edge points. If a pixel is not a zero-crossing point, this pixel is not an edge pixel. However, a zero-crossing point may not be an edge pixel. The degree of intensity changes and label contextual information are needed to further refine the edge map that was derived from zero-crossings.

To get zero-crossings of a given image, we adopt J. Shen and S. Castan's Exponential recursive filtering approach ([19], [70], & [46]). Their exponential filter (Figure 6.3) has infinitely large window size and can be realized by a simple and efficient recursive algorithm. An excellent feature of this filter is that the limited Laplacian of an input image filtered by this filter can be computed from the difference between the input and the output of the filter. Thus, second order difference of an image can be calculated efficiently. The 1-dimensional exponential filter has the form

$$f(x) = [\frac{a_0^2}{1-(1-a_0)^2}](1-a_0)^{|x|}$$

and can be implemented by two recurrent relation: First, scan from left to right using

$$y_1(i) = y_1(i-1) + a_0[x(i) - y_1(i-1)]$$

Then, scan from right to left using

$$y_2(i) = y_2(i-1) + a_0[y_1(i) - y_2(i-1)]$$

Since the filter is decomposable, a 2-dimensional exponential filter can be implemented by two 1-dimensional filters: one in x direction and the other in y direction. And the second order difference is calculated from the difference between filter output and filter input.

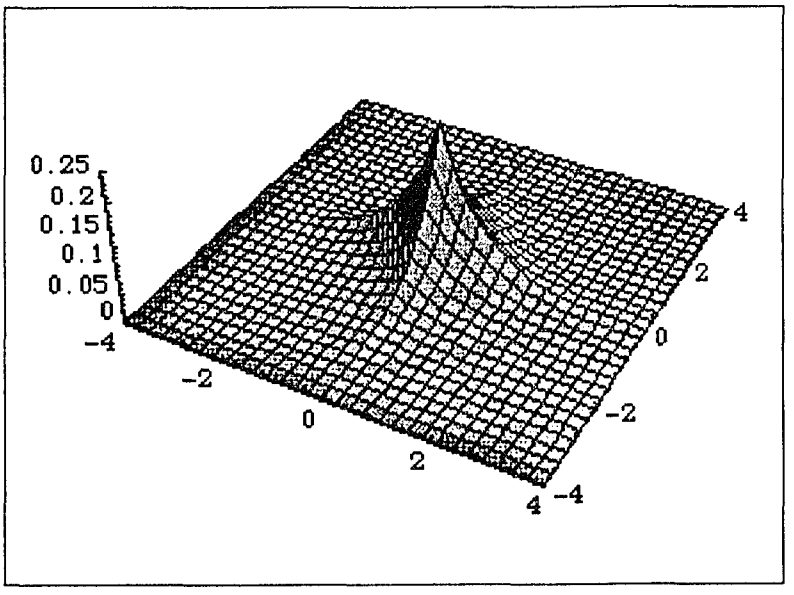


Figure 6.3: The exponential filter.

Binary Laplacian Image(BLI) is employed to find zero-crossings. A BLI is a binary image where a pixel gets value 1 (0) if the corresponding second order difference is non-negative (negative). The pixels lay in boundaries of 1-segments are zero-crossing pixels. Since very small, isolated 1(0)-segments in BLI are the results of random noise, an additional step is used to eliminate all small isolated segment, like those that have less than five pixels.

To get a better estimate of initial label assignment, a histogram $h(l)$ of the absolute intensity level changes of the image is calculated. This histogram is then used to estimated probabilities $P\{x_i = \text{“edge”}\}$ for initial assignments for the “edge” labels,

$$P\{x_i = \text{“edge”}\} = \begin{cases} 0 & x_i \text{ is not a boundary pixel of 1-segment} \\ 0 & \text{if } c_i \leq 6 \\ h(c) & \text{otherwise} \end{cases}$$

where $c_i = \max(\mid c_{x_i} \mid, \mid c_{y_i} \mid)$. and the estimated probabilities for initial assignments of the “non-edge” label are

$$P\{x_i = \text{“non-edge”}\} = 1 - P\{x_i = \text{“edge”}\}$$

If pixel i is not a zero crossing point, the probability that i's label is “edge” is zero. Otherwise, the intensity difference along x axis c_{x_i} and the intensity difference along y axis c_{y_i} is calculated. The maximum value c_i between the absolute value of c_{x_i} and that of c_{y_i} is used as the measurement of an intensity change for the pixel. We use maximum value c_i in the calculation instead of averaging $\mid c_{x_i} \mid$ and

$| c_{y_i} |$, because a significant intensity change in any direction is sufficient to consider the pixel as an edge pixel. It has been found that intensity changes below six gray levels (the *just noticeable difference* (JND)) in 256 gray level scaled images are not detectable by human eyes [85]. Therefore, in the case that c_i is less than or equal to gray level 6, the pixel is considered as a non-edge pixel.

To summarize, the procedure of label assignment initialization is shown in Figure 6.4. It first uses the exponential filter to calculate second order intensity difference of an image. With the help of the BLI, all zero crossing points are located. A histogram of the first order intensity difference of the image is then calculated and is used to assign initial probability estimates for initial label assignments. An initial label is assigned to pixel i by:

$$x_i = \begin{cases} \text{“edge”} & \text{If } P\{x_i = \text{“edge”}\} \geq P\{x_i = \text{“non-edge”}\} \\ \text{“non-edge”} & \text{Otherwise} \end{cases}$$

With the above method for initial label assignment, our second algorithm (Algo2) is developed using a different dictionary. Here, only two labels, “edge” and “non-edge” are used to classify pixels. No direction information is employed in the current implementation of the algorithm. The permissible configurations in this application are obtained from ideal edge maps where the three criteria listed in Section 6.2 are satisfied. Nine basic permissible configurations are found which are listed in Figure 6.5 (a dot in a pixel indicates an edge pixel). By rotation, reversal and reflection, we get 41 permissible configurations, out of which, 12 configurations support the assignment of “edge” label and 29 configurations support “non-edge” label. In this study, all permissible configurations are considered equally likely.

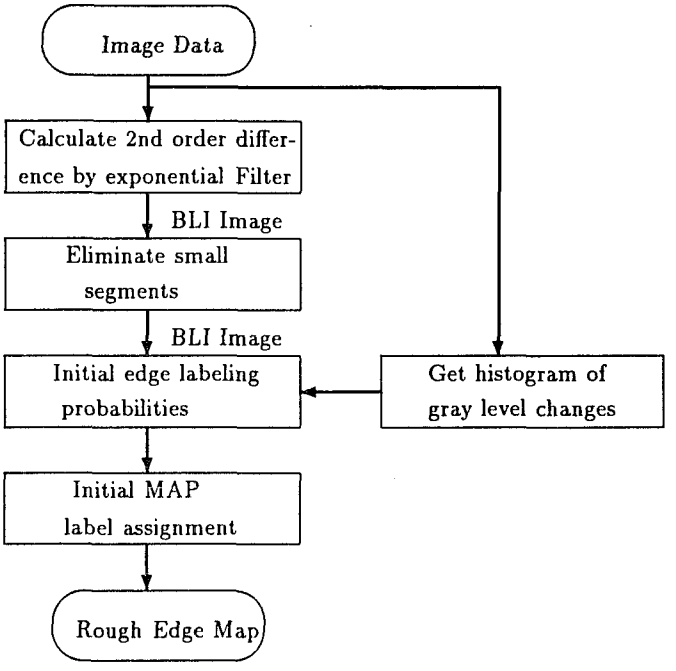


Figure 6.4: The initialization procedure.

Thus, each permissible configuration $\omega = \{x_i = \lambda_j, x_l = \lambda_l, l \in \partial i\}$ is associated with a probability of

$$P\{\omega\} = \frac{1}{41}$$

Each of these 41 configurations has a distinct neighborhood setting. Thus, there are 41 neighborhood settings and the dictionary has 82 configurations, 41 permissible configurations and 41 possible supporting configurations.

6.4 Experiments

To understand the performance of the new algorithm, we examine the behavior of the algorithm on natural images as well as artifact images. These experiments

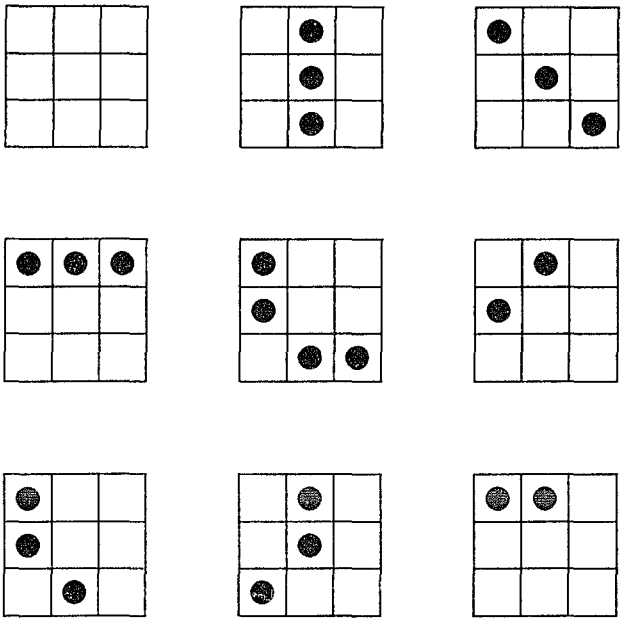


Figure 6.5: Permissible configurations for edge detection.

are intended to test the robustness of the new dictionary based probabilistic relaxation labeling algorithm. We compared our results with Kittler and Hancock's probabilistic relaxation algorithm (called K & H) and Shen and Castan's optimal exponential edge detector, the edge detection algorithm SDEF in image processing system **Khoros**. We developed two versions of our relaxation algorithm to examine the importance of initial label assignment estimation: One of them uses the initialization method proposed by Kittler and Hancock (Called **Algo1**); Algo1); The other follows the method described in last section (Called **Algo2**). All the algorithms are programmed on SiliconGraphics in C.

In the following presentation, for algorithms K & H, Algo1, and Algo2, all the edge outputs are collected after 10 iterations. The figures show the final maximum *a posteriori* label assignments. Black pixels correspond to non-edge pixels and white pixels are edge pixels. It should be noticed that all the results shown are obtained from initial probability assignments after applying the relaxation transformation

Table 6.1: Number of mislabeled pixels

σ	**20**	**60**	**100**	**140**	**180**
K & H	22	30	100	198	262
SDEF	65	48	61	77	68
Algo1	25	31	46	77	81
Algo2	49	37	41	14	32

functions 10 times. No postprocessing like linking, thinning, or cleaning, *etc.* is done.

A well structured simulated image (50 $\times$ 50 in dimension with a square and a circle) is used. Within the circle, the gray level is 56. Outside the circle but inside the square, the gray level is 231. Outside the square, the gray level is 115. This perfect image is then mixed with independent Gaussian noises using **Khoros**. These noises have a mean of zero and standard deviations of 20, 60, 100, 140, and 180 respectively. The artifact image and its standard one pixel-wide eight-connected edge map are shown in Figure 6.6a. This is to test the performance of the algorithms under different noisy situations and its ability to detect edges of various orientations and edges with high curvature. Figure 6.6 shows the result of these algorithms. In these figures, a black pixel indicates a correct labeling of an edge pixel. A red pixel is a pixel mislabeled as an edge pixel. A light blue pixel is a pixel mislabeled as a non-edge pixel.

For noise suppression, both SDEF and Algo2 have good noise resistance, and work consistently under different noise level. The performances of K & H and Algo1 are affected by noise. Noise with higher standard deviation causes more labeling errors. K & H method obtains more error labeled edge-pixels. Table 6.1 shows the number of mislabeled pixels for all the output images.

Table 6.2: Number of break points

σ	20	60	100	140	180
K & H	0	1	0	0	0
SDEF	4	5	3	6	3
Algo1	0	0	1	0	1
Algo2	0	0	0	0	1

For edge connectedness, if error labeled edge pixels are ignored, K & H and Algo2 capture the contours of the standard edge map (Figure 6.6a) quite precisely and the results are almost free of distortions. Algo1 obtains the standard contour without distortion when $\sigma = 20$ and $\sigma = 60$. The edge outputs are distorted for $\sigma \geq 100$ and break points are also presented. SDEF has break points in all the cases and the contour for the circle tend to deviates from the standard edge map. Figure 6.7 that shows the enlarged edge maps for $\sigma = 100$ clearly demonstrates the correctness of the edge outputs. Table 6.2 summarizes the number of breaks in each case.

The differences in both noise resistance and contour perfectness between the outputs from Algo1 (Figure 6.6d) and those from Algo2 (Figure 6.6e) reveal the importance of label assignment initialization.

To assess the effectiveness of our method to correctly label edges for natural images, four images from an image base in the University of Massachusetts are used (Figure 6.8).

Figure 6.9 is the edge maps for the office scene (Figure 6.8a). For the simple patterns on the wall. SDEF and Algo2 obtain clear one pixel wide edges. However, the outputs from K & H and Algo1 are not one pixel wide. An example of weak contrast edges is the seat of the couch. Figure 6.10 highlights this portion of the output. The outputs from K & H and Algo2 capture more weak contrast edges.

The results of processing Figure 6.8c are shown in Figure 6.11. K & H's method acquires more edges and also retains more noisy pixels. SDEF and Algo2 obtain clearer edge maps, especially in the areas near the fluorescent lamps. However, these two methods fail to capture the juncture between the left wall and the ceiling. SDEF also fails to capture the junctures between the wall and the floor.

Figures 6.8b and 6.8d are two more examples of natural images, where all methods perform reasonable well. In both cases, K & H method retains more noise. K & H and Algo1 cannot obtain one pixel wide edges in some areas, the wheels of the car and the eaves of the house. These two images also reveal that relaxation methods, through the use of neighborhood label context, achieve better edge connectedness. This is demonstrated by the stripes on the body of the car and the eaves of the lower roofs of the house, where the relaxation methods obtain connected lines while SDEF gets dashes.

To summarize, these experiments show that:

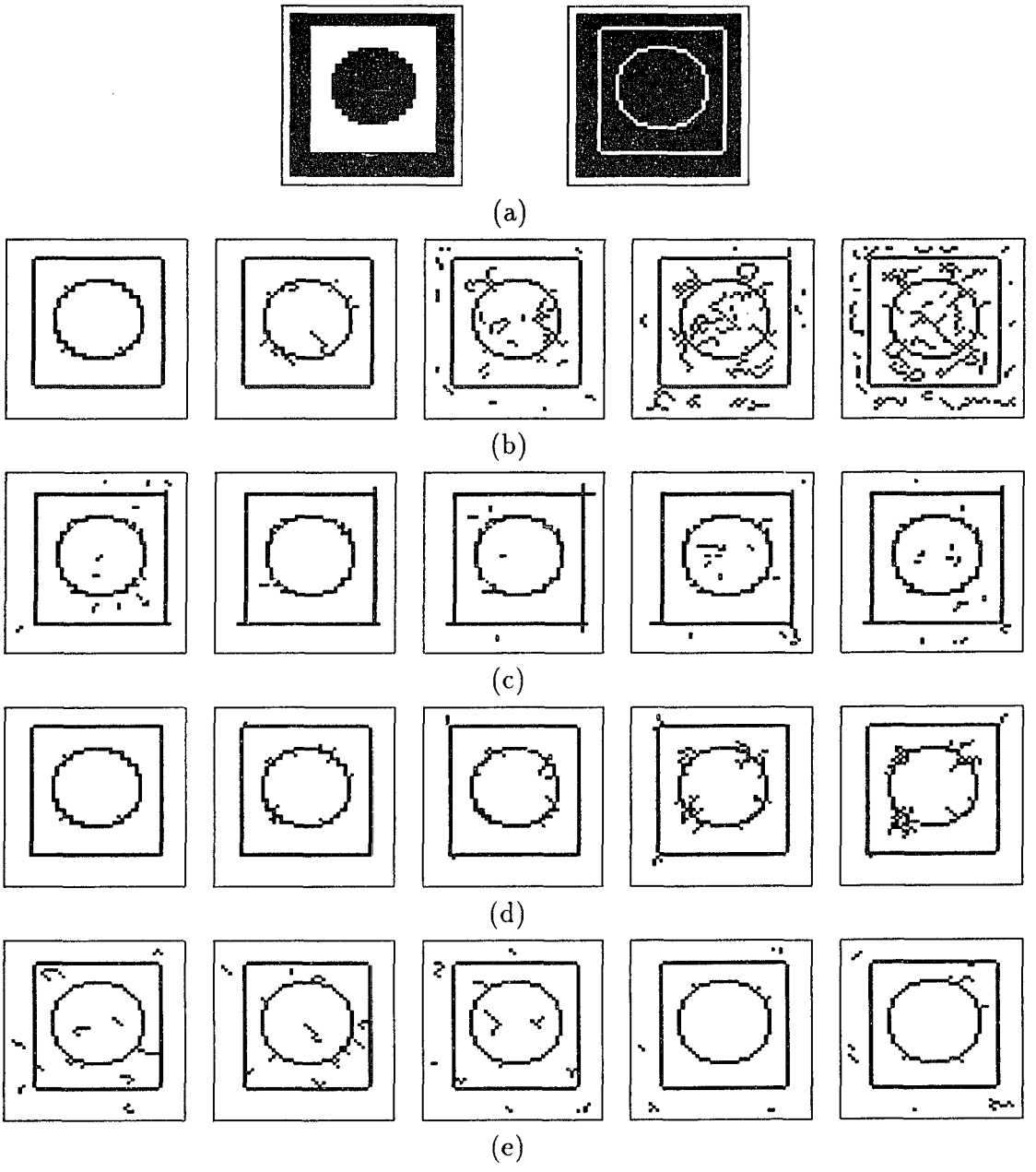
1. SDEF and Algo2 have better noise resistance.

2. Relaxation methods, K & H, Algo1, and Algo2, obtain better edge connectedness and better contour.

3. K & H and Algo2 achieve better weak edge detection.

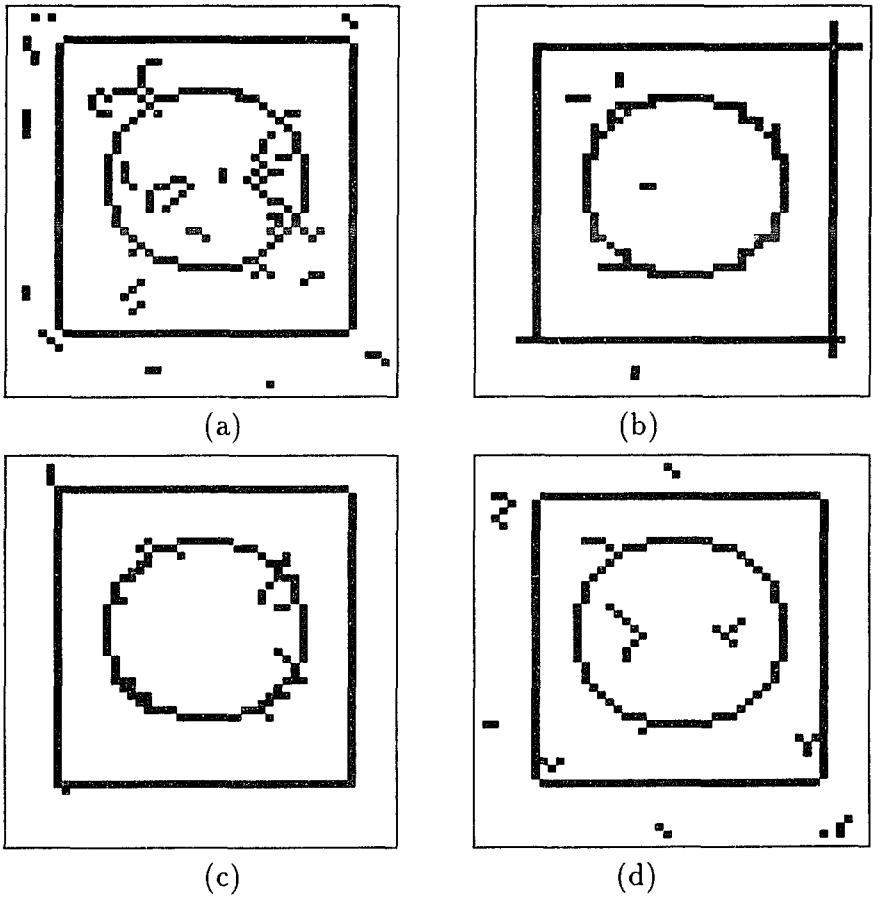
4. The estimation of initial label assignments is very important. The results from Algo2 is much better than those from Algo1.

5. An important feature of both Kittler and Hancock's algorithm and our algorithms is the rate of convergence. After 10 iterations, the relaxation processes essentially converge.



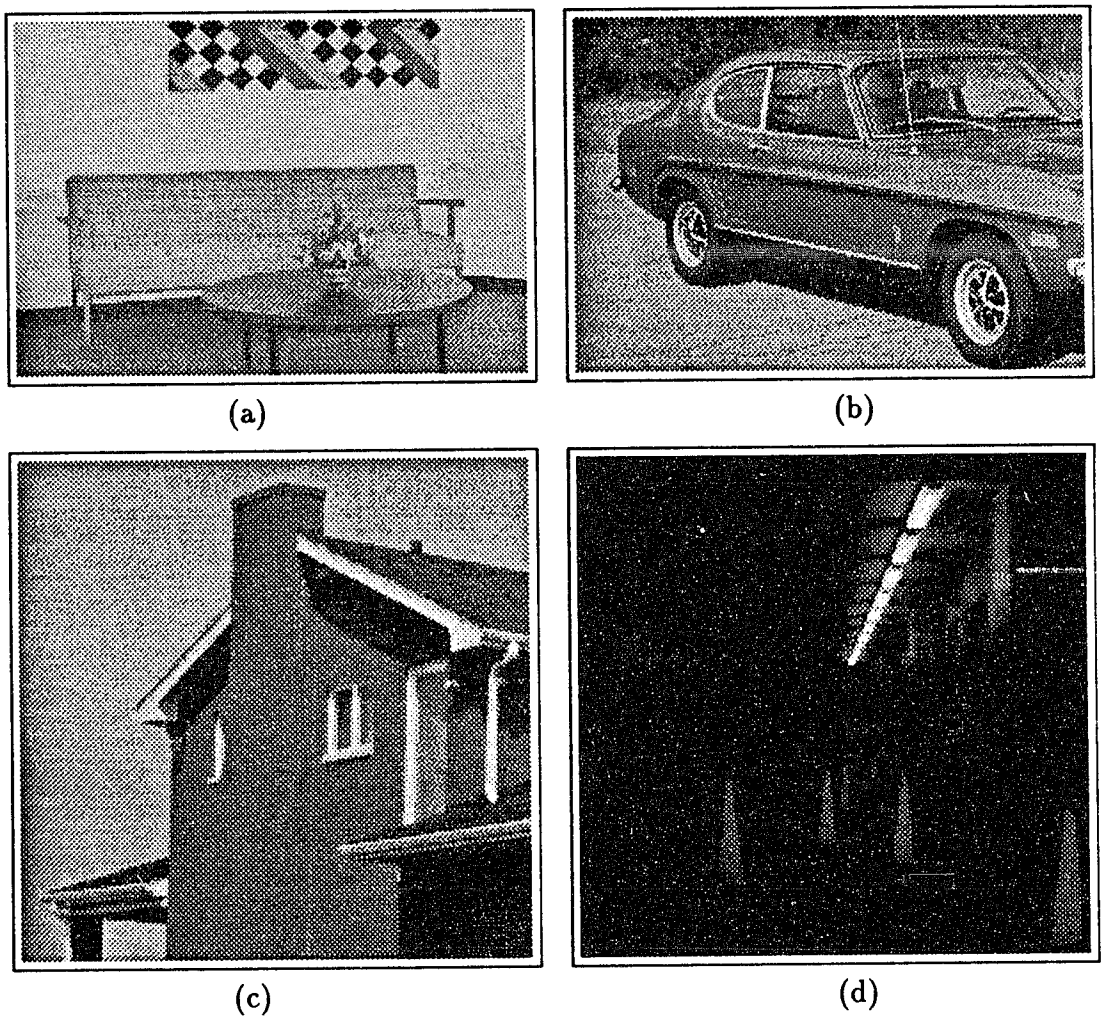
Synthetic images are with additional Gaussian noise(σ = 20, 40, 60, 80, 100, 120, 140, and 180. (a) is the original image without noise; (b) is the results of Kittler and Hancock's algorithm; (c) is the results of Shen and Castan's algorithm; (d) is the results of our first algorithm, and (e) is the results of our new algorithm.

Figure 6.6: Experiment results for the synthetic image.



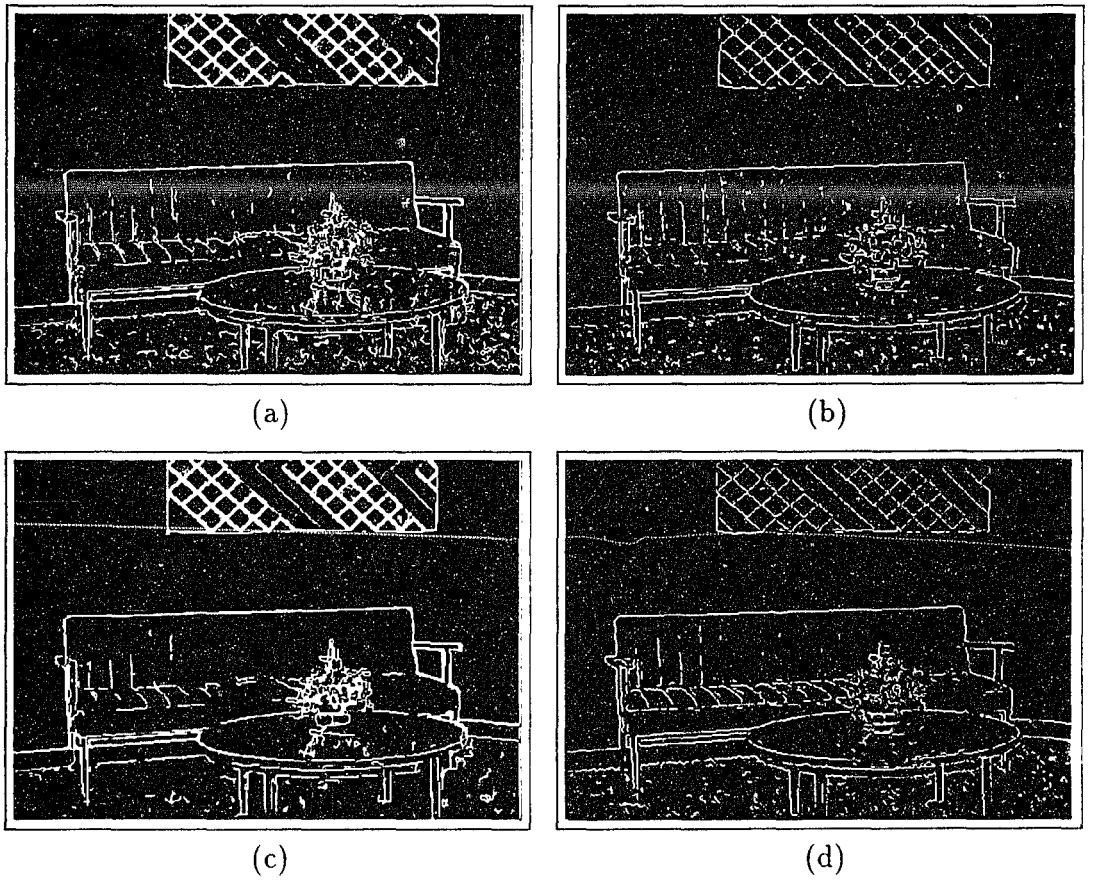
The Synthetic image is with additional Gaussian noise $\sigma = 100$. (a) is the results of Kittler and Hancock's algorithm; (b) is the results of Shen and Castan's algorithm; (c) is the results of our first algorithm, and (d) is the results of our new algorithm.

Figure 6.7: Magnified results for the synthetic image.



(a) a corner of an office; (b) a car; (c) a house; (d) an indoor scene.

Figure 6.8: Four natural images.

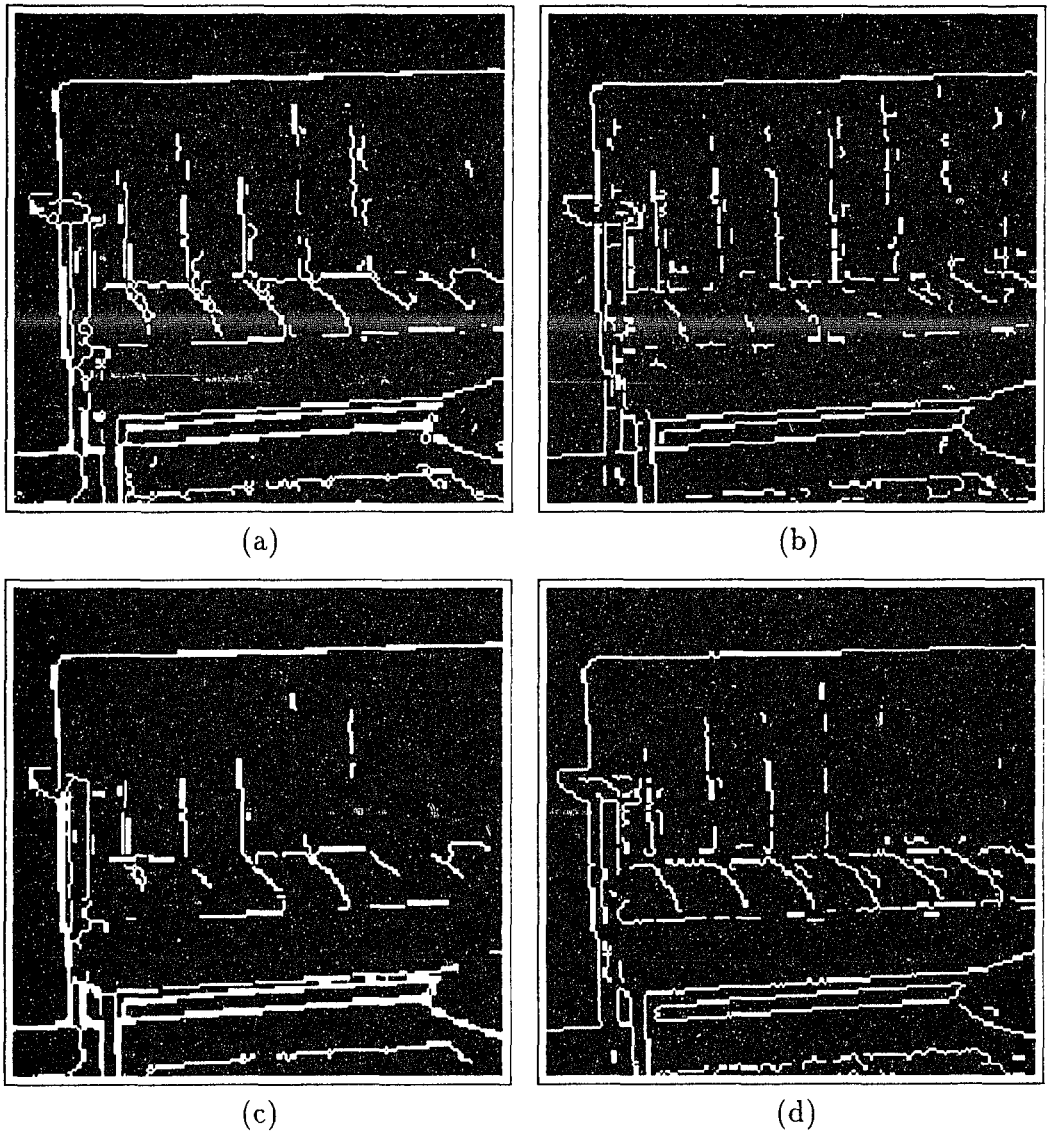


(a) (b)

(c) (d)

(a) is from Kittler and Hancock's algorithm; (b) is from Shen and Castan's algorithm; (c) is from our algorithm Algo1; and (d) is from our algorithm Algo2.

Figure 6.9: The results for the scene of an office.

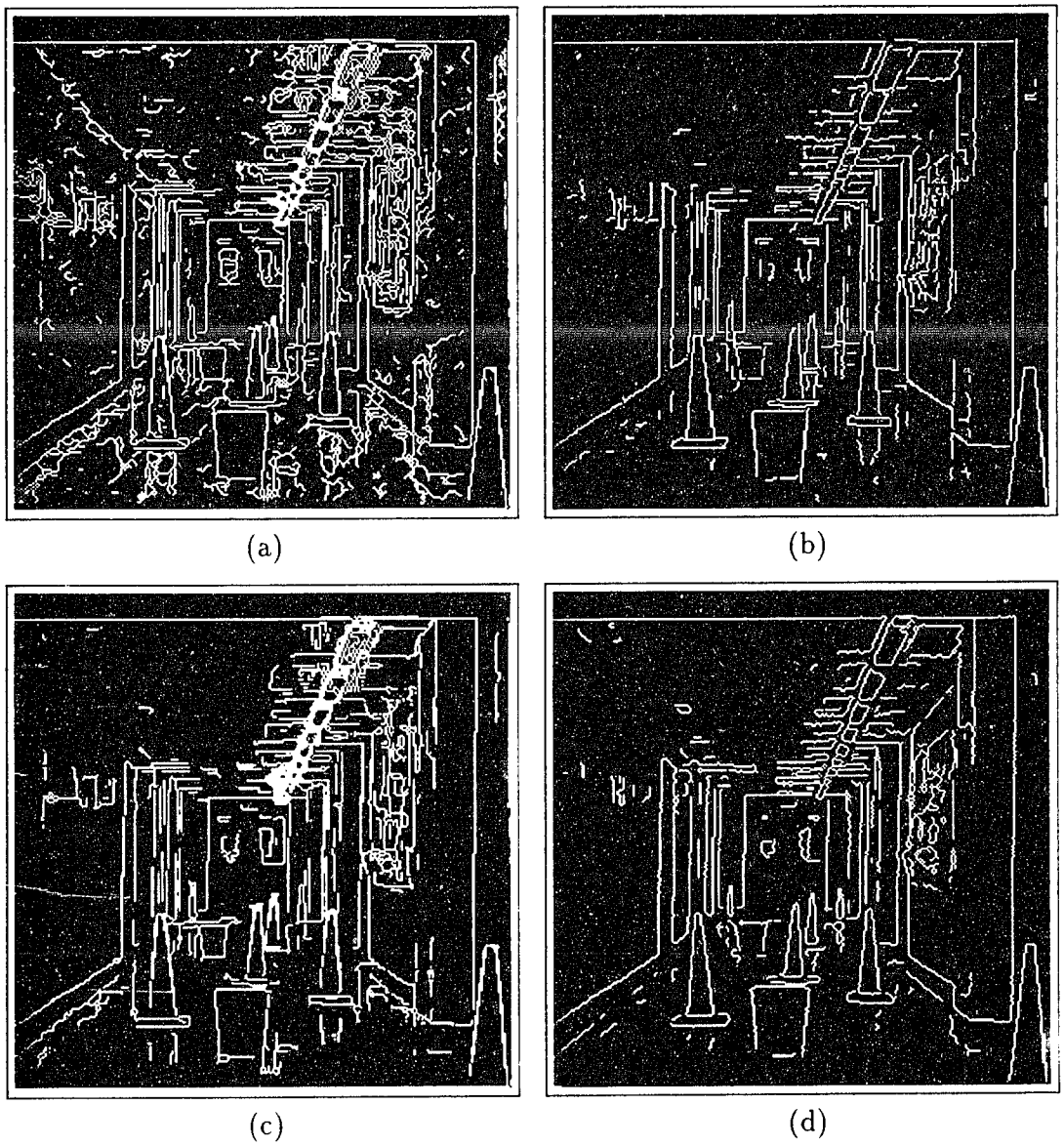


(a) (b)

(c) (d)

(a) is from Kittler and Hancock's algorithm; (b) is from Shen and Castan's algorithm; (c) is from our algorithm Algo1; and (d) is from our algorithm Algo2.

Figure 6.10: Weak contrast edges.

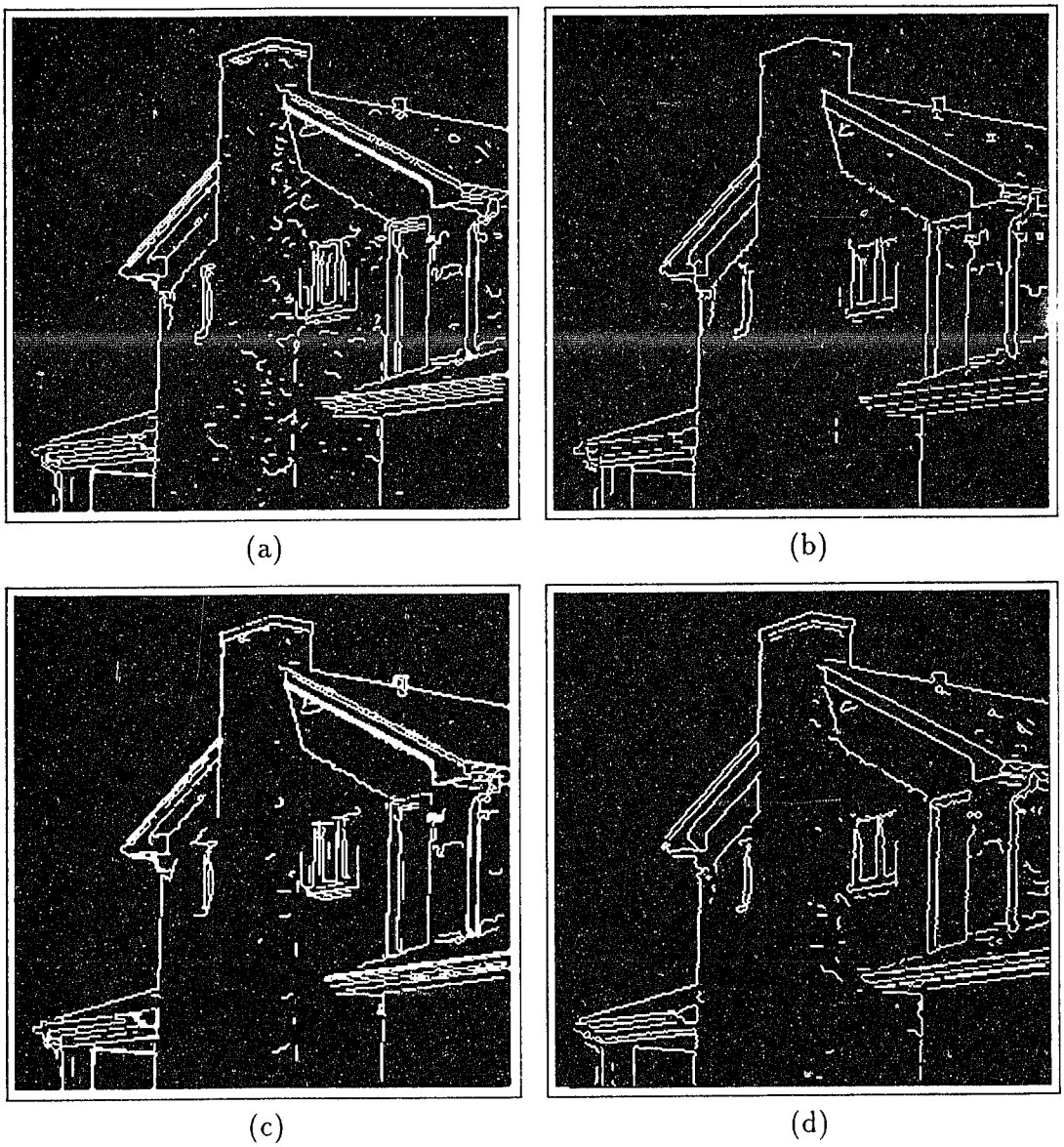


(a) (b)

(c) (d)

(a) is from Kittler and Hancock's algorithm; (b) is from Shen and Castan's algorithm; (c) is from our algorithm Algo1; and (d) is from our algorithm Algo2.

Figure 6.11: The results for an indoor scene.

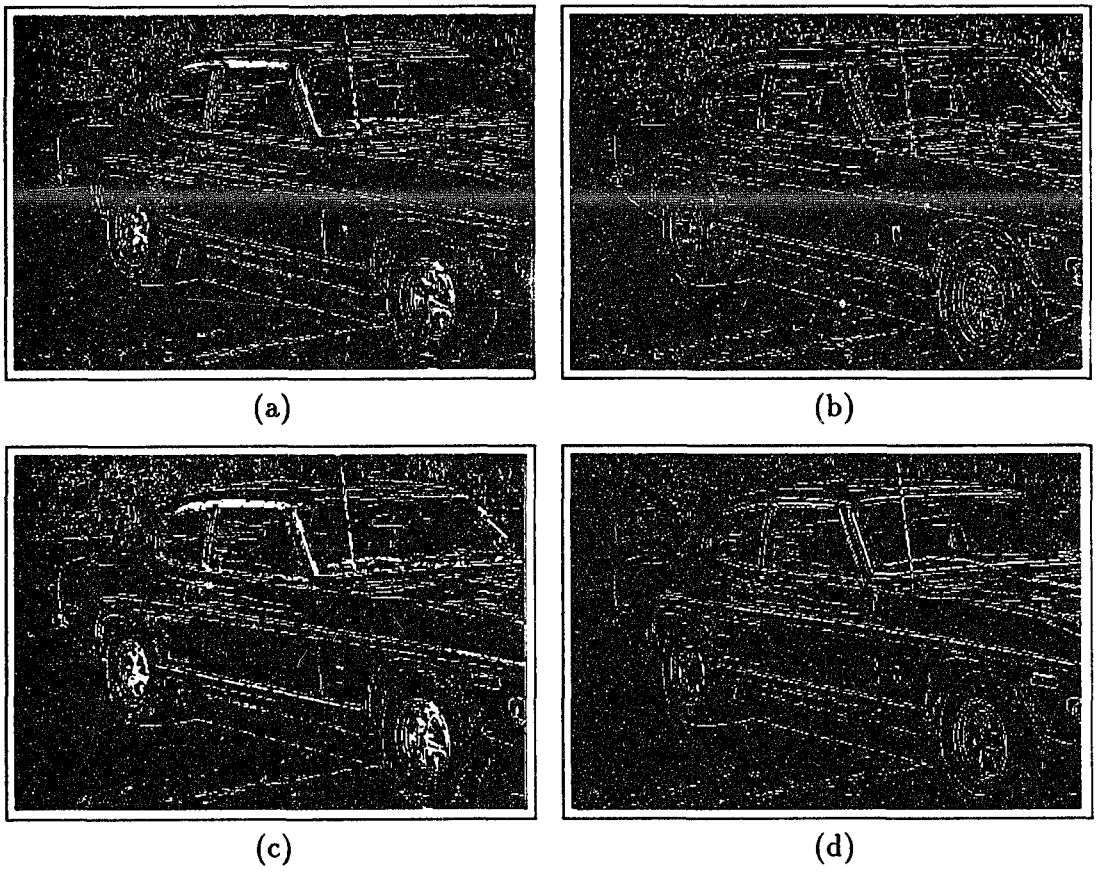


(a) (b)

(c) (d)

(a) is from Kittler and Hancock's algorithm; (b) is from Shen and Castan's algorithm; (c) is from our algorithm Algo1; and (d) is from our algorithm Algo2.

Figure 6.12: The results for the scene of a house.



(a) is from Kittler and Hancock's algorithm; (b) is from Shen and Castan's algorithm; (c) is from our algorithm Algo1; and (d) is from our algorithm Algo2.

Figure 6.13: The results for the scene of a car.

Chapter 7

Conclusions

This dissertation addressed the problem of parallelization for computer vision problems. Computer vision tasks process huge collection of pixel information. Traditional approach with von Neumann architecture cannot keep up with the computational demands. Parallel architectures backed by the increasingly sophisticated VLSI techniques have become more and more important. VLSI is best suited to computer architectures composed of a collection of simple processors connected together in a regular structure [59]. Mesh-locality which only requires local neighborhood communications and processors of a simple type is extremely useful.

In the foregoing chapters, we developed the theory and methodology to exploit mesh-local parallelism, mainly for computer vision problems. Using Markov Random Field theory as our theoretical basis and configuration dictionary method as our implementation tool, we have derived a new unified probabilistic relaxation scheme. This scheme conveys global information by neighborhood operations, and iteratively updates object labels. The mesh-local characteristics of the scheme makes it a good choose for parallel implementation.

We have used this technique for edge detection. Many other techniques including the recursive exponential filter, the zero-crossing of second order intensity difference, and the JND concept are also employed to make edge detection efficient and robust. The resulting algorithm obtains better edge connectedness and much improved noise resistance.

The problem of initial label assignment in our relaxation scheme has been discussed. Better initial estimation obtains better results.

This relaxation labeling scheme can also be used in many other lower- and intermediate-level computer vision tasks, such as noise suppression, image restoration, edge enhencement, image segmentation, pixel classification, and feature extraction. This new scheme can also be applied to a variety of other recognition problems in higher level computer vision and in the area of Artificial Intelligence.

We also presented a mesh-local parallel algorithm for binary image skeletonization. The implementation of the chessboard distance makes the algorithm faster, and makes the resulting skeletons preserve topological information better. This proposed algorithm has been successfully used in a hierarchical automatic route planning system. Parallel skeletonization techniques are also important for many other applications such as optical character recognition, binary image compression, fingerprint analysis, and biomedical applications.

For further research, the relation between relaxation technique and neural network is an important issue. [88, 23] have found that the general relaxation scheme and Hopfield networks are closely related. Since relaxation is a computational complex task, a study of dictionary based relaxation labeling methods and neural networks has the potential to find an efficient massive parallel implementation.

Other interesting questions or problems are:

- Compare and contrast relaxation labeling technique with simulated annealing. Both are iterative and have similar computing structure.
- For the transformation $\mathcal{T}$ derived in Chapter 4, Its behavior after many iterations is difficult to predict and needs further study.

- Recoverable parallel skeletonization technique are of interest to obtain fast binary image compression with high compression rate.

Bibliography

[1] C. Arcelli, L. P. Cordella & S. Levialdi, From local maxima to connected skeletons, *IEEE Trans. on PAMI*, V. 3, No. 2, pp.134-143, 1981.

[2] C. Arcelli & G. S. Di BaJa, A width-independent fast thinning algorithm, *IEEE Trans. on PAMI*, V. 7, No. 4, pp.463-474, 1985.

[3] M. Benn, A flexible method for automatic reading of hand written numerals, *Philips Technical Review*, V. 3, PP.37-47, 1981.

[4] J. R. Benton & A. Brink, Hierarchical route planner, *U.S. Army Engineer Topographic Laboratories*, Fort Belvoir, Virginia, 1990.

[5] H.Blem & R.N.Naggel, Shape description using weighted symmetric axis features, *Pattern Recognition*, V. 10, No. 1. pp. 167-180, 1978.

[6] H. Blum, A transfomation for extracting new descriptions of shape, *in Models for the Perception of speech & visional form*, Wathen-Dunn, W. ed. MIT Press Cambridge Pr., 1967.

[7] J. Canny, Computational approach to edge detection, *IEEE Trans. on PAMI*, V. 8, No. 6, PP. 699–714, 1986.

[8] V. Cantoni, S. Leviald, PAPIA: Acase History, *in Parallel Computer Vision*, ed. by L. Uhr. Academic Pr., pp.3-13, 1987.

[9] C. Chen & W. Tsai, A new fast one-pass thinning algorithm and its parallel hardware implementation, *Pattern Recognition Letters*, V. 11, No. 3, pp.471-477, 1990.

[10] J. S. Chen & G. Medioni, Detection, localization, and estimation of edges, *IEEE Trans on PAMI*, V. 11, No. 2, pp.191-198, 1989.

[11] R. T. Chin, H. Wan, D. L. Stover & R. D. Iverson, A one-pass thinning algorithm and its parallel implementation, *Computer Vision, Graphics, and Image Processing*, V. 40, No. 1, pp. 30-40, 1987.

[12] P. Clifford, Markov Random Fields in Statistics, *Disorder in Physical Systems*, ed. by G.R.Grimmett & D.H.A.Welsh, pp19–32, 1990.

[13] F.S.Cohen & D.B.Cooper, Simple parallel hierarchical and relaxation algorithms for segmentating noncausal Markov random fields, *IEEE Trans. PAMI*, V. PAMI-9, pp195-219, Mar., 1987.

[14] R.Cristi & M.Shridhar, A prallel algorithm for image segmentation based on the Gibbs field model, in *Proc. ISCAS 85*, pp127-130, 1985.

[15] L. S. Davis, C.Y.Wang & H.C.Xie, An experiment in multispectral, multitemporal crop classification using relaxation techniques, *Comput. Vision, Graph. Image Process*, V. 23, pp227–235, 1983.

[16] W. Deng & S. S. Iyengar, A new probabilistic relaxation scheme and its application to edge detection, *Revised for publication on IEEE Trans on PAMI*, 1994.

[17] W. Deng, S. S. Iyengar, & N. E. Brener A fast parallel thinning algorithm, *Accepted by The 1st International Workshop on Parallel Processing*, Bangolore, India, December 1994.

[18] W. Deng, S. S. Iyengar, & N. E. Brener A fast parallel thinning algorithm for the binary image skeletonixation, *Submitted for publication on Pattern Recognition Letters*, 1994.

[19] R. Deriche, Optimal edge detection using recursive filtering, in *Proc. First International Conf. on Computer Vision*, London, June 812, 1987.

[20] H.Derin & C.S.Won, A parallel image segmentation algorithm using relaxation with varying neighborhoods and its mapping to array processors, *Comput. Vision Graph. Image Process.*, V40, pp54-78, 1987

[21] A. R. Dill & M. D. Levine, Multiple resohition skeletons, *IEEE PAMI*, V. 9, No. 4, pp.495-504, 1987.

[22] S. Geman & D. Geman, Stochastic relaxation, Gibbs distributions and Bayesian restoration of images, *IEEE Trans on PAMI*, V. PAMI–6, pp.721–741, 1984.

[23] C. T. Genis, Relaxation and neural learing: points of convergence and divergence, *J. Parallel Distributed Comput.*, V. 6, pp.217-244, 1989.

[24] C.F.Gerald & P.O.Wheatley, *Applied numerical analysis*, Addison-Wesley Pub., 1985.

[25] R. C. Gonzalez & R. E. Woods, *Digital Image Processing*, Addison-Wesley Pub. Co. 1992.

[26] J. Gu, W. Wang & T. C. Henderson, A Parallel architecture for discrete relaxation algorithm, *IEEE Trans. on PAMI*, V. PAMI–9, No.6, pp.816–831, 1987.

[27] E. R. Hancock & J. Kittler, Edge-labeling using dictionary-based relaxation, *IEEE Trans on PAMI*, V. PAMI–12, No. 2, pp.165–181, 1990.

[28] E. R. Hancock & J. Kittler, Discrete relaxation, *Pattern Recognition*, N. 23, No. 7, pp.711–733, 1990.

[29] E. R. Hancock & J. Kittler, A label error process for discrete relaxation, *IEEE Trans on PAMI*, pp.523–528, 1990.

[30] E. R. Hancock & J. Kittler, Adaptive estimation of Hysteresis thresholds, in *IEEE CVPR 91*, pp.196-201, 1991.

[31] E. R. Hancock, Resolving edge-line ambiguities using probabilistic relaxation, in *CVPR 93*, pp. 300-306, 1993.

[32] F. R. Hansen & H. Elliott, Image segmentation using simple Markov Random fields, *Computer Graphics & Image Processing*, V. 20, pp.101–132, 1982.

[33] R. M. Haralick & J. S. J. Lee, Context Dependent Edge Detection, in *Proc. 9th ICPR*, Rome, V. 1, pp. 203-207, 1989.

[34] R.M.Haralick, J.L.Mohammed & S.W.Zucker, Compatibilities and the fixed points of arithmetic relaxation processes, *Comput. Graph. Image Process.*, V13, pp242-256, 1980.

[35] R. E. Hart, N. J. Nilsson, & B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans, System Science & Cybernetics*, Vol. 42, pp. 100-107, 1968.

[36] T.C.Henderson, in *Discrete relaxation techniques*, The International series of monographs on monographs on computer science, Oxford Univ. Pr., New York, 1990.

[37] W. E. Higgins & C. M. Hsu, Edge detection using 2-D local structural information, in *ICAPPS 91*, pp2549-2552, 1991.

[38] C. M. Holt, A. Stewart, M. Clint & R. H. Perrott, An improved parallel thinning algorithm, *Comm. ACM*, V. 30, No. 2, pp. 156-160, 1987.

[39] B. K. P. Horn, *robot vision*, The MIT Press, 1986.

[40] R. A. Hummel & S. W. Zucker, On the foundations of relaxation labeling processes, *IEEE Trans. on PAMI*, V. PAMI–5, pp267–287, 1983.

[41] S. S. Iyengar, W. Deng, & A. Nanavati, Dynamization of event based production system, *Proc. of KBCS-90*, Pune, India, Dec. 1990.

[42] S. S. Iyengar, & W. Deng, An efficient edge detection algorithm using relaxation technique, *Accepted for publication on Pattern Recognition*, 1994.

[43] J. G. Kemeny, J. L. Snell & A. W. Knapp, *Denumerable Markov Chains*, Springer-Verlag, 1976.

[44] J. Kittler & J. Foglein, On compatibility and support functions in probabilistic relaxation, *Comput. Vision, Graph., Image Processing*, 1986.

[45] J. Kittler & E. R. Hancock, Combining evidence in probabilistic relaxation, *Int. J. Pattern Recognition Artif. Intell.*, V. 3, pp29–52, 1989.

[46] J. Kittler & J. Illingworth, Relaxation labeling algorithms — a review, *Image and Vision Comput.*, V. 3, No. 4, pp206–216, 1985.

[47] J. Kittler, J. Illingworth, J. Foglein, & K. Paler, An automatic thresholding algorithm and its performance, in *Proc. 7th ICPR*, Montreal, 1984.

[48] T. Kreitzberg, T. Barragy, & N. Bryant, Tactical movement analyzer: a battlefield mobility toll, in *4th Joint Data Fusion Symposium*, Laurel, Maryland, 1990.

[49] N. Krishnakumar, S. S. Iyengar, R. Holyer & M. Lybanon, An expert system for interpreting mesoscale features in oceanographic satellite images, *Int. J. of Pattern Recognition and Artificial Intelligence*, V. 4 No. 3, pp341–355, 1990.

[50] D. Levine, *Vision in Man and Machine*, Ch. 1o, 1984.

[51] H. E. Lü & P. S. P. Wang, A comment on "A fast parallel algorithm for thinning digital patterns *Comm. ACM*, V. 29, No. 3, pp. 239-242, 1986.

[52] J. A. Malin, Automatic routing module, *Systems Control Technology Inc.*, Palo Alto, CA, 1986.

[53] R. Mehrotra, K. R. Namuduri & N. Ranganathan, Gabor filter-based edge detection, *Pattern Recognition*, V. 25, No. 12, pp.1479-1493, 1992.

[54] G. Mendel, Optical character recognition using morphological attributes, *Dissertation, Dept of Computer Science*, Louisiana State University, 1993.

[55] B. Moayer & K. S. Fu, A tree system approach for fingerprint pattern recgnition, *IEEE Trans. Comput.*, V. 25, No.3, pp.262-275, 1976.

[56] N. Naccache & R. Shinghal, SPTA: a proposed algorithm for thinning binary patterns *IEEE Trans. on SMC*, V. 14, No. 3, pp. 409-418, 1984.

[57] V. S. Nalwa & T. O. Binford, On Detecting Edges, *IEEE Trans. on PAMI*, V. 8, No. 6, pp.699-714, 1986.

[58] M. Nitzberg & T. Shiota, Nonlinear image filtering with edge and corner enhancement, *IEEE Trans on PAMI*, V 14, No. 8, pp.826-833, 1992.

[59] R. J. Offen, *VLSI Image Processing*, McGraw-Hill Book Co. 1985.

[60] T.Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, Berlin, German, 1977.

[61] L. Pelkowitz, A continuous relaxation labeling algorithm for Markov Random fields, *IEEE Trans. on SMC*, V. SMC–20, No. 3, pp.709–715, 1990.

[62] P. Perona & J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Trans. on PAMI*, V.12, No. 7, pp.629-639, 1990.

[63] M. J. Quinn, *Designing efficient algorithms for parallel computers*, McGraw-Hill Pub., 1988.

[64] L. G. Roberts, Machine perception of three-dimensional solids, *Optical and Electro-optical rmation Processing*, MIT Pr., Cambridge, Massachusetts, 1965.

[65] A. Rosenfeld, R. A. Hummel & S. W. Zucker, Scene labeling by relaxation operations, *IEEE Trans. on SMC*, V. SMC–6, pp.420–433, 1976.

[66] A. Samal & T. Henderson, Parallel consistent labeling algorithms, *International J. of Parallel Programming*,

[67] H. Samet, The Quadtree and Related Hierarchical Data Structures, *ACM Computing Survey*, Vol. 18, No. 2, pp.187-260, 1984.

[68] B. J. Schachter, A. Lev, S. W. Zucker & A. Rosenfeld, An application of relaxation methods to edge reinforcement *IEEE Trans. on SMC*, V. 7, pp.813–816, 1977.

[69] H. F. Schanty, *The history of OCR Optical character Recognition*, Tecognition Technologies Users Assoc. 1982. V. 16, No. 5, pp.341–364, 1987.

[70] J. Shen & S. Castin, An optimal linear operator for edge detection, in *Proc. SPIE' 87*, Miami, 1987.

[71] J. Shen & S. Castin, Further results on DRF method for edge detection, in in *9th I.C.P.R.*, Rome, 1988.

[72] R.V.Southwell, *Relaxation methods in engineering science*, Engineering Science Series, Oxford Univ. Pr., London, 1940.

[73] R.V.Southwell, *Relaxation methods in Physics*, Engineering Science Series, Oxford Univ. Pr., London, 1946.

[74] I. Sobel, Camera models and machine perception, *Stanford AI memo 121*, Dept. of Computer Science, Stanford University, 1970.

[75] L. A. Spacek, Edge detection and motion detection, *Image and Vision Computing*, V. 4, No. 1, PP. 43–56, Feb. 1986.

[76] R. Stefanelli & A. Rosenfeld, Some parallel thinning algorithm for digital pictures, *JACM*, V. 18, No. 2, pp. 255-264, 1971.

[77] Q. F. Stoat, Mesh and pyramid computers inspired by geometric algorithms, *Proc. workshop on Algorithm - Guided Parallel Arch. for Auto. Target Recognition*, pp.293-315, 1985.

[78] T. Sudkamp, C. Lizza, & C. Wagner, Hierarchic path generation, *SPIE*, Vol. 937, Appl. AI VI, pp. 442-449, 1988.

[79] S. Suzuki & K. Abe, Sequential thinning of binary pictures using distance transformation, in *Proc. 8th ICPR*, Paris, pp.289-292, 1986.

[80] V. Torre & T. A. Poggio, On edge detection, *IEEE Trans. on PAMI*, V. PAMI-8. No. 2, pp147-163, 1986.

[81] L. Uhr, *Parallel Computer Vision*, Academic Pr., 1987.

[82] D. O. Ungureanu, et al, A real time edge linker, in *CVPR 93*, pp793-794, 1993.

[83] D. Vernon & G. Sandini, *Parallel computer vision - the VIS $\vec{a}$ VIS system* , Ellis Horwood Ltd., 1992.

[84] D. Waltz, Understanding line drawings of scenes with shadows in P.H.Winston ed., *The psychology of computer vision*, McGraw-Hill, New York, pp19-91, 1975.

[85] Y. Wang & S. K. Mitra, Edge detection based on orientation distribution of gradient images, in *ICAPPS 91*, pp2569-2572, 1991.

[86] Rei-Yao Wu & Wen-Hsiang Tsai, A new one-pass parallel thining algorithm for binary images, *Pattern Recognition Letters*, V. 13, No. 10, pp.715-723, 1992.

[87] Y. Wu, S. S. Iyengar, & H. Min, Efficient edge extraction of images by directional tracing, *Tech. Report*, Dept of Computer Science, Louisiana State Univ., 1993.

[88] S.Yu & W.Tsai, Relaxation by the hopfield neural network, *Pattern Recognition*, V.25, No.2, pp. 197-209, 1992.

[89] Y. Xia Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes, *IEEE Trans. on PAMI*, V. 11, No. 10, pp.1076-1086, 1989.

[90] S.W.Zucker & J.L.Mohammed, Analysis of probabilistic labeling processes, *Proc. IEEE PRIP Conf.*, Chicago, IL, pp307-312, 1978.

[91] T. Zhang & C. Suen, A fast parallel algorithm for thinning digital patterns, *Comm. ACM*, V. 27, No. 3, 1984, pp.236-239.

Vita

Weian Deng was born on the 30th of Janurary, 1963, in Zhaoqing, Guangdong, China. He graduated with a B.E. in Computer Science and Engineering from Beijing University of Aeronatics and Astronautics, Beijing, China in 1984 and a M.E. in Computer Science from Zhongshan University, Guangzhou, China. Then, He was a lecturer in the Department of Computer Science of Zhongshan University from July, 1987 to August, 1990. He is a Ph.D. candidate in the department of Computer Science at Louisiana State University. Now, He is an assistant professor in the Computer Science Program at Teikyo Westmar University. He is interested in Image Processing, Computer Vision, Parallel Algorithm, Neural Networks, Natural Language Processing and Artificial Intelligence.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Deng, Weian

Major Field: Computer Science

Title of Dissertation: A Relaxation Scheme for Mesh Locality in Computer Vision

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

9/26/94