A KNOWLEDGE-BASED MACHINE VISION SYSTEM FOR AUTOMATED INDUSTRIAL WEB INSPECTION

by

Tai-Hoon Cho

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

APPROVED:

Richard W. Conners, Chairman

Morton Nadler

John W. Roach

Fred M. Lamb

Dong S. Ha

May 1991

Blacksburg, Virginia

A KNOWLEDGE-BASED MACHINE VISION SYSTEM FOR AUTOMATED INDUSTRIAL WEB INSPECTION

by

Tai-Hoon Cho

Committee Chairman : Richard W. Conners
Electrical Engineering

(ABSTRACT)

Most current machine vision systems for industrial inspection were developed with one specific task in mind. Due to the requirement for real-time operation, these systems are typically implemented in special purpose hardware that performs very specific operations. Hence, these systems inflexible in the sense that they cannot easily be adapted to other applications. However, current trends in computer technology suggests that low-cost general-purpose computers will be available in the very near future that are fast enough to meet the speed requirements of many industrial inspection problems. If this low-cost computing power is to be effectively utilized on industrial inspection problems, more general-purpose vision systems must be developed, vision systems that can be easily adapted to a variety of applications. Unfortunately, little research has gone into creating such general-purpose industrial inspection systems.

In this dissertation, a general vision system framework has been developed that can be easily adapted to a variety of *industrial web inspection* problems. The objective of this system is to automatically locate and identify "defects" on the surface of the material being inspected. This framework is designed to be robust, to be flexible, and to be as computationally simple as possible. To assure robustness this framework

employs a combined strategy of top-down and bottom-up control, hierarchical defect models, and uncertain reasoning methods. To make this framework flexible, a modular Blackboard framework is employed. To minimize computational complexity the system incorporates a simple multi-thresholding segmentation scheme, a fuzzy logic focus of attention mechanism for scene analysis operations, and a partitioning of knowledge that allows concurrent parallel processing during recognition.

Based on the proposed vision system framework, a computer vision system for automated lumber grading has been developed. The purpose of this vision system is to locate and identify grading defects on rough hardwood lumber in a species independent manner. This problem seems to represent one of the more difficult and complex web inspection problems. The system has been tested on approximately 100 boards from several different species.

Three different methods for performing label verification were tested and compared. These are a rule-based approach, a *k*-nearest neighbor approach, and a neural network approach. The results of these tests together with other considerations suggest that the neural network approach is the better choice and hence is the one selected for use in the vision system framework.

Also, a new back-propagation learning algorithm using a steep activation function was developed that is much faster and more stable than the standard back-propagation learning algorithm. This algorithm was designed to speed the learning process involved in training a neural network to do label verification. However this algorithm seems to have general applicability.

ACKNOWLEDGEMENTS

I wish to express my sincerest thanks to my advisor, Professor Richard W. Conners, for his guidance, patience, and support throughout the course of my Ph.D. studies. I also wish to thank Professor Morton Nadler, Professor John W. Roach, Professor Fred M. Lamb, and Professor Dong S. Ha for their suggestions and for the time they spent as members of my committee.

The financial support provided by the U.S. Forest Service is greatly appreciated. I am very grateful to Philip A. Araman for his help and support during the period of the project.

I would like to thank Mrs. Conners for her careful proof-reading of this thesis. I also wish to thank all members of the Spatial Data Analysis Laboratory for their friendship and support.

Special thanks to my mother and my deceased father for their love and encouragement. Also many thanks to my mother-in-law and father-in-law for their encouragement and support. I wish to thank my children, Jung-Yoon, Yoon-Suk, and Joon-Suk, for their love and understanding. Finally, I owe a tremendous debt of appreciation to my wife Sun-Duk. Without her devotion, support, and patience throughout the period of study, this work would not have been possible.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Motivation

The new world economy is going to mean increased competition among companies in various areas, especially in manufacturing industries. This competition among companies is going to put more pressure on company management to find more cost effective methods for increasing productivity and improving quality control.

Machine vision systems provide a mechanism to accomplish both of these objectives. Such systems can be used to inspect a product to assure that it meets quality standards and also to verify that there are no problems occurring along the production line. While humans perform well on most visual inspection problems, their performance is susceptible to a number of factors. For example, human inspectors can easily become bored or tired when performing long and routine tasks such as the visual inspection of a product. Therefore, the automation of this visual inspection task should improve quality control.

Machine vision systems can also be used to increase productivity. Such systems can be the "eyes" for an automated manufacturing equipment. The creation of automated manufacturing equipment that has sensory input has the potential to markedly reduce production costs.

The idea of using machine vision systems for automated industrial inspection is not new. An extensive survey of these systems is given in [CHI88]. Machine vision

technology has been successfully applied to a number of industrial inspection problems. To be industrially useful, an industrial inspection system must be able to meet the following requirements [CHI86].

1. Speed. The operation speed of the vision system must match that of the production line.

2. Accuracy. The accuracy of the vision system in defect detection must be high enough to be acceptable.

3. Flexibility. The vision system must be flexible enough to accommodate changes in products and competent enough to inspect parts in an uncontrolled environment.

Unfortunately, most current vision systems for automated industrial inspection have a number of marked limitations [CHI86]. First, most existing industrial inspection systems are binary image processing systems. They require well-positioned, isolated parts against a high-contrast, clean background. The use of a binary representation reduces the amount of data that must be handled. While simplification markedly reduces the cost of industrial inspection systems, it places severe limitations on the capabilities of these vision systems. For example, physical surface properties (e.g., textures) are lost in reducing a gray-scale image to a binary image. There exist only a few gray-scale vision systems and algorithms that are capable of extracting useful features from complex industrial scenes with considerable noise caused by dirt and unfavorable lighting conditions [CHI86]. Such systems are needed to inspect both simple surfaces as well as complex parts.

Most current vision systems for automated industrial inspection are very inflexible. They cannot be easily adapted to work on a problem other than the very specific application for which they were designed. As Chin [CHI86] states

> "Most current vision systems are custom-engineered systems with only one specific task in mind. They represent *ad hoc* approaches to a single problem. As more complex industrial parts must be handled, the vision system has to be flexible and more versatile. Until recently, we have not seen a strong desire for general-purpose machine vision systems. The capital costs of general systems are relatively high because they are difficult to design, but they can be justified in the long run by their utilization rate."

Many of the limitations of the current machine visions systems have resulted from the need for real-time processing. The lack of high-speed inexpensive computer systems is one cause of the problems. A second cause is the need for robustness. Researchers have not been able to create truly robust general purpose computer vision methodologies [BAR81b]. There have been only a few attempts to create fairly general purpose vision systems, e.g., [NAG80, HAN88]. Robustness of these systems is very questionable because of the small number of images to which each of these systems has been successfully applied.

The first problem of industrial machine vision systems should be mitigated by projected improvements in computational capabilities of microprocessors during the next decade. This development should motivate the rapid growth in the industrial machine vision area. However, if there is to be rapid growth, one cannot start from scratch in developing each new machine vision system. One needs to have at least a general framework from which to begin. This leads to the second problem that remains a major point of concern. Developing truly robust very general vision systems does not appear achievable in the near future. This dissertation primarily attempts to the address second problem, i.e., the robustness issue, in industrial web inspection, which is a rather general and very important subclass of industrial machine vision problems.

Industrial web inspection problems involve inspection of *planer* (flat) material or patterns. These typically have the following characteristics:

- Surface of the material is relatively flat.
- Material flow is continuous.
- Flow rate is relatively high to high (typically, 2 - 20 linear feet per second).
- Relatively high spatial resolution is required (up to 100 pixels per inch).
- Defects make up small percentage of total surface area.
- The same defect type manifests itself in many different ways.

This industrial web inspection is important because most industrial inspection problems except inspection of volumetric shaped fall into this category materials [CHI86]. Inspection of these volumetric objects requires 3-dimensional image analysis methods. Problems of inspecting printed circuit boards (PCBs), microcircuit photomasks, integrated circuits (ICs), paper, textiles, surfaces of lumber, etc. are included in the web inspection area.

1.2 Objectives

In this dissertation, a flexible computer vision system framework is described that attempts to address a fairly general class of industrial inspection problems, the web inspection problem. The purpose of the vision system is to locate and identify "defects" on the surface of the material being inspected. The definition of defects depends on a specific application problem. The input of the vision system is assumed to be a color image of the material captured through a color scanning device, e.g., a color line scan camera. The output of the vision system includes an accurate boundary for each defect present and an assigned label that identifies the type of defect present in each of the outlined areas.

The design criteria used to create this vision system framework are as follows.

1. The vision system must be robust. It must be able to accommodate without adjustment small changes in lighting and/or changes in the surface characteristics of the material caused by acceptable variations in processing.

2. The vision system must be flexible in order to enable the system to be easily adapted to a number of web inspection problems.

3. The vision system must be able to process high spatial resolution image data. In general, high resolution imagery is required to detect small defects, although exact resolution required depends on the application being considered.

4. The operation of the vision system must be fast enough to at least accommodate current production speeds.

Since robustness equates to accuracy, these design requirements represent only a restatement of Chin's three criteria.

To achieve the above performance requirements, a number of "sophisticated" vision methodologies were incorporated into the proposed vision system framework. These methodologies are briefly described below. A detailed discussion will be presented in Chapter 2.

To assure robustness, several concepts are incorporated into the system framework design. 1) A combination of bottom-up (data-driven) control and top-down (model-driven) control is used to perform scene analysis. This combined or mixed control strategy is necessary to make the system robust and flexible. Advantages of the mixed control strategies will be discussed in detail in a later chapter. 2) Hierarchical object models that consider defects at various levels of representation are used to recognize the defects. 3) A method for uncertain reasoning is incorporated into the vision system framework because there is uncertainty in all stages of visual

processing. Local errors are introduced by the imaging and digitization process, by imperfect KSs (e.g., segmentation) as the result of incorrect heuristics, or by inherent ambiguity in the scene itself.

To make the vision system flexible, the knowledge structure used is modular. It is important to keep a priori problem-dependent knowledge only for a particular web inspection problem separate from general knowledge for a variety of web inspection problems in order to make the system easily adaptable to different web inspection problems. It is also important to organize the problem-dependent knowledge in a modular manner so that it can be easily adapted to a change of operating environment, so that adding new knowledge or deleting obsolete knowledge can be easily accomplished. As will be described in detail later, the Blackboard framework provides this modularity of knowledge sources.

The third and fourth requirements force the vision system to handle and analyze a large amount of image data, at least at the speed of current human inspection. These requirements are addressed by attempting to create a framework that minimizes the computational complexity of the analysis task. To achieve this goal, several methods were used. 1) For segmenting images of material to be inspected, a histogram-based multi-thresholding is used because of its computational simplicity. 2) Other analysis algorithms are also optimized to minimize computational complexity. One way this optimization is accomplished is by using a focus of attention mechanism to guide the scene analysis process. This focus of attention mechanism marks candidate regions so that they will be operated on by the most appropriate defect detection procedures. This marking is accomplished using computationally simple methods. 3) Knowledge about defects is partitioned according to defect type. This allows concurrent parallel processing by running each defect detection procedure on

a separate processor, thus speeding up the system operation.

On the basis of the proposed vision system framework, a computer vision system for automated lumber grading has been developed. The purpose of this vision system is to locate and identify grading defects on rough hardwood lumber in a species independent manner. This problem seems to represent one of the more difficult and more complex web inspection problems. No vision system has been developed to tackle this problem. The main reason for this seems to be that there are a lot of factors that cause surface discolorations of normal surface. These discolorations must not be mistaken for a defect. Further, some defect types are difficult to recognize due to irregular characteristics of their shapes. Hence if one can use the previously described framework to develop a vision system for grading rough lumber, this framework should be versatile enough to work on a variety of other web inspection problems. In fact, the methodologies used in the vision system for grading would seem applicable to a number of other web inspection problems as well.

As a result of this research, a number of seemingly relative general purpose computer vision methodologies were developed. These include: 1) a focus of attention mechanism using fuzzy logic, 2) a multi-thresholding method, and 3) region merging operation using Dempster's theory of evidence. Also, a new back-propagation (BP) learning algorithm is introduced that is much faster and more stable than the standard BP learning algorithm. This new algorithm uses automatic initializations of weights in the network when convergence to a solution is very slow.

Three different methods for performing label verification were tested and compared. These are a rule-based approach, a *k*-nearest neighbor approach, and a neural network

approach. The results of these tests together with other considerations suggest that the neural network approach is the better choice and hence was selected for use in the vision system framework.

1.3 Organization

This dissertation consists of seven chapters. In Chapter 2, conceptual arguments are presented that discuss which methodologies should be employed for developing a computer vision system for industrial web inspection that can meet the design requirements discussed earlier. It is argued that a knowledge-based approach, in particular, a production system type structure using a mixed control strategy is appropriate for such a computer vision system. Chapter 3 gives a literature review of knowledge-based computer vision systems in various application areas. In this review, concentration is focused on how these systems compare with the proposed vision system in various aspects. In Chapter 4, justification for adopting the Blackboard framework among various production systems is presented with a detailed description of an adapted Blackboard framework tailored to the web inspection problem. The low-level segmentation module and the high-level recognition module of the proposed vision system framework are discussed in detail in Chapters 5 and 6, respectively. To demonstrate the robustness of this framework it is applied to a hardwood lumber inspection problem. This problem is described in Chapter 7. Finally, Chapter 8 gives conclusions of this dissertation.

Chapter 2

VISION SYSTEM CONCEPTUALIZATION

Usually, a computer vision system, especially a knowledge-based vision system, consists of two stages: *low-level vision* and *high-level vision.* The low-level vision tries to extract features characterizing input image data through image processing operations such as edge detection and region segmentation. This involves transforming a numerical matrix representation of an image to a symbolic representation that is composed of a set of spatially related image primitives, such as edges and regions. Also various features are extracted from the primitives. High-level vision seeks to attach a consistent interpretation to these primitives and construct a description of the scene. This scene analysis is carried out using a priori knowledge about the scene's domain. Interpretation of real world images is difficult because of uncertainties arising when formulating hypotheses based on noisy image data and imprecise models about what objects might appear in the scene. Furthermore, different hypotheses can even conflict. Consequently, a knowledge-based approach is commonly adopted for high-level vision.

The computer vision system for industrial web inspection can be conceptualized into two components : a *low-level segmentation module* and a *high-level recognition module.* In what follows, how these components should function and which methodologies are appropriate for the web inspection application are described.

2.1 The Low-Level Segmentation Module

The purpose of the segmentation module in industrial web inspection is to reduce the huge amount of image data of the material to be inspected by extracting regions that might potentially contain a defect. An objective is to do this as fast as possible. Since in web inspection problems defect area is very small compared to the area of normal surface, computational complexity can be markedly reduced if defect regions can be extracted using simple methods at an early stage of the processing. Once this has been done, higher level processing that is typically more sophisticated and time-consuming can be applied only to those regions believed to potentially contain a defect. Certainly, no segmentation method can produce perfect boundaries of defects. However, it is only important for the segmentation process to detect a key part of each defect, even if the defect's accurate area and boundary cannot be detected. These key parts of defects can then be used as clues for invoking top-down processing methods in the high-level module to find a more accurate boundary for each such defect.

There exist two major approaches to image segmentation: edge-based and region-based [NEV86]. For a survey of segmentation, refer to [HAR85] and [NEV86]. In edge-based methods, edges or local discontinuities in gray level are detected first using an operator and then connected to form longer, hopefully complete, boundaries. There are two problems with edge-based segmentation methods [BAL82]. First, edge detection methods often produce a number of "false" edges due to image "noise" where no meaningful scene boundary exist. These false edges increase the computational complexity of later processing stages. This is very undesirable. The other problem associated with the edge-based segmentation is that there are actually locations on an object boundary that cannot be detected using only local information,

but require the use of global contextual information. These locations cause gaps between detected edges. To find a complete boundary, these gaps must be filled by edge following or linking of edges. Filling these gaps further increases computational complexity. These problems may be somewhat alleviated by using recent advanced edge detection methods such as [MAR80, CAN86]. However, it is unwise to indiscriminately apply edge detection methods to every pixel of an image. In industrial inspection, images can be very large and applying an edge detector to every pixel can be time-consuming. Thus, edge detection methods should be selectively applied only to areas where edge information is needed for image analysis.

In the region-based segmentation, a connected set of pixels is found that share a common property (such as intensity or color) [NEV86]. Methods used for the region-based segmentation can be divided into two categories: 1) methods that are directly applied to *images* (e.g., region growing [BRI70] and split/merge methods [PAV77]) and 2) methods that are applied to a *feature space* of images where features are extracted from images (e.g., thresholding [SAH88]). Note that methods in the first category have a computational complexity that is proportional to the number of pixels in the input image. The computational complexity of methods in the second category depends on the number of features used and the complexity associated with feature extraction. Gray-level histogram-based thresholding is the most efficient of methods in the second category because it works on a 1-dimensional feature space (the gray-level histogram) and the features (gray-levels) are obtained directly from the original image. Also, this method is much more efficient than any method in the first category. This speed difference is particularly significant for high resolution images such as those typically encountered in web inspection problems. For example, in lumber inspection problem, boards can be up to 13.5 inch wide and 16 feet long, and 64 pixels per inch resolution is typically required.

A region growing method would have to be applied to approximately 10 Mbytes of image data while a histogram-based thresholding method needs to be applied only to a 256 element gray-level histogram.

The above edge/region based segmentation methods are based on the assumptions that objects consist of relatively homogeneous surfaces or that intensity changes correspond to object boundaries. In the texture images, these assumptions are no longer valid. To segment texture images, a texture operator is typically required. The size of the texture operator should be larger than that of the fundamental elements that comprise the textures. A number of texture segmentation methods have been reported in the literature. A survey of these methods is given in [HAR79, HAR86]. Texture operators are too computationally complex to be used to segment images for industrial web inspection. These operators also suffer from the fact that accurate boundaries of defects cannot be obtained because texture measures are usually calculated from large regions, not just a few pixels. Of course, some problems, e.g., inspection of sand paper, require use of texture segmentation methods. However, in general, it is advisable to apply texture operators selectively only to areas where texture measures are absolutely needed.

The above discussion of the various segmentation methods leads to the conclusion that a *gray-level histogram based thresholding method* seems to be the most appropriate segmentation method for use in computer vision systems for industrial web inspection. If texture features and edge features are extracted, they should be extracted at a later processing stage and should be computed from only those areas where these features seem to be necessary for more detailed analysis.

It is generally agreed that early vision methods used to segment images should be domain-independent. However, the lack of truly robust early vision methods means

that in actuality, one might have to incorporate problem-dependent a priori knowledge into the design of early vision operators in order to get good, "very accurate" segmentation results.

2.2 The High-Level Recognition Module

The purpose of the recognition module is to perform the scene analysis task. It analyzes each of the regions forwarded to it by the segmentation module and identifies the type of defect present in each of these regions. It will be argued that the vision system requirements force one to incorporate artificial intelligence methods into the design of the recognition module. Similar arguments to those that will be presented here have been given before by Nagao and Matsuyama [NAG80] but in the context of a different problem domain.

2.2.1 Scene Analysis Strategies

Conceptually, there are three basic approaches to scene analysis [CON87]. The first of these is the *bottom-up* type approach. Using a version of this type of approach image data is processed by a number of different operations, each operation producing a new data structure that makes some new facet of the image explicit. The last of the operations performed are those that actually label the region of an image.

Bottom-up approaches have their origin in very early computer vision research [ROB65, HUF71, CLO71]. Bottom-up approaches are known to be very sensitive to noise. Any mistake made by an early processing operation propagates up through the rest of the processing operations. As such, this type of approach has proven ineffective on real world images [CON87], e.g., images of rough lumber.

A second class of scene analysis strategies is *top-down* methods. The basic idea behind a top-down method is the formulation of a hypothesis about what is in the image. Once the hypothesis has been made operators are applied to the image to verify whether the formulated hypothesis is correct. Note that the initial hypothesis is generated without using any information collected from the image. Further, if the results of an operation disprove the current working conjecture of the scene analysis system, then another working conjecture or hypothesis is generated. The generation of this new hypothesis is also independent of any information obtained from the image.

Because no image derived information is used in formulating working hypotheses top-down methods are very limited in their generality. However, these approaches have been successfully used on very complicated albeit highly structured real world scenes, e.g., the analysis of chest radiographs [TSI74].

The third class of scene analysis strategies is the *combined, mixed* or *heterarchical* strategies. Such strategies use a combination of both bottom-up and top-down methods. Image derived information is used to guide the analysis and to formulate hypotheses -- bottom-up processing. Special operators are then applied to the image to verify the hypotheses -- top-down processing. It is generally agreed that a combined strategy is the most robust type of control strategy. It is also generally agreed that human vision uses a combined strategy. The high-level module of the proposed computer vision system proposed uses a combined strategy.

2.2.2 Production Systems

It will be argued that a knowledge-based architecture, particularly a *production system* is the most appropriate choice for the vision system, especially for the recognition module.

In creating this or any other industrial inspection system there is always a trade-off between increasing system identification accuracy and the need for real-time processing. The best design is one that gives acceptable accuracies in the shortest possible time. To accomplish this objective requires that the software be easily modifiable, allowing easy incorporation of additional algorithms for improving accuracy or the removal of algorithms that are unnecessary in order to meet established accuracy criteria. One way, perhaps the only way, that this can be accomplished is to have the program be made up of a number of loosely coupled modules that communicate in an explicit manner.

On real world applications the need for robust vision systems typically requires complex scene analysis strategies. That is, neither a strictly bottom-up nor a strictly top-down strategy will suffice. Instead, a combined strategy is required. An important part of such combined strategies involves making hypotheses about what is present at a particular location in an image. It also involves testing these hypotheses by performing operations on the image and examining the results. Finally it involves keeping track of all competing hypotheses, ordering the attempts to verify each based on which hypothesis is believed to be the most probable. This implies the need for a belief maintenance system.

These arguments all point to the fact that a *pattern directed inference system* should be an important part of vision system design. In particular, a *production system*

seems an appropriate choice for the system. A design based, in part, on a production type system would certainly provide a mechanism for providing the loosest possible coupling among program modules. The "if" part of each rule would specify the conditions under which a particular module should be executed. The execution of the module would be the "then" part of the rule.

Incrementally adding new program knowledge or removing unneeded program knowledge involves either adding or removing rules from the system. This is facilitated by the fact that the conditions for executing a module are explicitly given in the rule. Adding or removing rules does not require that other rules of the system be changed. It merely alters how the system responds to given sets of circumstances.

Ordering attempts to validate competing hypotheses is also easily accomplished. Each hypothesis is an assumption that a particular defect is present at a particular location in the image. Associated with each hypothesis are rules for validating it and "conclusively proving" that that defect is the one present. Structuring the knowledge base by grouping the validation rules by hypothesis represents a natural method for improving overall efficiency. This restricts the number of rules that must be considered in performing the match to see which rules might be applicable for firing at any given instant in time. Associated with each hypothesis is a belief. The hypothesis that is currently considered to be the most likely becomes the working hypothesis, the one whose rules will be fired by the system. A change of hypothesis causes a change in the rules that are considered.

With regard to object models for the various defects, robust vision methodologies typically demand hierarchical models to be used, ones that consider objects at various levels of abstraction. This implies that the knowledge base takes on much more structure than that described above.

Practically speaking it is important to use as few levels of description as possible since doing so can markedly affect not only the computational complexity associated with belief maintenance [BAR81a] but also the complexity of the structure that must be imposed on the knowledge base.

These arguments suggest that a production system represents a very good programming paradigm to use to address a number of the tasks this industrial inspection system must perform. Note that the tasks for which this paradigm seems most appropriate are all relatively high level tasks.

2.3 Summary

In this Chapter it is argued that a robust vision system for industrial web inspection system should have a number of characteristics. These characteristics are as follows:

1. Segmentation should be based on some type of gray-level histogramming operations.

2. The segmentation may have to incorporate certain characteristics specific to the web inspection problem in order to obtain required segmentation accuracies.

3. The control strategy used in the high-level recognition module should be a combined strategy employing both bottom-up and top-down components.

4. Object models used to recognize defects should be hierarchical in nature.

5. A production system type paradigm should be used in the software implementation of the vision system.

6. A belief maintenance system is required to help the vision system cope with the processing results obtained from noisy image data, matching against imprecise object models, and choosing the most appropriate hypothesis from a series of competing hypotheses.

Chapter 3

COMPUTER VISION SYSTEMS: A REVIEW

The purpose of this chapter is to describe a number of computer vision systems that have been developed. There are four objectives associated with this literature review. The first is to acquaint the reader with the current state-of-the-art in computer vision technology. The second is to demonstrate the very special purpose nature of existing industrial inspection systems. The third is to review a few of the "more general" systems that have been developed. These "more general" systems are all knowledge-based. The last objective is to compare some of the more sophisticated systems described here with the desirable attributes described in the last chapter.

As might be expected, as the role of a vision system becomes more general the robustness of the system typically suffers. A goal of this research is to create a rather general system that still has a good deal of robustness.

3.1 Industrial Inspection

Most vision systems that are currently in use in industry were developed with one specific inspection task in mind. Due to the requirement for real-time operation, each was typically implemented in special-purpose hardware, hardware designed for performing very specific operations. Hence, the resulting machine vision system is typically inflexible in the sense that it is not easily adapted to other applications. To show the inflexibility of typical industrial inspection systems, several examples will be presented. It should be pointed out that while these systems are inflexible,

they are robust enough to handle the day to day analysis tasks they were designed to perform. Hence, each of these systems has merit based on its established industrial utility.

The first industrial inspection system to be described is INSPECTOR [PER83]. INSPECTOR is a computer vision inspection system that learns the difference between good and bad parts by being shown several identified good and bad parts. The model, formed during the training session, contains a set of identifying points that are used for locating parts, a set of inspection regions, and a set of inspection tests. Using the model, the system can distinguish between good parts and bad parts with an arbitrary number of defects. This system employs a rapid and reliable search technique for obtaining a registration between a part and its part model. It can be used to inspect parts for missing components, wrong components, misplaced components, and damaged components. However, it uses a few very simple features, and cannot handle high-level descriptions or scale changes. Hence, the flexibility of this system remains in doubt. Perkins does not state whether this system is currently being used by industry.

Yoda et al. [YOD88] described an inspection system that can reliably detect submicron defects on multilayered LSI wafer patterns. This is a dedicated system for the inspection of repetitive areas such as memory cell arrays in memory chips. Pixel-to-pixel comparison of two adjacent memory cells is performed in a real-time mode to find portions of the cells that are different. The resultant portions are considered as the candidates for defects and analyzed by referring to digital design patterns generated automatically from the CAD lithography data. Each step of the image analysis in this system is executed by a pipeline-structured image processor

in real-time. There is no doubt about the robustness of this system. It is currently in use in industry. However, it clearly cannot be easily adapted to problems other than LSI wafer inspection.

Hara et al. [HAR88] developed a printed circuit board (PCB) pattern inspection system based on fluorescent light optics. In this system, two binary images are obtained using florescent light from the corresponding parts of two printed circuit patterns, f and g, created from the same mask. Any difference between the images of these two patterns should reveal defects. Instead of comparing these images directly, a set of boundary features are extracted from them. The comparison of the two patterns is based on these features. All this image processing is performed by pipelined real-time hard-wired logic circuits. Test results in the plant showed that this system performed better than employees and also at a much faster speed. It clearly performs its function in a very robust manner. However, as the above should suggest, this system is very limited as to the number of applications to which it could be applied.

Only a very few industrial inspection systems have been created that are knowledge-based. One reason for this might be that real-time processing is usually a critical requirement in industrial applications. Unfortunately, by their very nature, knowledge-based vision systems can be somewhat computationally complex, and hence achieving a real-time implementation of such systems has been difficult in the past. However, the rapid evolution of computers will assure that in the very near future much more powerful and faster computing machines will be available at a reasonable cost. This should allow a cost effective real-time implementation of such systems. Some notable knowledge-based vision systems for industrial inspection are reviewed below.

Darwish and Jain [DAR88] used a rule based approach for detecting defects in printed circuit boards. Their approach is composed of segmentation and rule-based verification of defects. The segmentation consists of binary thresholding, thinning and pruning into a line drawing, decomposition into shape primitives (e.g., line segment, circle), and labeling the original binary image using the obtained primitives as seeds. Then a semantic network constructed from the decomposed line drawing is matched to the model stored for the scene. Missing and extraneous segments can be found by this matching process. However, since the semantic network is based on the representation of line segments, patterns cannot be detected during this matching that are thinner (thicker) than the required minimum (maximum) width of patterns. Hence, testing of different regions is required to ensure that they meet the design requirements such as minimum (maximum) pattern width and minimum spacing between patterns. These requirements are formalized by a set of rules. Some of the advantages of this system are that it is fairly modular, so it is flexible in the sense that models and design rules can be adapted to different specifications, and that it uses two levels of representations for patterns, segmented regions and segmented line drawing. On the other hand, it still lacks some desirable features needed in robust vision systems. 1) Its control is done in a bottom-up manner. 2) No uncertainty management approach is employed. 3) No focus attention mechanism is used. Hence each segmented region must be tested to see if it meets the design rules. This is very inefficient. 4) Binary thresholding is used in the segmentation without using gray-level information. Thus, the segmentation method is not appropriate for applications that require characterization of texture properties. This system was designed at the University of California at Davis. No indication is given as to whether this system is used by industry. System testing involved 95 double-sided printed circuit boards. One hundred percent correct detection of

defects was reported. However, the images used in the study were created using "manual image acquisition" to guarantee good illumination conditions for each board. Hence, it is uncertain as to how well this system would perform in an industrial environment.

Sanz and Petkovic [SAN88] presented a prototype system for automated inspection of thin-film disk heads. This system can be divided into three steps performing image-to-image operations, image-to-symbol operations, and classification of symbolic objects. The first step consists of noise removal, shading correction, boundary fitting of the disk head, and histogram-based thresholding. The second step performs connected component labeling and feature extraction. The third step consists of defect classification, measurement, and comparison with inspection specifications. For defect classification, a rule-based system was used, one whose inputs are both geometrical and gray-level features extracted from previously segmented regions. These three steps were implemented, respectively, in general-purpose image analysis pipeline architectures, a special reconfigurable image processing system, and a conventional microprocessor-based computer, which hosts the pipeline architecture. One of the desirable characteristics of this system is that the choice of a rule-based system for defect classification and comparison to inspection specifications allows flexibility in modifying, adding, or deleting defect classes and specifications. Most importantly, this system was extensively tested using over 10,000 disk head images. The authors' report that this system was effective in its analysis of these images which contained over 850 verified defects. However, some disadvantages exist in this system. 1) Its control strategy is purely bottom-up. 2) Its rule-based classifier is not efficient because there is no focus of attention mechanism for determining which potential defect should be tested next. 3) No uncertainty management scheme

is used. 4) There is only one level of defect descriptions, i.e., the features extracted from each segmented region. These disadvantages markedly limit the flexibility of this system to be easily adapted to other inspection problems.

Barlett et al. [BAR88] explored two approaches to automatic solder joint inspection: statistical pattern recognition and expert systems. (It is assumed that manual registration of solder joint images is performed.) In the statistical pattern recognition approach, features are extracted from a subimage, i.e., that portion of the digitized PCB image containing a single solder joint. Using a dimensionality reduction technique, these features are decorrelated and automatically weighted according to their contribution to the decision making process. The expert system uses features more analogous to the visual clues that a human inspector would rely on for classification. Rules using these cues were developed and a voting scheme was implemented to incrementally accumulate classification evidence. Both methods compared favorably with human inspector performance. However, both approaches have a number of limitations. 1) Perfect registration of images is assumed. 2) Both lack desirable characteristics for a robust and adaptable vision systems such as combined control, focus of attention, hierarchical object models, and uncertainty management schemes. This system was tested on 825 subimages each of which contained a solder joint. However, the results reported included both training and testing samples and hence were overly optimistic. Reported accuracies of corrected classification were 74 - 94 percent for the statistical pattern recognition approach and 86 percent for the expert system approach. As this description should indicate, this system is very problem-specific.

Extensive references of industrial inspection systems are found in a survey paper [CHI88]. There are also special issues on industrial machine vision that appeared in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, No. 1 and 3, 1988.

3.2 Knowledge-Based Vision Systems in Other Applications

More powerful and sophisticated knowledge-based vision systems have been developed for application areas where real-time processing is not required. These areas include aerial image interpretation [NAG80, MCK85, HWA86, KUA88], outdoor scene interpretation [HAN88, LEV81, LI87], medical image analysis [NIE85, STA86], etc. Notable systems are reviewed and compared with the desirable attributes described in the last chapter. There is a good survey paper [BIN82] that provides a summary of model-based vision systems as of the early 80's. Most vision systems reviewed in that survey paper are omitted in this review.

Nagao and Matsuyama [NAG80] developed a vision system for automatically interpreting aerial images using structural analysis. The analysis process in this system is divided into the following steps: 1) image smoothing, 2) region segmentation, 3) extraction of characteristic regions, and 4) object-detection subsystems. The characteristic region is a set of regions that has characteristic properties, e.g., elongated, large, homogeneous. These characteristic regions are used to estimate approximate areas that are likely to contain objects. After extraction of characteristic regions, a set of object-detection subsystems performs knowledge-based analysis to locate objects in a scene. Each object-detection subsystem focuses its attention on specific local areas by combining several characteristic regions. Then it checks for the existence of specific objects by consulting the knowledge stored in the subsystem. All the information about the properties of elementary and characteristic regions and

recognized objects is stored in a blackboard. All the subsystems communicate indirectly only via the blackboard. When a region happens to be recognized as multiple objects, the system cancels the recognition of other objects except the most reliable one. The results of interpretation are very good; however, only five images were presented to show the performance of the system. Hence the robustness of the system has not been clearly established. Nagao and Matsuyama's system has most of desirable characteristics described in the last chapter. One exception is that no uncertainty management scheme is used in this system. Detailed comparison with the proposed system will be given in Chapter 4.

SPAM [MCK85] is a rule-based system that uses map and domain-specific knowledge to interpret airport scenes. The process begins with the labeling of individual regions in the image. These regions are collected to form consistent interpretations of the major components of an airport model. These interpretations are ranked on the basis of their overall spatial and structural consistency. Compared with the characteristics for robust vision systems, it has many characteristics such as confidence factors, combined control, and multiple levels of knowledge representations. However, it lacks good focus of attention mechanism so it is inefficient. Also, some other problems remain in the system. First, its segmentation method needs to be more refined and reliable. Second, the performance of the current system seems to be inefficient because it has so many rules that were encoded using an expert system building tool OPS5. The system successfully interpreted a difficult airport scene where the initial segmentations were poor. However, only one airport scene image was used for testing the performance of the system. Hence, this system's robustness is still in question.

SIGMA [HWA86] uses a symbolic, hierarchical model to represent the possible spatial organization of objects in an image. A set of image structures is computed at initial segmentation. Then, partial interpretations are incrementally constructed based on the results of the initial segmentation. However, during the construction, more image structures can be computed, if needed. When all construction activities finish, SIGMA provides a query-answering module for selecting "good interpretations" and displays the reasoning paths used to derive these interpretations. During the entire analysis, SIGMA maintains a database to record all the intermediate results generated at each stage. Although this system was suggested as a general paradigm for image understanding systems, little experimentation was done to demonstrate the full capability of the system. Only one image was used in to demonstrate this system's capabilities. Hence, system robustness remains to be demonstrated. Compared with the characteristics of robust vision systems, it has combined control, a focus of attention mechanism, and hierarchical object models. However, it does not employ any uncertain reasoning methods.

[KUA88] describes a constraint-based image understanding system for aerial image interpretation using a blackboard architecture. One notable feature of this system is that constraints between domain objects are represented and organized hierarchically similar to the way objects in the scene are represented and organized. It is said that this representation provides greater modularity. Also, it has combined control strategy, multiple levels of description of objects, opportunistic reasoning for focus of attention, and uncertainty management schemes. However, only one image was used to demonstrate this system's capabilities. So once again, system robustness is a concern.

The VISIONS system [HAN78, RIS84, DRA88, HAN88], which operates on multi-spectral images of outdoor images, can be divided into two main parts: the low-level segmentation processes and the high-level interpretation processes. The segmentation processes yield an intermediate symbolic representation (e.g., regions and lines and their attributes) of the image data without making use of any knowledge about specific objects in the domain. At the beginning of the interpretation process, object hypothesis rules are applied to the region and line representation to rank-order candidate object hypotheses. This initial iconic-to-symbolic mapping provides an effective "focus of attention" mechanism for later processing. Reliable hypotheses are selected as image-specific exemplars, and extended to other regions and lines through an object-dependent similarity matching strategy. At this point, intermediate grouping is performed for merging and modifying region and line elements to match expected object structures. Verification strategies exploit particular spatial relationships between an hypothesized object and other specific expected object labels or image features. Feedback to the lower-level process for more detailed segmentation can be requested for correcting errors detected in the interpretation process. Scene knowledge is represented in a hierarchical *schema* structure organized as a semantic network. Each schema defines a highly structured collection of elements in a scene or object. A number of desirable features are embedded in this system. The processing results presented include only two images. A detailed comparison of VISIONS with the proposed vision system is given in Chapter 4.

Levin and Shaheen [LEV81] discussed a proposed and partially implemented vision system that is both data directed and knowledge based. It has a short term memory for the image and most current interpretations of the scene and a long term memory containing a detailed model of the scene under consideration. After region segmentation, a continuous relaxation process is used to interpret regions of the segmented

image. Its advantages include a focus of attention mechanism and refinement of confidences by a relaxation method. Its disadvantages include bottom-up control and no multiple representation of object descriptions. Also, the only processing results shown was that of a single outdoor image. Consequently, the robustness of the system remains to be demonstrated.

Li and Uhr [LI87] describes a search and control strategy that is made feasible by the use of key features in a pyramid-like hierarchical structures. Following the model of preattentive processes, the transforms that successfully extracted key features are used to trigger other transforms to focus attention into areas and also for lateral search in the same pyramid level in extracting other key features. Since the searches are concentrated in relatively small and especially important subsets in the feature space, recognition is accomplished efficiently and effectively. The pyramid like hierarchical structure facilitates the combining of both bottom-up and top-down searches. Other advantages are uncertain reasoning, a focus of attention mechanism, and multiple levels of representation. However, the system lacks modularity. Also, it was tested on only four images of buildings. As before, system robustness is an issue.

Niemann et al. [NIE85] developed a system for obtaining a complete diagnostic description of an image sequence taken in nuclear medicine from the human heart. Organ contours are extracted after image smoothing, and are input for high level processing. A semantic net was used as knowledge representation, and conclusions are drawn by a production rule approach. Scoring of alternative diagnoses is based on fuzzy membership functions. Tests using several image sequences produced correct descriptions as compared to diagnosis by a physician. There are a number of desirable features appearing in this system such as confidence factors and

hierarchical object models. However, it suffers from a basic limitation in that its control is bottom-up, since high-level processing for diagnosis is performed based only on the low-level output, i.e., the left ventricular contour detected by low-level processing. A total of 40 image sequences were used as data for both the training and the testing of this system. While the author briefly mentions two series of tests that were conducted on the system, no processing results were presented. Hence, their system's robustness again remains in question.

ANGY [STA86] is a rule-based vision system that identifies and isolates the coronary vessels, given a subtracted digital angiogram of a chest. It embodies a domain-independent knowledge of segmentation, grouping, and shape analysis. Using both regions and edges, it determines such relations as parallel and adjacent, and attempts to refine the initial segmentation. Then it identifies vessels and eliminates all others by applying domain-dependent knowledge of cardiac anatomy and physiology to the objects and relations. This system is modular and flexible. It has two representation units: regions and lines. But, as disadvantages, it has bottom-up control, no uncertainty reasoning method, and no focus of attention mechanism. No test data were presented. Therefore, this system's robustness remains in doubt.

3.3 Summary of Review

Hopefully this discussion has shown that current industrial inspection systems are very application-specific and hence inflexible in their structure. Obviously, if industrial inspection technology is to become widely used such special-purpose, created from scratch, inspection systems are going to have to be replaced with more general systems that can be easily tailored to specific applications. As was stated in the introduction, this represents an objective of this research.

While the industrial inspection systems described are inflexible, some have proven to be very robust at the task they were designed to perform. As such these systems should not be treated too lightly.

The use of knowledge-based systems for industrial inspection purposes has been very limited. A very small number of the knowledge-based industrial inspection systems have been reported in the literature. The primary reason for not using knowledge-based systems is probably due to the fact that there are no inexpensive "high speed" computers currently on the market. This should change. By the year 2000 Intel predicts that a benchmark microprocessor will be able to execute 2 billion instructions per second. Such computational capability should spur the development of more knowledge-based industrial inspection systems.

The use of knowledge-based vision systems has been more prevalent in other application areas. A number of these systems were described. The objective of these systems is to be able to attack a rather "general" set of problems. Many have incorporated most of the desirable features mentioned in the last chapter. However, the real question is not whether these features were incorporated but rather how robust was the implementation. Of these "sophisticated" systems none have been extensively tested. Hence, the robustness of these systems remains to be established. Creating a robust vision system that can attack even a fairly general class of problems has proven to be a difficult task. An objective of this research is to create a general framework that will work only on a fairly general class of problems in a robust manner.

Chapter 4

OVERVIEW OF THE PROPOSED VISION SYSTEM

In Chapter 2, it was argued that a production system type structure should be used as the knowledge system architecture of the proposed vision system. Among various kinds of production or rule-based systems, the Blackboard framework [ERM80, ENG88] has been selected for this purpose. There are several advantages of the Blackboard framework over straightforward rule-based systems [ENG88]. 1) *Modularity*: Inherent modular characteristic of the Blackboard framework allows easy design, testing, and maintenance of the system. 2) *Dynamic control*: This framework provides a wide range of capabilities for controlling the problem-solving behavior of the system including top-down, bottom-up, and combined control. 3) *Efficiency*: The Blackboard framework provides a convenient method for employing a focus of attention mechanism. Focusing the attention of the system allows the best data and the most promising methods to be exploited first. 4) *Concurrency*: The modularity and flexible control structure of the Blackboard framework can support concurrent parallel processing, which can play an important role in speeding up the system operation [HAN88]. The Blackboard framework has been used in other very complicated problems such as speech recognition [ERM80] and natural scene understanding [HAN78, NAG80, LEV81, RIS84, HAN88].

In what follows, the general Blackboard framework will be briefly described. Then, the Blackboard framework used in the proposed vision system framework will be described.

4.1 The Blackboard Framework

The blackboard framework is a problem solving model that is a highly structured, special case of opportunistic problem solving. It usually consists of three separate components: *knowledge sources* (KSs), *blackboard data structure*, and *control modules* [ENG88]. This is illustrated in Figure 4.1.

4.1.1 The Knowledge Sources

The domain knowledge needed to solve a problem is partitioned into KSs, which are kept separate and independent of each other. The objective of each KS is to contribute information that will lead to a solution to the problem. A KS takes a set of current information on the blackboard and updates it in the manner encoded in its specialized knowledge.

Each KS is a condition/action module. The condition component dictates when the KS can be put to use and the action component specifies the contribution that the KS will make. A KS can communicate with other KSs only through a global data base, a blackboard which records hypotheses generated by KSs. One can view a KS as a large production system rule. The major difference between a production system rule and a KS is the grain size of the knowledge each holds. The grain size of a KS is usually larger than a rule.

4.1.2 The Blackboard Data Structure

The purpose of the blackboard is to hold all the computational and the solution-state data needed by and produced by the KSs. The KSs use the blackboard data to interact with each other indirectly.

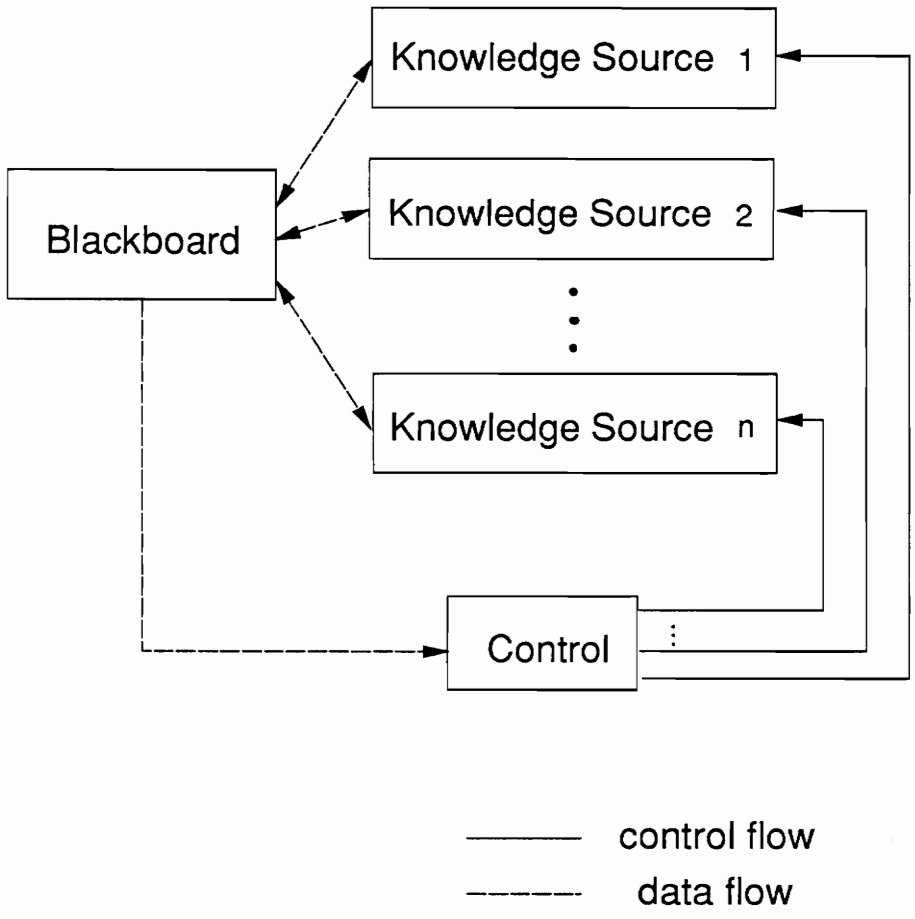


Figure 4.1. The Blackboard framework.

The blackboard consists of objects from the solution space. These objects can be input data, partial solutions, alternatives, and final solutions (and possibly, control data). The objects in the blackboard are hierarchically organized into levels of analysis. Properties of objects on one level serves as input to a set of KSs, which, in turn, place new information on the same or other levels.

4.1.3 Control

The knowledge sources respond opportunistically to changes on the blackboard. There is a set of control modules that monitor the changes on the blackboard and decide what actions to take next. Various kinds of information are made globally available to the control modules. The information can be on the blackboard or kept separately. The control information is used by the control modules to determine the focus of attention. The focus of attention indicates the next thing to be processed. The focus of attention can be either the KSs (i.e., which KSs to activate next) or blackboard objects (i.e., which objects to pursue next), or a combination of both (i.e., which KSs to apply to which objects).

The solution is built one step at a time. Any type of reasoning step (data driven, goal driven, and so on) can be applied at each stage of solution formation. As a result, the sequence of KS invocation is dynamic and opportunistic rather than fixed and preprogrammed. However, in practice, it is more efficient to use both sequential and opportunistic scheduling of KSs, depending on the characteristics of KSs, rather than to use purely opportunistic scheduling.

4.2 A Blackboard Framework for the Proposed Vision System

Within a Blackboard framework, the proposed vision system can be conceptualized to have two modules: a low-level segmentation module and a high-level recognition

module. The block diagram of the system is shown in Figure 4.2. The function of each module will be briefly described first. Then a Blackboard framework adapted for the computer vision system will be described in detail.

The low-level module consists of two parts : segmentation of an input image containing the material to be inspected and extraction of region properties. The main purpose of the segmentation in the web inspection problems is to reduce the volume of the image data to be processed by extracting potential defects from the image. A multi-thresholding method is used to do the segmentation. (Refer to Chapter 2 for a detailed discussion about various segmentation methods.) The thresholds are adaptively and automatically selected on the basis of valley points and inflection points of the gray level histogram of the input image. A priori knowledge specific to web inspection problems is incorporated in selecting the thresholds in order to obtain segmentation results with acceptable accuracy. After segmentation, the low-level module identifies all connected regions, eliminates small noise regions, merges adjacent regions that have similar average gray levels, and computes region properties of merged regions. The result of this processing is a more accurate and concise description of each of the regions passed on for higher level processing.

The high-level module performs the scene analysis operation, i.e., recognizes defect areas. Basically, the high-level module consists of three parts: a focus of attention mechanism, defect detection procedures, and verification using contextual information or spatial constraints. Confidence vectors are used as the focus of attention mechanism to screen candidate regions in an effort to determine the type or types of defect each region might represent. A region's confidence vector determines which defect detection procedures will be applied to that region.

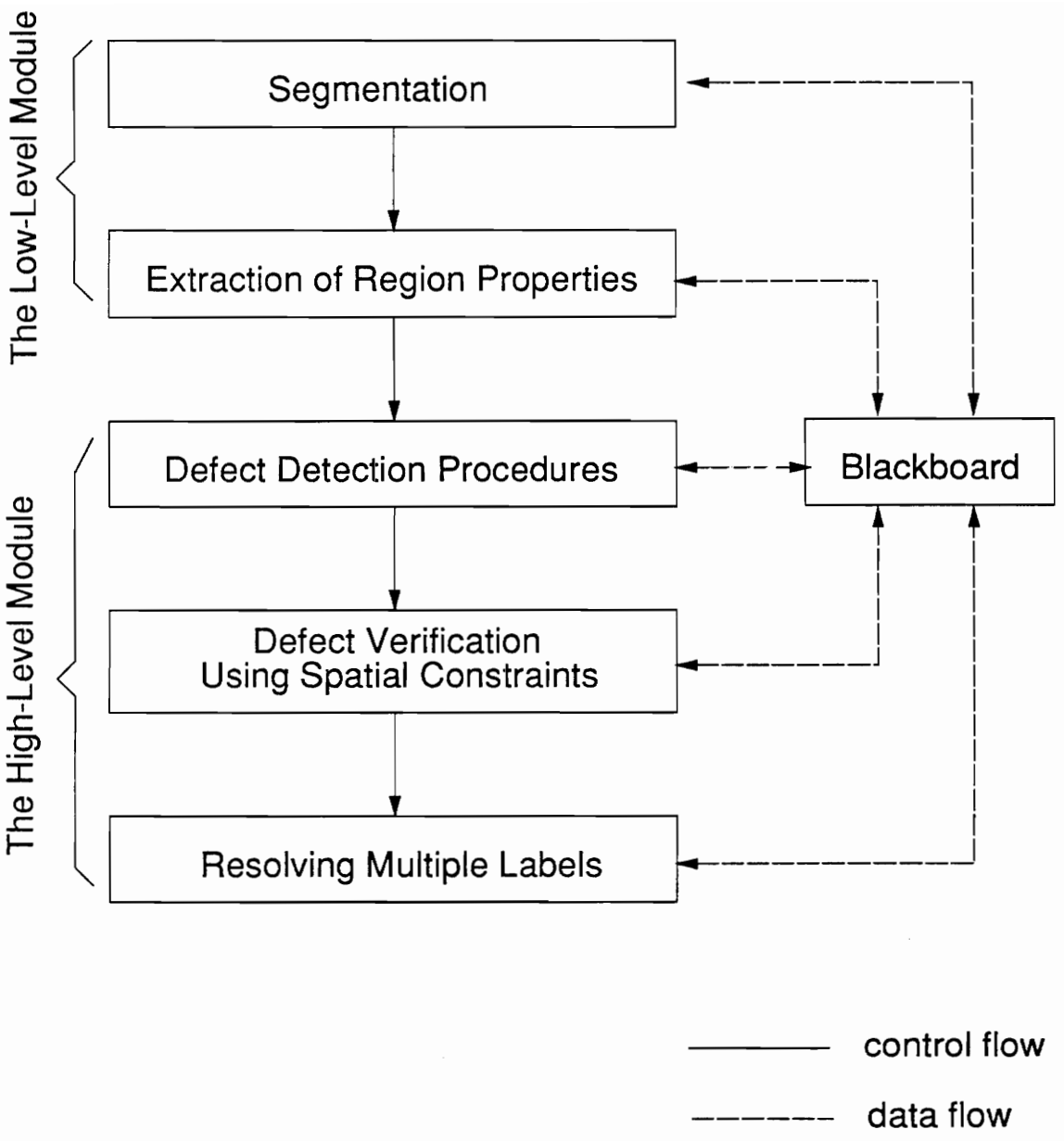


Figure 4.2. Block diagram of the proposed vision system.

Each defect detection procedure is designed to detect a particular type of defect. Each defect detection procedure operates completely independent of the other defect detection procedures. Each defect detection procedure is applied only to regions in the image whose confidence vector indicates it might be the corresponding defect type. In each defect detection procedure, top-down processing using special operators applied to the original image is used when more evidence is needed to support a hypothesis. Each defect detection procedure consists of two steps: initial labeling and labeling verification. The initial labeling step constructs the sets of connected regions, called DEFECT_OBJECTs. Each DEFECT_OBJECT is a set of connected regions all of which have been assigned the same defect label by a defect detection procedure. The label verification step tries to verify the assigned label of each DEFECT_OBJECT generated.

Finally, the label of each DEFECT_OBJECT is further verified using spatial constraints among adjacent DEFECT_OBJECTs. This is necessary because each defect procedure encodes knowledge specific only to its defect type and does not contain contextual knowledge among the different defect types.

4.2.1 Knowledge Sources

Several knowledge sources exist in the low-level vision module. The first one, SEG, performs a segmentation. A KS CCL does the connected component labeling. A KS SRE eliminates small noisy regions. A KS RMERGE (region merge) merges adjacent regions which are similar in average gray levels. Finally, a KS RPT extracts region properties for each region.

There are a number of knowledge sources for the high-level vision module. A KS, INIT, does some initialization. A KS CALCV calculates a confidence vector for

each region based on its region properties, using fuzzy logic. The *i*th component of the confidence vector for region R is a confidence value that R is a defect of type *i*. LBLCON is a KS for labeling unlabeled regions using spatial contextual dependency. VRFDFC is a KS for verifying labels assigned to regions by defect detection KSs using spatial constraints between different defects. RMLBL is a KS that resolves labeling ambiguity if multiple defect labels are assigned to a region. Other KSs are defect detection procedures, each of which is designed to detect one kind of defect in the surface of the material being inspected. It is natural to partition knowledge about defects, according to defect type. The major advantage in partitioning knowledge this way is the ease with which additional defects can be incorporated into the system by simply adding new defect detection procedures.

4.2.2 Blackboard Data Structure

The blackboard structure is organized into the following entities:

Original image
Symbolic segmented image
Region property table
DEFECT_OBJECT property table
Region adjacency matrix
DEFECT_OBJECT adjacency matrix

The symbolic segmented image is a symbolic image where each pixel in a region is labeled with the same unique region number. The symbolic image is generated by the segmentation step in the low-level module. The region property table is a collection of property lists for all the segmented regions. A property list is associated with each region; it consists of a region attribute vector, a confidence vector, a

pointer vector, and an interpretation label. The region attribute vector consists of various region properties whose values are assigned by the low-level module. The interpretation label is the label (normal surface or one of the defect types) assigned to the region. Confidence vectors, pointer vectors, and interpretation labels are assigned values by the high-level module.

There are two basic units in the blackboard: regions and DEFECT_OBJECTs, that are hierarchically linked by a part-whole relationship. A DEFECT_OBJECT consists of a set of connected regions that have the same defect label. The concept of the DEFECT_OBJECT is illustrated in Figure 4.3. The *i*th component of a region's pointer vector points to a DEFECT_OBJECT with *i*th defect label associated with the region. For example, in Figure 4.3, if DO1 has *i*th defect label, the *i*th component of each pointer vector associated with regions R1, R2, R3, and R3 point to DO1. Note that one can determine whether a region is assigned multiple defect labels by simply checking how many DEFECT_OBJECTs are pointed to by the region's pointer vector. Each DEFECT_OBJECT generated stores its properties on the DEFECT_OBJECT property table. The DEFECT_OBJECT property table has the same structure as the region property table except for the pointer vector field. Each component of the pointer vector associated with a DEFECT_OBJECT is always 0 because there is no parent node of the DEFECT_OBJECT. If a DEFECT_OBJECT is generated and verified by a defect detection procedure that was designed to detect defect type *i*, the interpretation label of the DEFECT_OBJECT is assigned defect type *i*. The region adjacency matrix is defined as a matrix whose (i,j)-th element is the common perimeter between region *i* and region *j* divided by the perimeter of region *i*. The DEFECT_OBJECT adjacency matrix is defined similarly. These adjacency matrices provide contextual information among regions or DEFECT_OBJECTs that is vitally important in interpreting images.

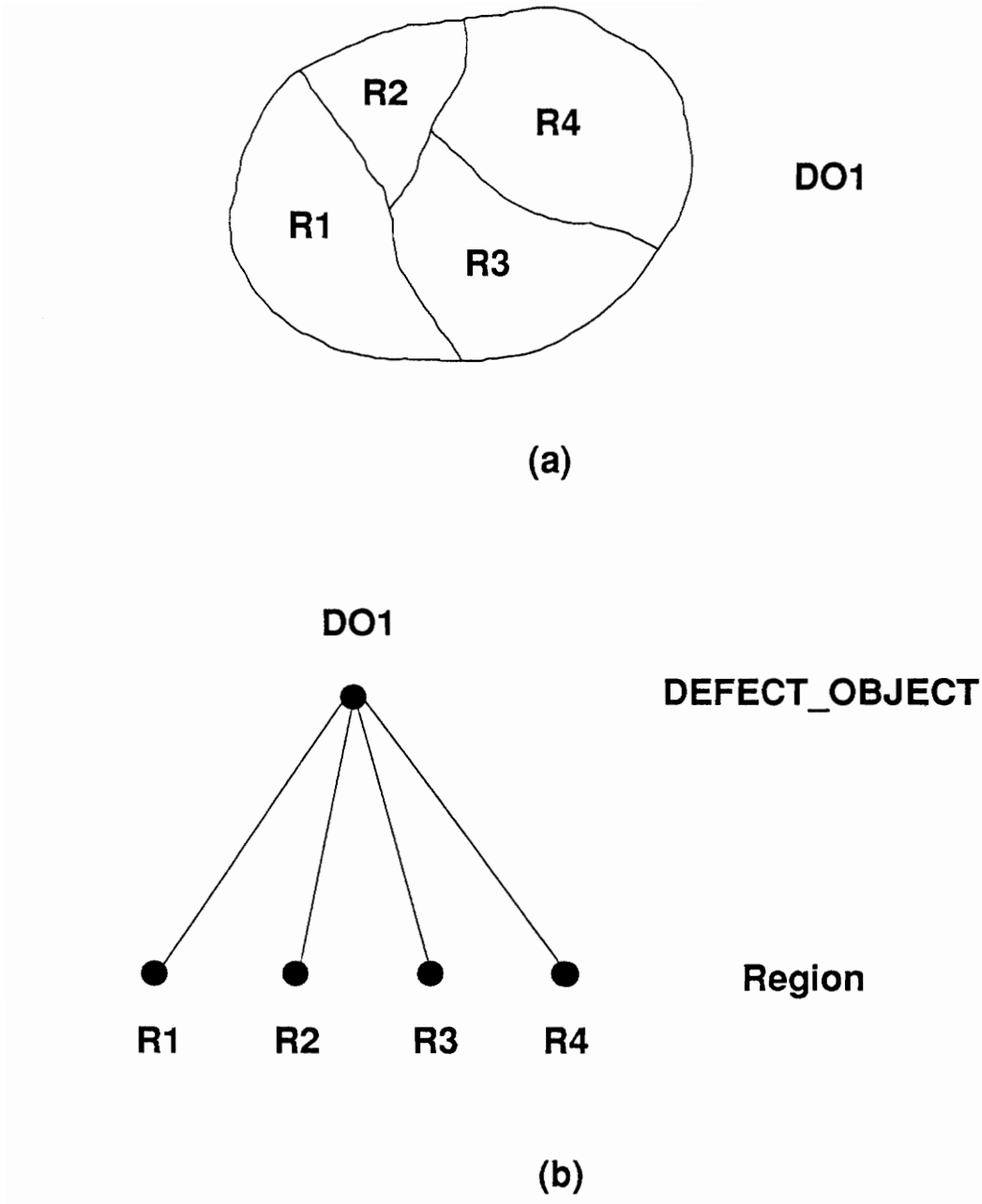


Figure 4.3. A DEFECT_OBJECT consisting of four regions. (a) Four connected regions labeled as the same defect type. (b) A DEFECT_OBJECT DO1 generated.

4.2.3 Control

The control of processing flow is done using a combination of both sequential control and opportunistic control. This combination is more efficient than purely opportunistic control used in typical blackboard systems. Control in the low-level module is sequential. KSs in the low-level module are applied in the following order: SEG, CCL, SRE, and RMERGE. At an initial stage of high-level processing, KS INIT is applied for initialization that includes calculation of confidence vectors. Then, KSs for defect detection are run independently and opportunistically. A region's confidence vector is used as a "focus of attention" mechanism to selecting which defect detection KSs should be activated to process the region. Once KSs for defect detection have finished their operations, KS VRFDFC (verification of defects using spatial constraints) and KS RMLBL (resolving labeling ambiguity) are performed in sequence. Before and after applying VRFDFC, a determination is made as to whether there are any regions that are still unlabeled. If there are unlabeled regions, then KS LBLCON is invoked to label those regions using spatial contextual dependency.

4.3 Comparison with Other Knowledge-Based Vision Systems

The structure of the proposed vision system was mainly influenced by two knowledge-based computer vision systems: Nagao and Matsuyama's system for aerial image interpretation [NAG80] and Riseman and Hanson's system VISIONS for natural scene understanding [HAN78, RIS84, DRA88, HAN88]. In this subsection, the proposed vision system is compared with these knowledge-based vision systems in various aspects. Although the proposed vision system shares some aspects with

these two vision systems, there are also several major differences among these images. Even in the aspects where there is a similarity, the proposed vision system has significantly different internal structures than the other systems.

First, each system was designed to work on a different problem domain. The proposed vision system is for industrial web inspection while the others are for aerial image and natural outdoor scene understanding. The requirements for the web inspection systems is significantly different from those for the other systems. Obviously, this difference in requirements means the vision systems are designed based on different performance criteria.

With regard to knowledge structures, all the three systems use Blackboard frameworks. (Although *schema system* was actually used in the VISIONS system, it is basically a variation of the Blackboard framework.) The advantages of the Blackboard systems over straightforward rule-based systems was discussed earlier.

All three have one KS for detecting each *object* class / *defect* category. Such an assignment of one KS per type of defect or object allows an implementation of coarse-grained parallelism by concurrently running each KS on a different processor.

As to the control strategy, a mixed strategy of a top-down and a bottom-up control is employed in all systems. Advantages of this mixed control were documented in a previous chapter.

Significantly different segmentation methods were used in the three vision systems. A simple region growing segmentation scheme is used in the Nagao and Matsuyama's system after an image smoothing operation. Both a region segmentation based on histograms of artificially partitioned area of an image and a line segmentation method

are used in the VISIONS system. A multi-thresholding method based on one global gray-level histogram is used in the proposed vision system due to the critical requirement for high speed processing of large high resolution images.

It was argued in Chapter 2 that a method for uncertainty management is necessary in a vision system because there is uncertainty in all stages of the visual processing. Local errors are introduced by the imaging and digitization process, by imperfect KSs (e.g., segmentation) as the result of incorrect heuristics, or by inherent ambiguity in the scene itself. In Nagao and Matsuyama's system no uncertainty management scheme is used; all decisions are Boolean TRUE or FALSE type. The VISIONS system uses a hybrid approach to uncertainty management. For global communication among KSs it uses a coarse five point numeric scale, where confidence values range from "no_evidence", the lowest value, through "slim_evidence", partial_support", "belief", and finally "strong_belief". On the other hand, for local communication within a object detection KS, uncertainty is treated symbolically using *endorsements* that are a symbolic record of the object-specific evidence supporting or denying the presence of an object instance. The proposed vision system maintains a continuous valued confidence measure that is updated by each defect detection KS.

Each system employs a focus of attention mechanism so that the scene analysis operation can be performed as efficiently as possible. This focus of attention mechanism extracts possible candidate areas that might belong to a defect or object category of interest with a minimum of computational effort. In Nagao and Matsuyama's system, several kinds of regions with prominent features (characteristic regions) are first extracted by simple picture processing methods. Then these characteristic regions are combined using Boolean intersection (AND), union (OR), and complement (NOT) operations to produce candidate regions for each object

class. One critical limitation of the use of this type of Boolean logic is that it does not allow partial matches. Suppose, for example, candidate regions for an object class are produced by intersecting several characteristic regions. Then if a region does not belong to one of these characteristic regions but belongs to all the rest, this region will not be a candidate region for the class. In VISIONS, for a specific object class, various simple token (region/line) attribute rules are defined, each of which maps an attribute value into a confidence that the token is an instance of the object. These simple token rules are combined into a complex rule for an object class, i.e., an object hypothesis rule, using a hierarchical weighted average. Regions and lines having a high response of an object hypothesis rule are selected as focus of attention areas for the object class. In the proposed vision system, characteristics of each defect are first described using a predefined set of linguistic qualifiers and operators. Then, each qualifier is assigned a fuzzy membership function that is a function of one or more region property values. Based on these membership functions, the elements of the confidence vector of each region are computed using intersection, union, and complement operation on fuzzy sets. (The ith element of the confidence vector of region R is a confidence value of R being defect type i.) Regions with a high confidence value for a defect type are considered as focus of attention areas for that defect type.

To verify working hypotheses, the proposed vision system uses a nonparametric supervised classifier using a 3-layer neural network. (Actually, a rule-based and a k-nearest neighbor approach were also implemented for this purpose. However, a comparison of these three approaches on the application of lumber inspection presented in Chapter 7 have led to the selection of the neural network approach for verification of working hypotheses.) The other two systems basically use rules for verifying hypotheses.

Finally, these systems differ in a very important way. Robustness of a vision system is very important in any application domain. It is a particularly critical requirement in for vision systems used in industrial inspection. An important criterion for evaluating robustness is the number of images the system can successfully analyze. Admittedly, Nagao and Matsuyama's system and the VISIONS system are both aimed at very complex problem domains (outdoor scene / aerial image interpretation). However, Nagao and Matsuyama's system was tested on only five aerial images and the VISIONS system was tested on only two outdoor scenes (a house and a road). Obviously, the good testing results these systems obtained on this very limited number of images does not in anyway guarantee robustness. On the other hand, the proposed vision system has been successfully tested on approximately 100 images of hardwood lumber. The problem of rough lumber inspection is also a nontrivial problem. (Details of this testing results will be given in Chapter 7.)

Chapter 5

THE LOW-LEVEL MODULE

5.1 Segmentation

The first KS to be invoked in the low-level module is SEG, a KS for performing image segmentation. Its purpose is to reduce the volume of data that needs to be processed by the higher level modules. This KS performs two functions. First it separates pixels of background from pixels of the material being inspected. Then it separates pixels of normal surface area from pixels that might be from areas containing defects. This KS uses as input a full color image of the material being inspected with 256 gray levels for each color.

The idea for separating pixels of background from pixels of the material is to choose a background color that trivialize this segmentation operation. One way to do this is to choose the background color so that either the red/green plane, the red/blue plane, or green/red plane can be used to do the segmentation. The idea is to use a simple 256x256 look up table to do the segmentation. The background area of the look up table can be easily determined by examining images of the background and images of the background in shadow. An example of how one can use color to trivialize background/material surface segmentation is given in [CON89].

Differentiating pixels of normal surface area from pixels that might be of a defect is a more difficult problem. Although a color image is input to the system, only the black and white (B/W) version of the color image is used here. The reasons for this are twofold: 1) This greatly saves computation time. 2) Although some

application problems require the use of color information, there are still plenty of inspection problems where using only B/W gray-level information will discriminate defects from normal surface area. To keep computation time minimal, color information is used only in the areas where it is required to identify the type of defects detected. For application problems that require color information to do the segmentation, a color segmentation method could be developed and used instead of the current B/W segmentation method. But other structures in the proposed vision system would remain unchanged.

A histogram based thresholding is used for the reasons discussed in a previous chapter. A number of thresholding techniques have been reported in the literature since Chow and Kaneko's classic paper [CHO72]. (An extensive survey on the various thresholding techniques is given in [SAH88].) Most of these deal with bilevel thresholding using a single threshold. However, when gray level histograms of the image are multimodal, using only a single threshold destroys the information about the possible multimodal characteristics of the image data. Hence a multi-thresholding technique should be used to handle the possible multimodes of the gray-level histogram.

One other possible technique that can overcome the weakness of the global single thresholding is *local thresholding*. In local thresholding, the original image is artificially partitioned into smaller subimages and a threshold is determined for each subimage. A smoothing operation is applied to the thresholded image to remove gray-level discontinuities that arise at the boundaries of the two different subimages. The Chow and Kaneko's technique [CHO72] is an example of local thresholding. In this dissertation, however, local thresholding was not favored due to greater computational complexity than the multi-thresholding. For example, if an original

image is partitioned into n subimages, then the histogram of each subimage must be analyzed to determine a threshold. Hence, the total time complexity of the local thresholding method would be approximately n times the time complexity of the multi-thresholding.

While there have been many multi-thresholding methods reported in the literature (e.g., [BOU85, KOH81, WAN84]), there seems to be no proven domain-independent technique that will work well on all the web inspection problems. Therefore, a multi-thresholding method was developed here, which uses a priori knowledge or assumptions about the industrial web inspection problem. The assumptions made for this method are: 1) Total area of defects are very small compared with that of normal surface. 2) Areas of defects are always darker than those of normal surface. These assumptions are valid in many web inspection problems (e.g., inspection of lumber or white paper). This method involves setting multiple thresholds based on both valley points and inflection points of the gray level histogram. Inflection points are examined as candidates for thresholds because small defects frequently embed themselves in the normal area peak without forming a peak of their own. Figure 5.1 shows a valley point and an inflection point of a gray-level histogram, that would be selected as thresholds by the proposed multi-thresholding method.

The inflection points or zero-crossings of the second derivative were originally used for edge detection [MAR80] and signal description [WIT83]. Later they were used in the analysis of histograms [CAR85, FUN85] but in different ways from the method presented here. In [CAR85], inflection points of a Gaussian smoothed histogram were used to roughly estimate parameters of the underlying distribution when the smoothed histogram is approximated by the sum of normal distributions. In [FUN85] thresholds for CT images of sawlogs were selected based on the

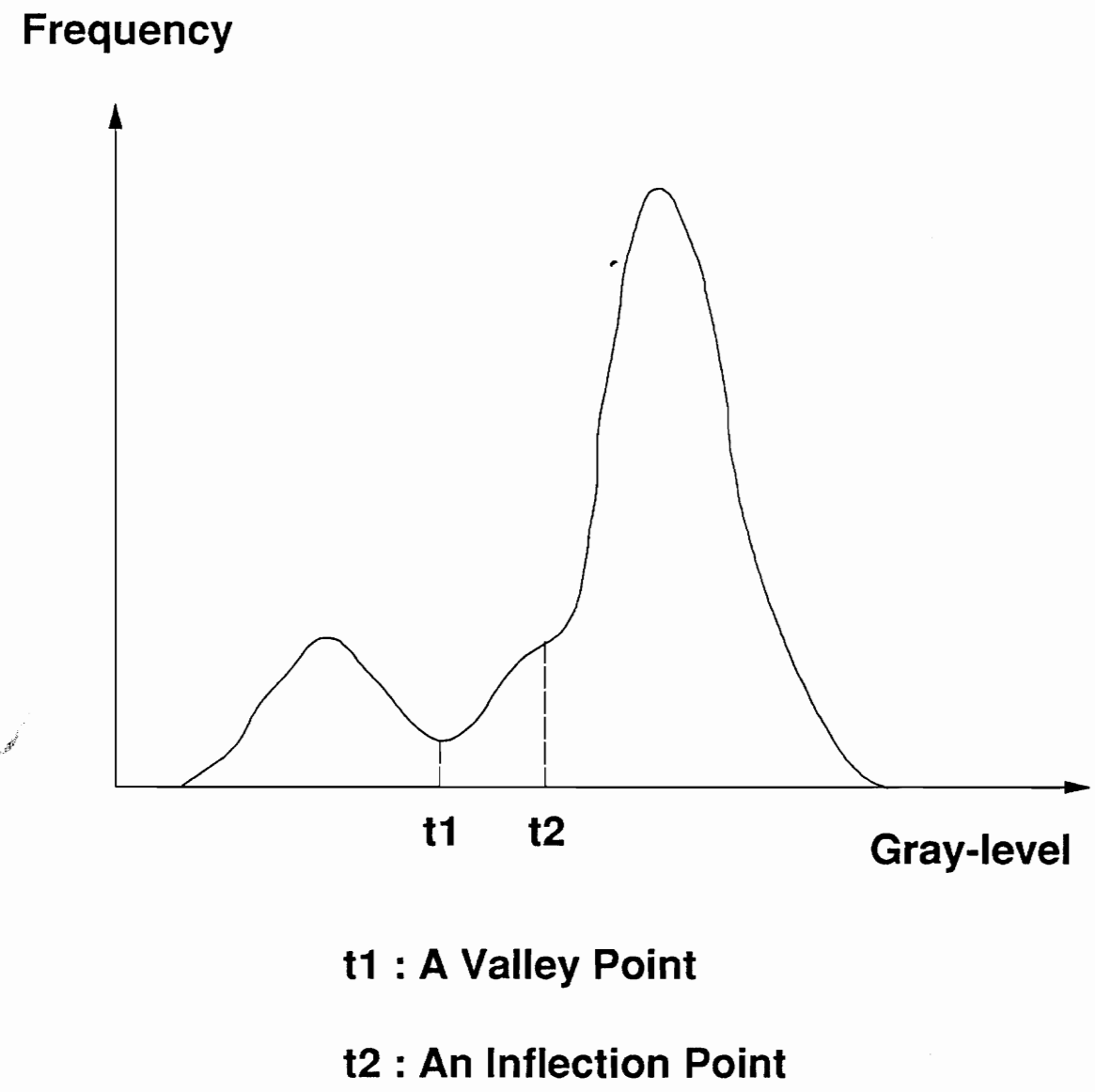


Figure 5.1. Thresholds selected t_1 and t_2 in a gray-level histogram.

zero-crossings of the second and third derivatives of the histogram. Some assumptions were made about the structure of the histogram in selecting the thresholds. These assumptions are valid only for the CT images of sawlogs. The problem-dependent assumptions used to create the proposed multi-thresholding method are significantly different from those used by [FUN85].

The steps used to perform this segmentation are given below.

Step 1. Construct a gray level histogram using only black and white information derived from the color pixels of the material. Let Hist(k) denote the value of the histogram at gray level k.

Step 2. Choose conspicuous valley points in the histogram as thresholds, $t_1,t_2,...,$ and t_n.

Step 3. To find possible smaller defects that might embed themselves in the normal area peak, let t_0 be the minimum gray level and t_{n+1} be the maximum gray level in the histogram. Find the interval $[t_i,t_{i+1}]$ within which the largest peak exists. This peak will always be the normal area peak.

Step 4. Perform a Gaussian smoothing [SHI87] on the histogram in the interval $[t_i,t_{i+1}]$. The Gaussian function is given by

$$G(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

where x is a variable representing gray-level. This smoothing is needed to avoid spurious inflection points in the histogram. The standard deviation σ of the Gaussian filter used is a problem-dependent variable.

Step 5. Find the subinterval $[t_j,t_{j+1}]$, of $[t_i,t_{i+1}]$ such that Hist(k) $\geq$ CMIN for all k $\in$ $[t_j,t_{j+1}]$. CMIN is chosen by experimentation, depending on the typical size of the defects present on the material surface. (CMIN can be selected as X percent

value of the largest value of Hist(.).) The reason for choosing the subinterval is that when Gaussian smoothing is applied to intervals where the values of Hist(.) are small typically it gives many spurious inflection points.

Step 6. If $t_j > t_i$+MIND select t_j as a threshold. MIND is a parameter which controls the minimum distance between consecutive thresholds.

Step 7. For each p in the subinterval $[t_j,t_{j+1}]$ determine if (a) the first derivative of the smoothed histogram at p is positive and the second derivative of the smoothed histogram changes from negative to positive or if (b) the first derivative of the smoothed histogram at p is negative and the second derivative of the smoothed histogram changes from positive to negative. If either of the above is true, p is selected as a threshold.

With regard to Step 7, it should be noted that even with Gaussian smoothing the second derivative is frequently noisy. To prevent this noise from adversely affecting inflection point determination, an interval about p is used to determine the sign change of the second derivative. A voting scheme is applied to each side of the interval to determine the sign of the second derivative. Experimentation has shown this to be an effective way to reduce the affect of noise on inflection point determination.

Once the thresholds have been found, regions formed by each pair of thresholds are marked. All pixels having gray level value between two consecutive thresholds are marked the same.

5.2 Connected Component Labeling

After the initial segmentation is performed using the above multi-thresholding method, connected regions are formed from pixels that have the same identification number.

Connected component labeling [SHI87] is performed by a KS CCL to give each connected region a unique number. All connected regions are found and each such region is given a unique label to differentiate it from the rest of the connected regions. Currently, 4-connectedness is used to form the connected regions since finding 4-connected components is faster than finding 8-connected components. Further, experiment has shown that 8-connectedness makes little improvement in the region boundaries over 4-connectedness.

5.3 Small Region Elimination

Unfortunately, segmented output typically contains many small meaningless regions due to both rough surface structure of a material and digitization noise. Also, some regions are overfragmented. Therefore, to provide more reliable region descriptions to the high-level module, it is necessary to eliminate those small regions and to merge overfragmented regions. Reducing the number of regions also decreases computation time needed to extract properties from all the regions. The small region elimination operation is performed first because it is computationally fast and it usually significantly reduces the number of regions produced by the original segmentation. A region merging operation is then performed. This region merging operation is computationally more complex than the small region elimination operation because the former is based on a statistical test. It is applied to only those regions that remain after the small region elimination operation.

The small region elimination is done by KS SRE using the following steps [PAV77].

Step 1. Select R_1, the region with the smallest area. If the area of R_1 is greater than THAREA, a predetermined area threshold, return. Otherwise proceed to the next step. (THAREA is chosen empirically.)

Step 2. Find R_2. Of all the regions adjacent to R_1, R_2 is the region that is the most similar in average gray level. Assign R_1 the region label of R_2.

Step 3. Repeat step 1 and 2 until there exists no region with area less than THAREA.

5.4 Region Merging

After all the small regions have been eliminated, further region merging is performed by KS RMERGE. A rule-based region merging scheme was proposed in [HAN88]. In this scheme, given two adjacent regions, a set of rules are evaluated, each of which measures the similarity of the two regions on the basis of a single feature and returns a value centered on one (implying no information about merging); values less than one indicate a vote for a merge (bounded by zero which guarantees a merge), and values greater than one indicate a vote against a merge. The results of such rules are combined to yield the overall similarity measure through a combination function, the product of the score of each rule. The value of this overall similarity measure becomes the basis of deciding on merging the two regions.

The approach to the region merging operation used here is also rule-based; however, both evaluation and combination of each rule's response are based on a simple application of the Dempster-Shafer (D-S) theory of evidence [SHA76]. In Appendix A, fundamental D-S theory is briefly reviewed. Given two adjacent regions, each rule or test in the series that is applied checks for the validity of merging operation based on its own criterion, and outputs a confidence value that the adjacent regions should be merged together. The reason for using multiple tests rather than a single test is that in general, multiple sources of evidence are likely to provide a more reliable indicator than a single source of evidence. Confidence values generated by

multiple tests are combined to produce a final confidence or belief value using Dempster's rule of combination. If this final confidence is high, the two regions are merged.

There are a number of reasons for selecting D-S theory in the region merging over other major uncertain reasoning methods available, e.g., Bayesian reasoning, fuzzy logic, and MYCIN's Certainty Factor (CF). The main disadvantages of the Bayesian reasoning include: 1) A priori probabilities must be assigned, an assignment that is difficult to do in this problem. 2) It is difficult to represent the measure of ignorance [HEN88]. The MYCIN's CF model is an *ad hoc* technique. In fact, the Dempster combination rule includes the Bayesian and CF combining functions as special cases [GOR84]. This leaves fuzzy logic as the remaining possibility. Suppose there is one test with very low confidence and many other tests with high confidence. Then final confidence value computed using fuzzy intersection operations is completely determined by the lowest confidence value, even if many other tests have good credibility. This seems to be inconsistent with the human decision making. On the other hand, Dempster's rule of combination seems to pool independent multiple sources of evidence [SHA76]. In the above case, the final confidence is affected not only by the lowest confidence but other confidence values as well. Thus the final confidence computed in such a case would not be as pessimistic as the value computed using fuzzy intersection operations. Also, D-S theory has a capability of representing the degree of ignorance. Detailed general comparisons of several uncertain reasoning theories including the D-S theory can be found in the literature [HEN88, LEE87].

In the D-S theory of evidence, a *frame of discernment* Θ is defined as a set of all possibilities that are mutually exclusive and exhaustive. Any subset of Θ is a

hypothesis. In the region merging problem, $\Theta = \{merge, don't\_merge\}$, where *merge* (*don't_merge*) represents a hypothesis that the given regions should (should not) be merged. Associated with each piece of evidence is a *basic probability assignment* (*bpa*) that represents the impact of the evidence on the subsets of Θ. In the region merging problem, each test can be viewed as a independent piece of evidence that generates its own *bpa*. Let m_i be *bpa* generated by *i*-th test. Since by definition $\sum_{A \subseteq \Theta} m_i(A) = 1$ and $m_i(\phi) = 0$, $m_i(\Theta)$ given by

$$m_i(\Theta) = 1 - (m_i(merge) + m_i(don't\_merge))$$

can be greater than 0. $m_i(\Theta)$ defines the measure of *ignorance* representing the extent to which the *i*-th test cannot discriminate between *merge* and *don't_merge*. Contrast this with Bayesian reasoning that always requires $m_i(\Theta) = 0$. As such Bayesian logic provides no such measure. Since Dempster's rule of combination is commutative and associative, m_i's from any number of tests can be combined regardless of order. Let m_1 and m_2 be *bpa*'s corresponding to two independent pieces of evidence. Then new *bpa* m can be calculated using Dempster's rule of combination given by

$$m(A) = K \sum_{X \cap Y = A} m_1(X) m_2(Y) \quad if\ A \neq \phi$$

$$= 0 \quad if\ A = \phi$$

where

$$K^{-1} = 1 - \sum_{X \cap Y = \phi} m_1(X) m_2(Y).$$

Presently, two tests are used to check for validity of region merging. These are **GTEST** (test for difference of global average gray-level) and **LTEST** (test for

difference of local average gray-level). (If other tests are available, they could be added to the set of tests in order to make the decision for region merging more reliable.) The steps involved in the region merging are described below.

Step 1. Select R_1, the region with the smallest area.

Step 2. Find R_2. Of all the regions adjacent to R_1, R_2 is the region that is the most similar to R_1 in average gray level.

Step 3. Apply both **GTEST** and **LTEST** to R_1 and R_2. Let m_1 and m_2 be *bpa*'s generated by these tests.

Step 4. Combine m_1 and m_2 into m, the final *bpa*, using Dempster's rule of combination.

If *m(merge)* is greater than 0.5, then assign R_1 the region label of R_2.

Step 5. Repeat previous steps until all pairs of adjacent regions have been tried.

The **GTEST** referred above is a statistical T-test for equality of *average* gray levels between adjacent regions R_1 and R_2. The underlying assumptions associated with the use of the T-test is that the gray levels of the pixels in both R_1 and R_2 are independent, that they have identically distributed normal distributions, and that the variances of the distributions are unknown but equal. The steps associated with using the T-test are given below. Similar statistical tests have been used by other investigators, for example, [YAK76, HAR85]. The **GTEST** involves the following steps:

Step 1. Compute the sample mean, $\overline{X}_1$, and the sample variance, S_1^2, of the gray levels of R_1. Also compute the sample mean, $\overline{X}_2$, and the sample variance, S_2^2, of the gray levels of R_2.

Step 2. Let n_1 and n_2 be the number of pixels in R_1 and R_2, respectively. Using

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

compute the statistic

$$t_{obs} = \frac{\overline{X_1} - \overline{X_2}}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}.$$

Step 3. Compute m_1 based on the statistic t_{obs}. The higher the absolute value of t_{obs}, the larger $m_1(merge)$. The function $m_1(.)$ is determined experimentally. For example, one possible simple assignment of $m_1(.)$ might be:

IF $|t_{obs}| < T_1$ THEN

$m_1(merge) = 0.5$; $m_1(don't\_merge) = 0.1$

ELSE IF $|t_{obs}| > T_2$ THEN

$m_1(merge) = 0.1$; $m_1(don't\_merge) = 0.5$

ELSE

$m_1(merge) = 0.2$; $m_1(don't\_merge) = 0.2$

END IF

where T_1 and T_2 are parameters that are determined experimentally.

The **LTEST** referred above is also a T-test to check if *local* gray level averages between two adjacent regions are similar. The **LTEST** involves the following steps:

Step 1. Find W, the intersection area of two focus windows of R_1 and R_2, as illustrated in Figure 5.2. The focus window of region R is determined based on its Minimum Bounding Rectangle (MBR), being slightly larger (e.g., 10 % larger) in size than the MBR.

Step 2. Apply the same steps as in **GTEST** except that only pixels of R_1 and R_2 *inside W* are considered in the steps rather than all pixels of R_1 and R_2. Let m_2 be the *bpa* generated by **LTEST**. m_2 is determined in a manner similar to the

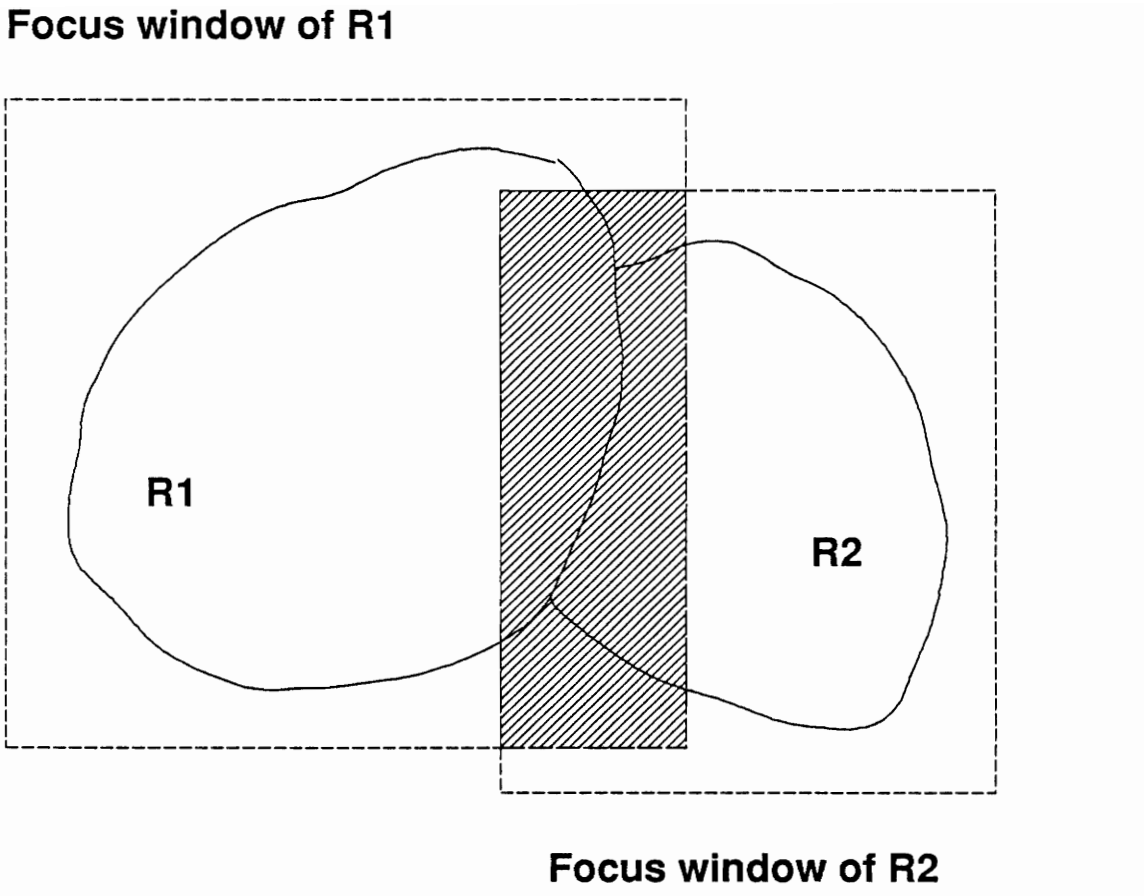


Figure 5.2. Focus windows of region R_1 and region R_2

way m_1 was determined.

5.5 Extraction of Region Properties

The next step of the low-level module is to extract a variety of properties of segmented regions that are used for labeling regions as normal surface area or a particular type of defect in the high-level module, that is, the recognition module. After the region merging operation is complete, KS RPT extracts a vector of the basic features from each of the resulting regions. The set of features extracted is problem-dependent. As an example, a set of basic features extracted for the problem of lumber inspection are shown below. For each region R, basic features extracted are:

1) *Area*, the number of pixels in R;

2) *Average gray level*, average of all the gray levels of the pixels in R;

3) *Center of mass*,

$$(\bar{r},\bar{c}) = \left(\frac{1}{N}\sum_{(r,c)\in R} r, \frac{1}{N}\sum_{(r,c)\in R} c\right)$$

where $(r,c) \in R$ represents the row-column coordinates of a pixel in R, and N is the area of R;

4) *Minimum bounding rectangle* (MBR), whose upper left vertex point, (r_1,c_1) and lower right vertex point, (r_2,c_2) are defined by

$$(r_1, c_1) = \left(\min_{(r,c)\in R} r, \min_{(r,c)\in R} c\right)$$

and

$$(r_2, c_2) = \left(\max_{(r,c)\in R} r, \max_{(r,c)\in R} c\right);$$

5) *Elongatedness*,

$$ELONG = e_1/e_2$$

where e_1 and e_2 are eigenvalues of the covariance matrix of the distribution of $(r-\bar{r}, c-\bar{c})$ and $e_1 > e_2$;

6) *Perimeter*, the length of the boundary of R;

7) *Compactness*,

$$C = \frac{(perimeter)^2}{4\pi(area)};$$

8) *Touch_Background*, indicating what portion of R's perimeter is touching background;

9) *Touch_Image_Boundary*, indicating what portion of R's perimeter is touching image frame boundary.

The center of mass of a region gives a rough estimate for the location of the region. There are four kinds of basic shape features : MBR, *Elongatedness*, *Perimeter*, and *Compactness*. The MBR of a region is used to locate the region in the original image or the segmented symbolic image. The MBR also provides a rough estimate for elongatedness of a region. *Elongatedness* of a region is necessary to differentiate "long" defects from other defects. *Compactness* of a region is useful for finding compact and round defects holes. *Perimeter* of a region is used to calculate *Compactness* of the region. *Touch_Background* of a region is used to determine how much the region is touching background. *Touch_Image_Boundary* of a region is used to determine how much the region is touching image frame boundary. *Touch_Background* and *Touch_Image_Boundary* are helpful because a defect can be significantly deformed when it appears on the edge of a material or the image frame.

Chapter 6

THE HIGH-LEVEL MODULE

The high-level module's function is to perform the scene analysis task. It analyzes each of the regions forwarded to it by the low-level module. Its goal is to identify the type of defect present in each of these regions and to extract the appropriate characteristics associated with each defect. In this operation, top-down processing, using special operators applied to the original image, occurs when more evidence is needed to support a working hypothesis. The basic strategy is to first identify defects of each defect type using independently operating defect detection procedures, and then to disambiguate those instances where regions have been assigned multiple labels by using spatial contextual information or constraints among adjacent defects.

6.1 Initialization

The first KS to be applied in the high-level module is INIT (initialize), a KS for initialization. All region pointer vectors and interpretation labels are initialized as null, that is, all regions are unlabeled. The largest region is labeled as normal surface since defect area is very small in relation to the area of normal surface. Next a region adjacency matrix (RAM) is constructed and is stored in the Blackboard for later use.

6.2 Confidence Vectors : A Focus of Attention Mechanism

For each connected region passed to the recognition module by the segmentation module, a confidence vector is determined by KS CALCV (calculate confidence

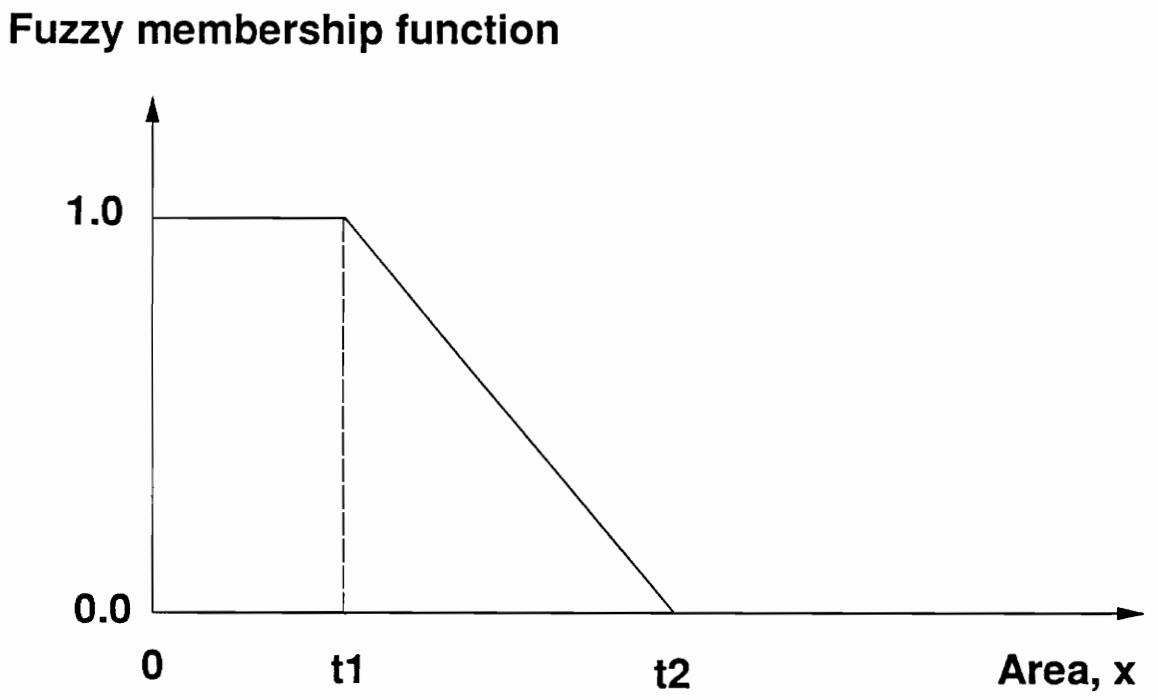


Figure 6.1. A fuzzy membership function for "small".

vectors). The evaluation of each component of a region's confidence vector is based on the region property (or attribute) vector computed by the segmentation module. The *i*th component of a region's confidence vector is the confidence that the region is a defect of type i. These confidence vectors are used as a "focus of attention" mechanism. A particular defect detection procedure will be applied to a region only if the confidence vector indicates that there is some reasonable confidence that the region is the type of defect the defect detection procedure has been taught to recognize. The evaluation of these confidences are based on fuzzy set theory [ZAD65].

To use fuzzy set theory to assign confidences, the characteristics of each defect must first be described using a predefined set of linguistic qualifiers and operators. Then each qualifier must be assigned a fuzzy membership function that is a function of one or more region properties. Then, based on these membership functions, the elements of the confidence vector of each region are computed using intersection (for logical ANDing), union (for logical ORing), and complement operation (for logical NOT) on fuzzy sets. These operations are typically defined as

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

where $\mu_{A\cap B}(.)$, $\mu_{A\cup B}(.)$, and $\mu_{\neg A}(.)$ represent intersection, union, and complement operation, respectively, and $\mu_A(.)$ denotes a fuzzy membership function for a set A.

For example, assume that a defect type, D is given a description, "small" and "dark". Also assume that a fuzzy membership function for "small" is defined, as shown in Figure 6.1, by a piecewise linear function

$$\mu_{small}(x) = 1, \qquad x < t_1,$$

$$= \frac{x - t_2}{t_1 - t_2}, \quad t_1 \le x \le t_2,$$

$$= 0, \qquad x > t_2$$

where x is the area of a region, and where t_1, t_2 are parameters determined empirically. Similarly, a fuzzy membership function for "dark" $\mu_{dark}(y)$ can be defined, where y is the average gray-level of the region. Let *cf(R,D)* be the confidence that region R is a defect type D. Then, *cf(R,D)* is calculated using

$$cf(R,D) = \min(\mu_{small}(x), \mu_{dark}(y)),$$

where x is the area of region R and y is the average gray level of region R.

The reasons for using fuzzy logic in computing confidence vectors are twofold. First, it provides a convenient mechanism for "roughly" representing the concept of an object using terms of natural language. A similar view was presented in [SUB86]:

> "... the true advantage of the fuzzy set theoretic approach is that it can incorporate a linguistic framework that expresses relations among the concepts in the domain and provides a scheme for defining terms that do not have precise quantitative formulation."

Typically each defect type can be described using imprecise linguistic terms, even if it is impossible to accurately characterize the defect type. As a focus of attention mechanism, the purpose of region R's confidence vector is only to approximately estimate the possibility of R being each defect type. Hence, fine-tuning of membership functions for the linguistic terms is not necessary.

Second, it is also computationally very simple because operations performed are minimum, maximum, and subtraction. If n fuzzy membership functions are involved in computing a confidence value, the time complexity is $O(n)$.

6.3 Defect Detection Procedures

Once all the regions have had their initial confidence vectors computed, defect detection procedures are applied. These defect detection procedures are very specialized KSs. Each defect detection procedure is designed to detect a particular defect using knowledge about spectral and shape characteristics of that defect. Each procedure runs independently of the others: The order of applying the procedures is unimportant. A region's confidence vector determines which defect detection procedures will be applied to that region. Each defect detection procedure is only applied to regions in the image whose confidence vector indicates it might be the corresponding defect type.

6.3.1 Initial Labeling

Basically, each defect detection procedure performs two steps. The first step in each defect detection procedure is *an initial labeling step*. In DF*i*, a defect detection procedure for detecting defect type *i*, this step assigns defect label *i* to a region if the region seems to be defect type *i*.

The basic strategy used in this step is to first detect regions that represent "exemplary" examples of the defect the procedure was designed to recognize. An "exemplary" region is defined as one that can be recognized alone without using context, i.e., its relationship with adjacent regions. Specialized defect detection methods using model-driven or top-down processing are applied to detect such "exemplary" defects. To detect such "exemplary" defects in DF*i*, operators are applied that are specialized

to detect defect type *i* using a priori knowledge about defect type *i*. These special operators are applied to only regions having some confidence of being defect type *i*. Only regions substantiated by the results of these special operators are labeled as defect type *i*.

After all of the exemplary defects have been recognized, then regions representing more "ambiguous" examples of that defect type are considered. An "ambiguous" region is defined as one that cannot be recognized alone, but can only be recognized using its context with its adjacent "exemplary" regions. Labeling these regions is based on defect detection methods that gauge contextual dependency and similarity of region properties. Similar strategies have been shown effective for the scene analysis task in knowledge-based vision systems for outdoor scenes and aerial images [NAG80, RIS84]. The initial labeling step is implemented as rules that encode a priori knowledge about the defect type being considered using region properties, confidence vectors, and special operators when needed. As the labeling is being performed links between adjacent regions with the same label are being formed. Such linked regions form the next level in the descriptive hierarchy, DEFECT_OBJECT.

6.3.2 Label Verification

The next step of a defect detection procedure is *a label verification step*. It is necessitated by the fact that a segmented region in an image does not usually correspond to a complete defect, i.e., typically a defect is fragmented into several regions during segmentation. At this stage of the processing a basic property list is computed for each DEFECT_OBJECT. The basic properties at the level of DEFECT_OBJECT should be more representative of the actual defect than those computed from the individual regions that comprise a DEFECT_OBJECT. This basic property list is used together with other knowledge about the defect to verify

the labeling of the DEFECT_OBJECT. Three approaches for DEFECT_OBJECT verification have been implemented and tested. These include a rule-based approach, a *k*-nearest neighbor approach, and a neural network approach.

Each of these three methods has a group of that advocates its use. As of this writing there is not general consensus as to which is the best approach. Each does seem to have its own unique characteristics, characteristics that may make it more appropriate for use on a class of problems than the other two. As such all three will be described.

Because of the ambiguity as to which method might be more appropriate for the label verification, a test was conducted to compare the three methods. This test is described in Chapter 7. The results of this test suggest that the neural network approach represents the best choice for this application. There are other reasons for selecting neural networks as well. These will be described below.

6.3.2.1 A Rule-Based Approach

In the rule-based approach, a series of tests are applied to each DEFECT_OBJECT to be verified. Each test generates confidence values that the DEFECT_OBJECT is / is not the defect type based on the test's own criterion. The confidence value of each test is determined by its own mapping function that typically uses some parameters or thresholds. Each defect detection procedure has its own series of tests. Depending on the type of defect label being verified, a test might involve computing a new feature (e.g., a texture measure) from the original image. Confidence values generated by those tests are combined to yield a final confidence or belief value for the defect type using Dempster's rule of combination. Only if this confidence value is greater than 0.5, is the DEFECT_OBJECT's assigned label

verified. (The Dempster's rule of combination was selected over other uncertainty management methods for the same reasons as given in the region merging operation in Section 5.5.)

6.3.2.2 A K-Nearest Neighbor Approach

The *k*-nearest neighbor classifier is a conventional nonparametric classifier that provides good performance for optimal values of *k*. In the *k*-nearest neighbor rule, a test sample is assigned the class most frequently represented among the *k* nearest training samples. If two or more such classes exist, then the test sample is assigned the class with minimum average distance to it. It can be shown that the *k*-nearest neighbor rule becomes the Bayes optimal decision rule as *k* goes to infinity [DUD73]. However, it is only in the limit as the number of training samples goes to infinity that the nearly optimal behavior of the k-nearest neighbor rule is assured.

There are at least two ways one can use the *k*-nearest neighbor classifier to perform the label verification operation. One way is to use a single *k*-nearest neighbor classifier to classify all defect types. If there are *N* defect types to be recognized, the *k*-nearest neighbor classifier must be able to handle *N+1* classes (one class for normal surface). If this is the method employed, each defect detection procedure will use this (*N+1*)-class *k*-nearest neighbor classifier to verify the labels this defect detection procedure has assigned. Hence the set of features used to do the verification is fixed and each defect detection procedure must extract these features from each DEFECT_OBJECT it creates. A defect detection procedure for recognizing defect type *i* will verify the label it has assigned to a DEFECT_OBJECT only if the *k*-nearest neighbor classifier classifies this DEFECT_OBJECT as being of defect type *i*.

Another way the k-nearest neighbor classifier can be used to do label verification is to use a number of two-class *k*-nearest neighbor classifiers. If *N* defect detection procedures are used to recognize *N* defect types, then there would be *N* two-class *k*-nearest neighbor classifiers employed. Each defect detection procedure would have its own specially tailored classifier. If a defect detection procedure is designed to recognize defect type *i*, its *k*-nearest classifier would attempt to determine whether a DEFECT_OBJECT is really of defect type *i* or not really of defect type *i*. Having such a special purpose *k*-nearest neighbor classifier inside each defect detection procedure has a number of advantages. Chief among these is that each defect detection procedure can use its own special purpose features without having to use the same feature set as all the other defect detection procedures. It is conjectured that this "pairwise" classifier should yield better results than when only one (*N+1*)-class *k*-nearest neighbor classifier is used.

6.3.2.3 A Neural Network Approach

In this approach, a multilayer feedforward artificial neural network is used as the classification method instead of a *k*-nearest neighbor classifier. As in the *k*-nearest neighbor classifier there are at least two ways neural networks can be used to verify labels. One way is to use only one network to make all the verifications. As before, this approach requires all the defect detection procedures to extract the same set of features. It also means that if there are *N* defect types to be recognized the neural network must be able to handle *N+1* classes (one class for normal surface). The other approach is to use *N* two-class neural network classifiers. The advantages of this approach are the same as those described above for the set of two-class *k*-nearest neighbor classifiers. It is believed that the use of *N* two-class neural

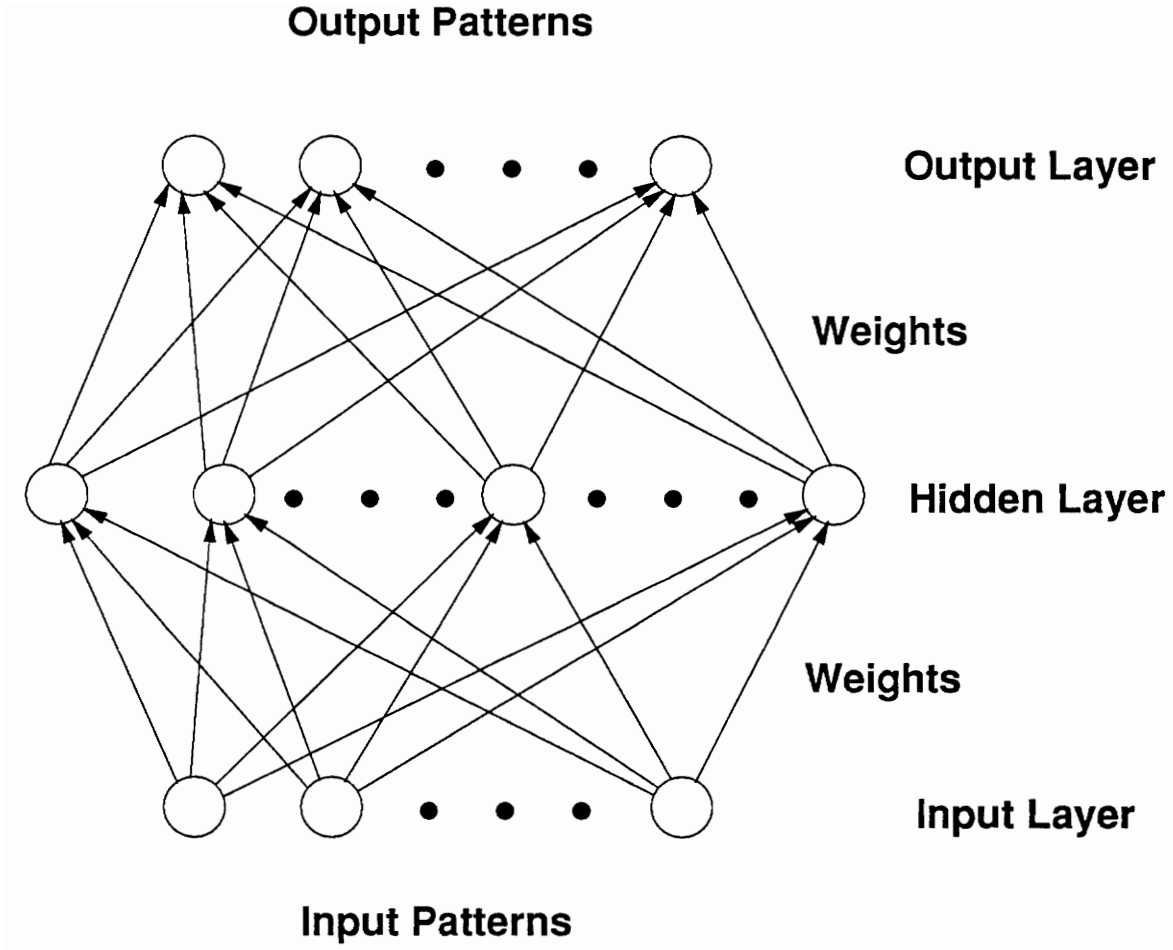


Figure 6.2. A 3-layer feedforward artificial neural network.

network classifiers will provide better accuracy than the use of one $(N+1)$-class neural network classifier, and as such is the recommended approach for using neural networks.

Since the readers may not be familiar with multilayer feedforward neural networks, some background information on these networks will be provided. Then use of the neural network for the label verification will be described in detail.

A *multilayer feedforward artificial neural network* [RUM86] provides a methodology for determining discriminant functions for classification via supervised learning. This network has one *output layer* and one *input layer* with one or more *hidden layers*. Each layer has a set of *units*, *nodes*, or *neurons*. It is usually assumed that each layer is fully connected with an adjacent layer without direct connections between layers which are not consecutive. Each connection has a *weight*. Figure 6.2 shows a typical 3-layer feedforward network with one hidden layer.

The input of each unit in a layer (except input layer) is given by

$$net_{pi} = \sum_j w_{ij} a_{pj}$$

where net_{pi} is the net input to unit i produced by the presentation of pattern p, w_{ij} is the weight from unit j to unit i, and a_{pj} is the output value of unit j. The output of unit i is specified by

$$a_{pi} = f(net_{pi})$$

where f is a *semilinear activation function* which is differentiable and nondecreasing.

The learning of multilayer neural networks is mostly accomplished using the well-known *back-propagation (BP)* learning algorithm [RUM86]. For details of the derivation of the rule, see [RUM86] or Appendix B of this dissertation. The BP

algorithm tries to find a set of weights that minimizes an overall measure E. E is the sum of an error measure on pattern p, E_p, which is defined by

$$E_p = \frac{1}{2}\sum_i (t_{pi} - o_{pi})^2,$$

where t_{pi} is the *target value (desired output)* of output unit i for pattern p, o_{pi} is the actual output of output unit i produced by presenting input pattern p, and i indexes over the output units.

The weight update rule typically used in the BP algorithm [RUM86] is given by

$$\Delta_p w_{ij}(n+1) = \eta(\delta_{pi} a_{pj}) + \alpha \Delta_p w_{ij}(n)$$

where $\Delta_p w_{ij}$ is the change to be made to w_{ij} following presentation of pattern p, n represents the nth presentation of the training set, η is called *the learning rate*, and α is called the *momentum term* that determines the effect of past weight changes on the current weight changes. When unit i is an output unit, then δ_{pi} in the above equation is given by

$$\delta_{pi} = (t_{pi} - o_{pi}) f'(net_{pi})$$

and when unit i is a hidden unit, then δ_{pi} is given by

$$\delta_{pi} = f'(net_{pi}) \sum_k \delta_{pk} w_{ki}.$$

Typically, the semilinear function f is given by the *logistic function* $f(x)$, where,

$$f(x) = \frac{1}{(1 + e^{-x})}.$$

In this case,

$$f'(net_{pi}) = a_{pi}(1 - a_{pi}).$$

Hence for an output unit i,

$$\delta_{pi} = (t_{pi} - o_{pi}) o_{pi}(1 - o_{pi}),$$

and for a hidden unit i,

$$\delta_{pi} = a_{pi}(1 - a_{pi}) \sum_k \delta_{pk} w_{ki}.$$

Recently, it was shown that the multilayer perceptron, trained using BP learning algorithm, approximates the optimal discriminant function defined by Bayesian theory [RUC90, WAN90, BOU90]. Specifically, the outputs of the multilayer perceptron approximates a posteriori probability functions of the classes being trained. How closely the multilayer perceptron approximates a posteriori probabilities depends on the architecture of the network and the functional form of the underlying probability density function. It was also shown that a multilayer perceptron with more than one hidden layer has the capability of approximating any continuous mapping to any desired degree of accuracy, if a sufficient number of hidden neurons are provided [HOR89, FUN89, CYB89]. These results suggest that the asymptotic behavior of the neural networks should be the same as that of the k-nearest neighbor method [DUD73].

Among multilayer perceptrons with hidden layers, a 3-layer perceptron with one hidden layer is the least complex. Further, the more hidden layers are used, the more difficult it is to choose an appropriate number of nodes for each hidden layer. Therefore, a 3-layer feedforward artificial neural network has been chosen as a classifier for use in the label verification step employed in each defect detection procedure. Note that 2-layer neural networks without a hidden layer were excluded from the consideration because it is well known [LIP87] that they can form only linear decision boundaries, not non-linear decision boundaries. The number of hidden neurons is determined experimentally. Enough hidden neurons must be provided so that successful learning can be obtained from the training set.

Weights in the network are obtained by using a training set. The training set consists of feature vectors computed from DEFECT_OBJECTs representing normal surface and the various defect classes. If a training sample belongs to a class i, then its *desired output* or *target value* is 1 for output node i, and 0 for all other output nodes. During training, weights are updated after one training sample is presented. This process is repeated until for each training sample, the difference between every output value and its target value is less than 0.1. (Actual target values used are 0.1 and 0.9 instead of 0 and 1, respectively, to prevent over-learning.) After training is completed, an n-class neural network classifier assigns class i to an input feature vector if output node i of the neural network is the highest of all the n output node values.

In practice, incremental learning is probably the best way to perform the training phase of the neural network. An incremental learning method can be performed by first constructing an initial training set that consists of typical exemplars for each defect type. This initial training set is then used to train the neural network. Whenever a test sample is misclassified using this trained network, this sample is added to the training set and the neural network is re-trained using the new training set. This re-training process is repeated until an acceptable classification performance is achieved.

The above neural network based decision rule for classifying a DEFECT_OBJECT is an approximation of Bayesian minimum-error-rate classification with zero-one loss function and without a reject option. In general, a multilayer neural network can be used to approximate Bayesian minimum risk classification with reject option. Assume that one has the option either to assign an input pattern to one of n classes, or to *reject* it as being unrecognizable. If the cost for rejects is not too high,

rejection may be a desirable action. Let $\{\omega_1, \ldots, \omega_n\}$ be the finite set of classes and $\{\alpha_1, \ldots, \alpha_{n+1}\}$ be the finite set of $n+1$ possible actions, where α_i represents the decision that the true class is α_i for $i = 1,...,n$ and α_{n+1} represents rejection. Let $\lambda(\alpha_i \mid \omega_j)$ be the loss incurred for taking action α_i when the true class is α_j. Let the feature vector $\mathbf{x}$ be a d-component vector-valued random variable and $P(\omega_j \mid \mathbf{x})$ be a posteriori probability.

Then the *Bayes decision rule* states that given a particular $\mathbf{x}$, to minimize the overall risk, compute the *conditional risk* associated with taking action α_i

$$R(\alpha_i \mid \mathbf{x}) = \sum_{j=1}^{n} \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid \mathbf{x})$$

for $i = 1,...,n+1$ and select the action α_i for which $R(\alpha_i \mid \mathbf{x})$ is minimum.

$P(\omega_j \mid \mathbf{x})$ can be estimated from the value of output node j on input $\mathbf{x}$ of the neural network because the value of output node j is a biased estimator of a posteriori probability for class j [WAN90]. That is, $P(\omega_j \mid \mathbf{x})$ can be given by the *normalized* value of output node j of the neural network. This normalization is performed so that the sum of nonzero output node values should be 1. Determination of the conditional risk $\lambda(\alpha_i \mid \omega_j)$ is problem-dependent. Once the a posteriori probabilities and the conditional risks have been obtained, each $\mathbf{x}$ can be classified as a class for which the total risk is minimum.

This neural network is trained based on a training sample set using an improved back-propagation (BP) learning algorithm with fast learning speed and stable converging property. This improved algorithm will be described later in detail and compared with the standard BP learning algorithm using several benchmark problems.

6.3.2.4 Comparison of the Three Approaches

An advantage of the rule-based approach is that one can subjectively set parameters or thresholds used in each test to map the feature value obtained from the test into a confidence value, and get moderate performance even if no training data is available. Obviously, it is very desirable to incorporate information contained in training set by using it to fine-tune parameter values. If a small amount of training data is available, one can use it to tune the subjectively established values. However, if the mount of training data available is large, the ability to fine-tune parameter values can be a complex problem if one attempts to do this subjectively. This requires a significant amount of effort in knowledge engineering, if not performed automatically. Automatic setting of parameter values would be possible but not easy. Hence, the applicability of the method to industrial inspection is in doubt.

If a large training set is available, then k-nearest neighbor and neural network classifiers have a number of advantages over the rule-based classification method. Note that generating a large number of training samples is not difficult in most industrial inspection problems. These advantages include: 1) Both are robust and can outperform conventional parametric classifiers when the actual distribution of data is different from the assumed distribution [HUA87]. 2) The parameters or weights of the neural network are determined automatically by a learning algorithm based on a proper training set. The k-nearest neighbor classifier also establishes the decision boundary automatically based on a training set. 3) Both allow incremental learning; that is, its classification performance can be incrementally refined when new training samples are added to the existing training samples.

However, there are disadvantages in using these two approaches as well. The *k*-nearest neighbor classifier has a disadvantage in that it is difficult to find an

optimal value of k that produces the best performance for a training set with a finite number of training samples. On the other hand, the training of a neural network typically requires a large number of presentations of the training set. Another difficulty in using neural networks is selection of the proper number of hidden neurons. The number of hidden neurons must be carefully selected by experimentation. If too few hidden neurons are used, proper training is impeded. If too many are used, there is a performance degradation in generalization of the network, i.e., the network will perform poorly on unknown samples, even if it works perfectly on the training samples. Note that these disadvantages for both approaches really only represent computational problems in the training phase.

The reason the neural network classifier was chosen over k-nearest neighbor classifier is the speed of classification. The k-nearest neighbor classifier is too time-consuming. To classify a test sample, the k-nearest neighbor classifier requires that the distances between the test sample and each stored training sample be computed. This computation is proportional to the number of the training samples. This problem might be overcome by using the nearest neighbor rule on an "edited" and "condensed" set of an original training set [DEV82]. However, it is still not certain how much the training set can be reduced without performance degradation. On the other hand, the computational complexity of the neural network classifier is proportional to the number of weights. Hence, the neural network classifier has an advantage in the computational complexity and storage complexity over the k-nearest neighbor classifier since a large number of training samples should be available for these classifiers. Furthermore, it allows parallel hardware implementation, which can considerably speed up classification.

It was the above consideration together with the results of the test reported in Chapter 7 that has led to the selection of neural networks as the preferred method for doing the label verification. One unfortunate aspect of this selection is the rather slow learning rate typically associated with the BP learning algorithm. In an attempt to address this training phase problem, efforts were extended to develop a method that would increase learning speed without affecting stability. The results of these efforts will be reported in Section 6.7.

6.4 Labeling Regions by Context

Even after all the defect detection procedures have been applied, there may still exist unlabeled regions. Such regions are labeled using spatial contextual dependency by the KS LBLCON (label by context). First, every unlabeled region is labeled as normal surface. This step is motivated by the fact that the defect detection procedures should be able to identify any region that truly is a defect. Based on this assumption any unlabeled region should be a normal surface. In reality, however, some unlabeled regions are really a part of a defect. To allow for this possibility after LBLCON has finished step 1, it checks to see if each "small" normal surface region is almost "enclosed" by any DEFECT_OBJECT. (The extent of "enclosure" is determined from the region adjacency matrix.) If this is the case, then this region is given the label of the DEFECT_OBJECT with which it shares the longest common boundary. The word "small" is used above to indicate that the largest region, i.e., the region that corresponds to normal surface, is not subjected to this analysis. This prevents a defect around the boundary of the material from causing the interior normal surface area to be labeled as a defect. As normal surface regions are assigned new defect labels by LBLCON, applicable DEFECT_OBJECTs are updated

to reflect the new labeling. Any updated DEFECT_OBJECT has a flag set to indicate it has been updated. At this time no updated property list is generated for any of the DEFECT_OBJECTs.

6.5 Defect Verification Using Spatial Constraints

The KS VRFDFC (verify defects) attempts to further verify the labeling of each DEFECT_OBJECT by using spatial constraints among adjacent DEFECT_OBJECTs. This is necessary because each defect procedure encodes knowledge specific only to its defect type and does not contain contextual knowledge among the different defect types. Relational predicates that can describe spatial relationships such as ENCLOSED_BY and ADJACENT_TO, are used. If the validation procedure fails to verify the label assigned to a DEFECT_OBJECT, the DEFECT_OBJECT is removed from further considerations. The pointer vector of each region associated with this DEFECT_OBJECT has its component corresponding to the DEFECT_OBJECT's label re-initialized as null. (Note that the pointer vector of a region is used to identify a DEFECT_OBJECT that contains the region.) If after KS VRFDFC has been applied there are any unlabeled regions, KS LBLCON is used to label these regions using spatial contextual dependency.

6.6 Resolving Multiple Labels

The final step in the processing is to resolve instances where regions have been assigned multiple labels, i.e., where a region is a part of more than one DEFECT_OBJECT. Note that after the application of the independently operating defect procedures it is possible that a region might belong to more than one DEFECT_OBJECT. The KS RMLBL (resolve multiple labels) attempts to resolve any such instances so that each region has only one unique label. To find regions

that are part of more than one DEFECT_OBJECT and to identify the DEFECT_OBJECTs involved, the region pointer vectors are used. Each DEFECT_OBJECT corresponds to one possible label for the region. To determine which label the region should have, a new property list may be computed for each DEFECT_OBJECT involved if this is required. The determination as to whether a new property list needs to be computed is based on whether a flag has been set by KS LBLCON for the DEFECT_OBJECT under consideration. Each DEFECT_OBJECT's property list is used to calculate a confidence value that the DEFECT_OBJECT has its defect label. This confidence value is determined by the output value of the neural network corresponding to the defect type of the DEFECT_OBJECT. In the rule-based approach, the confidence value of the DEFECT_OBJECT is computed using Dempster's rule of combination. In the k-nearest neighbor approach, k_l/k can be used as the confidence value, where k_l is the number of nearest neighbors that is the defect type of the DEFECT_OBJECT among k-nearest neighbors. The region is assigned the label of a DEFECT_OBJECT whose confidence value is the highest among the confidence values of the DEFECT_OBJECTs associated with the region.

6.7 A Fast Back-Propagation Learning Algorithm

While all the three approaches have been tried in the label verification step of defect detection procedures, testing results reported in Chapter 7 have led to the selection of the neural network approach to the label verification. The neural network approach uses 3-layer feedforward neural networks that are the least complex multilayer neural networks. These multilayer neural networks are typically trained using the standard backpropagation (BP) algorithm. One major difficulty in using the standard BP algorithm is that training by the standard BP algorithm typically requires long learning

time, i.e., a large number of *epochs*. An *epoch* is one presentation of the whole training set to the neural network. Hence, faster learning algorithms are needed. There have been a number of BP speed-up methods described in the literature, e.g., [FAH88, JAC88].

In this section, a new fast BP learning algorithm that uses a steep activation function is introduced. It will be shown later that the BP learning algorithm using a steeper activation function converges to a solution faster. However, as an activation function becomes steeper, learning becomes more unstable, i.e., convergence failures occur more often. To overcome this instability, the proposed algorithm detects and reinitializes weights when convergence speed becomes "very slow" due to a *local minimum.*

The proposed BP learning algorithm uses an activation function defined by

$$f(x) = \frac{1}{(1+e^{-gx})},$$

where g is a constant called *gain.* Note that this activation function becomes the usual logistic function if $g = 1$. If this activation is used, only updating formulas for δ_{pi} are changed while others are the same as the above. Since

$$f'(net_{pi}) = a_{pi}(1-a_{pi})g,$$

δ_{pi} for an output unit i given by

$$\delta_{pi} = (t_{pi} - o_{pi})o_{pi}(1-o_{pi})g$$

and δ_{pi} for a hidden unit i is given by

$$\delta_{pi} = a_{pi}(1-a_{pi})g \sum_k \delta_{pk} w_{ki}.$$

In the BP learning algorithm, the initial weights are usually randomly chosen. These initial weights may significantly affect the convergence speed of the learning algorithm.

Some initial weights lead to very slow convergence to a solution or, in the worst case, to divergence. It will be shown later that the BP algorithm using the modified activation function with $g>1$ converges to a solution much faster than when the standard logistic activation function is used. But the modified function with $g>1$ diverges, i.e., does not converge to a solution more often than the standard logistic activation function.

To help avoid divergence the proposed algorithm randomly reinitializes the weights whenever convergence becomes slow because of a local minimum. To determine when the convergence to solution is slow, the sum of the error, E, is recorded after some number of *epochs*, 50 epochs in the current implementation. If the difference between two consecutive E's is less than a preselected threshold, then the convergence is considered slow and the weights are randomly reinitialized.

The performance of the proposed algorithm was compared with the standard BP algorithm using a standard set of diverse benchmark problems : exclusive-or (XOR), encoder, and parity problems [RUM86]. For the XOR problem, a 2-2-1 network was used. As a notational convention, a *K-L-M* neural network denotes a 3-layer neural network with K input nodes, L hidden nodes, and M output nodes. Two encoder problems, 4-4 encoder and 8-8 encoder, were used in this experiment. The function of the *N-N* encoder is to map N orthogonal input patterns into N orthogonal output patterns. In this experiment, both encoders implemented the identity mapping. The *K-L-M* structure used to create the two encoders is $N\text{-}log_2N\text{-}N$ where in one instance N=4 and on the other N=8. In the N-bit parity problem, the output value should be 1 if the input pattern contains an odd number of 1's and 0 otherwise. This is a very difficult problem because the most similar patterns (those which differ

by a single bit) require a very different response. An *N-N-1* architecture was used for the *N*-bit parity problem. Experiments were performed on 3-bit, 4-bit, and 5-bit parity problems.

During learning, weights are updated after one pattern is presented. This process is repeated until for each pattern, the difference between every output value and its target value is less than 0.1. (Actual target values used are 0.1 and 0.9 instead of 0 and 1, respectively.) Since the initial values of weights affect the convergence of a learning algorithm, it is reasonable to judge each algorithm by the statistics obtained from multiple runs. In our experiment, 20 sets of different initial weights were randomly generated. These weights were used to run each of the two algorithm on the XOR, encoder, and parity problems. The gain parameter g on the proposed algorithm was also varied on each problem. The learning time was measured by the average and standard deviation of the number of epochs required to reach a solution. If the solution was not reached in 5000 epochs, divergence was assumed. Also, a *failure rate* of the algorithm was measured by the ratio of the number of *unsuccessful* runs to the number of total runs. An unsuccessful run is a run in which a solution is not reached in less than 5000 epochs. In all these experiments, the learning rate (or weight step size) and the momentum term were set to 0.1 and 0.9, respectively.

Tables 6.1 - 6.5 show the effects of gain g on the performance of the BP learning algorithm. It is noted that the BP learning algorithm using the activation function with steeper slant (i.e., larger values of g) generally converges faster to a solution. However, it is also notable that more failures in convergence to solution occurred when the steeper activation function was used. Table 6.6 shows a comparison of

the proposed BP algorithm (FBP, g=2), i.e., the one employing automatic reinitializations with the standard BP algorithm (SBP, g=1). This table shows that automatic reinitializations of weights in the proposed algorithm greatly improves the success rate in convergence to solution while still maintaining much faster speed of learning.

TABLE 6.1

XOR

		Gain			
		1	2	3	4
Epochs executed	Mean	859	164	227	94
	Standard Dev.	322	68	348	76
Failure rate (%)		10	30	20	35

TABLE 6.2

Encoder 4-4

		Gain			
		1	2	3	4
Epochs executed	Mean	360	94	57	108
	Standard Dev.	83	23	24	208
Failure rate (%)		0	0	0	5

TABLE 6.3

Encoder 8-8

		Gain			
		1	2	3	4
Epochs executed	Mean	826	208	318	1151
	Standard Dev.	330	53	240	761
Failure rate (%)		0	0	0	30

TABLE 6.4

3-bit Parity

		Gain			
		1	2	3	4
Epochs	Mean	1132	348	386	392
executed	Standard Dev.	789	435	507	1088
Failure rate (%)		0	0	0	45

TABLE 6.5

4-bit Parity

		Gain			
		1	2	3	4
Epochs executed	Mean	1613	1669	841	386
	Standard Dev.	854	1630	819	83
Failure rate (%)		60	30	65	90

TABLE 6.6

Comparison of SBP and FBP

	Epochs Executed		Failure Rate
	Mean	Standard Dev.	(%)
		XOR	
SBP	859	322	10
FBP	253	187	0
		Encoder 4-4	
SBP	360	83	0
FBP	94	23	0
		Encoder 8-8	
SBP	826	330	0
FBP	208	53	0
		3-bit Parity	
SBP	1132	789	0
FBP	272	244	0
		4-bit Parity	
SBP	1613	854	60
FBP	931	620	5
		5-bit Parity	
SBP	3888	786	90
FBP	1837	1135	15

Chapter 7

A MACHINE VISION SYSTEM FOR AUTOMATED LUMBER GRADING

On the basis of the proposed vision system framework, a computer vision system for automated lumber grading has been developed. The purpose of the vision system for automated lumber grading is to locate and identify grading defects on rough hardwood lumber in a species independent manner. This problem seems to represent one of the more difficult and complex web inspection problems. No vision system has been developed to tackle this problem. The main reason for this seems to be that there are a lot of factors causing surface discolorations. Further, some defect types are difficult to recognize due to irregular characteristics of their shapes. Hence if one can use the previously described framework to develop a vision system for grading rough lumber, this framework should be versatile enough to work on a variety of other web inspection problems. In fact, the methodologies used in the vision system for grading would seem applicable to a number of other web inspection problems as well.

In what follows, a brief background will be presented about the lumber grading problem. Then a detailed description of the vision system designed for this problem is described. Finally, experimental results are presented to demonstrate the capability of this vision system.

7.1 Automatic Lumber Grading

7.1.1 Motivations

Sawmills saw logs into lumber. They sell this lumber to secondary remanufacturers who use this raw material to manufacture goods such as furniture, cabinets, etc. The price a sawmiller can charge for a volume of lumber depends on its grade [NAT86], e.g., Firsts and Seconds, No.1 common, No.2 common, etc. For a number of species the price of a given volume of material can double in going from one grade to the next higher grade. Thus, accurate and consistent grading of hardwood lumber is very important to both seller and purchaser of lumber.

Currently, the grading of hardwood lumber is done by inspectors according to standardized grading rules developed by the National Hardwood Lumber Association (NHLA). Skilled inspectors can accurately grade lumber. However, it takes considerable time to develop truly skilled graders and time translates into money. While few studies have been conducted to determine factors that can affect grading accuracy, it can be assumed that these factors correspond to those of other human endeavors. Therefore, it can be assumed that boredom, state of mind, fatigue, etc. can all affect the accuracy of the grading process.

An automatic grading system suffers none of those human failings. It never gets tired or bored. It attacks each new board with equal enthusiasm. An automatic system would be an expert at the very first minute it is turned on. In other words, it would be more efficient and consistent than human grading.

While the grade of a board does to a large extent depend on the distribution of defects on the board's surface, appropriate edging and trimming can increase the grade of a board. Hence optimal edging and trimming provide a mechanism for

a sawmiller to increase his profits. One preliminary study indicates that optimal edging and trimming could increase profits by at least 20% [KLI90]. Optimal edging and trimming would be possible if an automatic grading system existed. Thus, there are a number of reasons to believe that developing an automatic grading system could have a very positive impact on the sawmilling industry.

Any automatic grading system must have two components, as shown in Figure 7.1. The first of these is a computer vision system for locating and identifying grading defects on rough hardwood lumber. A grading defect is a surface defect that can affect a board's grade under NHLA rules. The surface defects that occur most frequently include knots, wane, holes, decay, stain (mineral and blue), checks, and splits. (Refer to Appendix C for definition of these defects.) Table 7.1 [CON83] gives an indication of the relative probability of occurrence of some of these defects in red oak for the grades of lumber typically used by the furniture and fixture industry. The second component required for an automatic grading system is a program that can establish the grade of a board based on the output of the vision system. It is this program that incorporates the NHLA grading rules and grades a board based on the distribution of defects present on the board's surface. Significant progress has already been made on creating the second of these components [KLI88]. The effort described here is aimed at developing the first component, the vision system for locating and identifying grading defects.

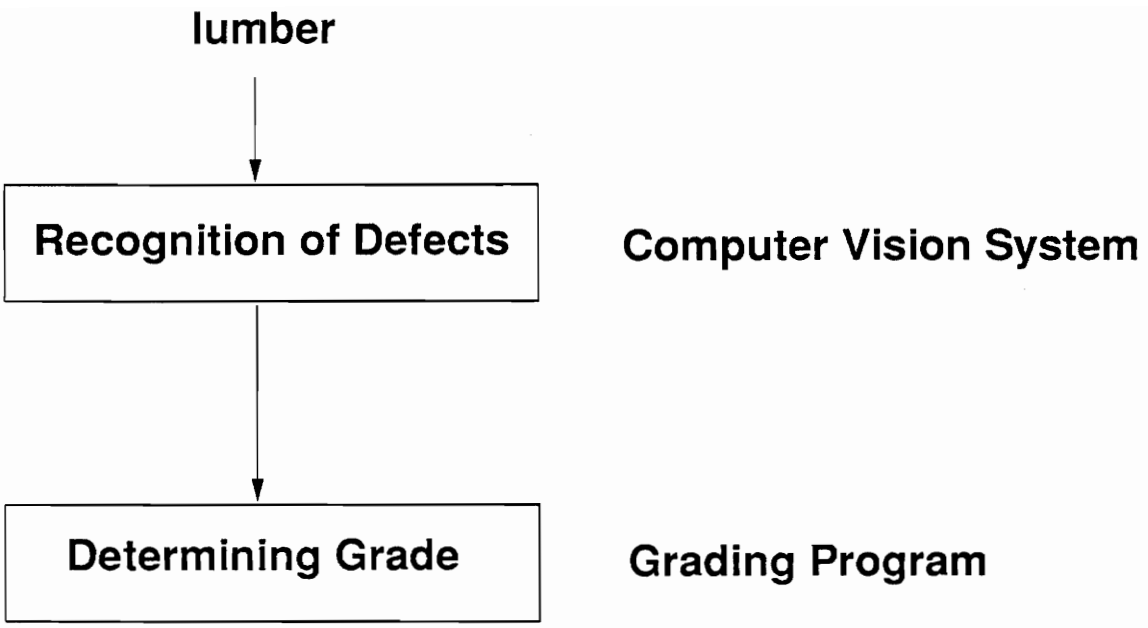


Figure 7.1. An automatic grading system

TABLE 7.1

AVERAGE PERCENTAGE OF TOTAL SURFACE AREA OCCUPIED BY MOST
COMMON DEFECTS IN THREE GRADES OF SOUTHERN RED OAK

DEFECT	No. 1 Common % Area	No. 2 Common % Area	No. 3 Common % Area
Knots	1.7	2.3	3.7
Splits/Checks	1.5	2.7	3.7
Wane	0.8	0.7	1.3
Bark Pockets	0.5	0.7	1.2
Stain	0.4	0.4	0.4
Mineral	0.2	0.6	0.7
Decay	0.2	0.4	1.5
Holes	0.04	0.07	0.17
Grub Holes	0.02	0.08	0.27
Worm Holes	0.01	0.07	0.17
Total % Area	5.5	8.8	13.0

7.1.2 Grading of Hardwood Lumber

Within the temperate zone a hardwood is a deciduous tree, a tree that drops its leaves each year. In contrast, a softwood is a coniferous or evergreen tree. As such, the terms softwood and hardwood do not imply anything about the material properties of wood. Some softwoods are harder than some hardwoods. Because of the differing inherent qualities, growth characteristics, and end uses of hardwoods as compared to softwoods, the grading methods applied to hardwoods are substantially different from those of softwoods. In general, softwood lumber is used to build structures. As such grading defects in a softwood are those that affect its strength. Hardwoods, on the other hand, are typically used to make furniture, cabinets, and other items where the primary purpose is to be both functional and esthetically pleasing. Hence hardwood lumber is graded based on its appearance. The grades are based on amount, distribution, and size of clear areas.

The hardwood grades have been adopted to establish the comparable value of lumber and to provide the user with a standard on which he may base his purchase depending on the particular end use. Lumber grades also provide purchasers a mechanism for estimating the volume of lumber required to produce a given number of clear parts. Obviously, the volume required depends on the dimensions of the parts to be cut.

Hardwood lumber is graded by inspectors. The grading is usually done on rough lumber, not on surfaced lumber. Inspectors grade each individual board and calculate its volume according to grading rules. The grade of a board is determined from its poor side, i.e., its worse face.

The seller, typically a sawmiller, employs inspectors to grade lumber to establish its selling price. To assure that the right grade is assigned, a volume of lumber

may be graded a number of times. The buyer, typically a secondary remanufacturer, employs inspectors to grade lumber to assure he got the grade of lumber he paid for. Obviously, to avoid disagreements between buyers and sellers, both groups must use the same criteria for establishing the grade, the value of lumber. The common criteria used are the NHLA grading rules.

The NHLA grading rules are the result of long and careful study by practical lumbermen, e.g., sawmillers, working in cooperation with the lumber buyers, e.g., secondary remanufacturers, with the aim of providing the best available products, conservation of timber from which lumber is cut, and maintaining a lumber language of terms and specifications. This lumber language permits a ready and understandable meeting of the minds among the buyers and sellers wherever and for whatever use hardwoods are required. As a consequence, the rules are used universally. The grading rules are of further advantage in that they apply generally to all species with certain exceptions.

The underlying codification of the rules has been and continues to be revised and/or enlarged by the NHLA to reflect the industry's needs. Revisions are based on the advice of a standing committee representing the various species and producing areas.

To provide protection of buyers and sellers of hardwoods, the NHLA maintains a staff of full-time salaried, highly qualified inspectors whose services are available upon request. These inspectors act as final arbitrators between a buyer and a seller over the quality/price of a given volume of material. Their duties are to grade the lumber and to provide, where applicable, a certificate covering their work. Such certificates are financially guaranteed by the NHLA as covered by the regulations governing their issuance.

7.1.3 Rough Lumber Grading Problem

There are a number of problems that are unique to the inspection of rough lumber. First, there is the problem of surface moisture. Theoretically, lumber can be graded right after it is first sawn to a time after it is kiln dried. The variation in surface moisture content during this period of time is substantial and is known to cause a significant variation in the visual appearance of the material [SUL67].

Next there is the variability in exposure to ultra-violet radiation. After lumber is sawn it is typically stacked, usually outside, where the stack is exposed to direct sunlight. Boards on the exterior portion of the stack receive a significant exposure to ultra-violet radiation while boards on the inside of the stack receive very little exposure to ultra-violet light. This difference in exposure can and does cause a marked variation in appearance depending on a board's location within the stack [SAN62]. Stacks stored outside are exposed to the weather. This weathering can also cause a variation in the visual appearance of a board, again depending on a board's location within a stack.

There is also the problem of boards getting dirty during the various materials handling operations that occur prior to grading. Obviously dirt can be mistaken for a grading defect. Most graders carry a grading stick to assure that a particular spot on a board is not just dirt that can be scraped off but a real grading defect that is present in the board. Even the drying process introduces potential color variation in the material. The stickers used to separate the boards during drying can leave marks on lumber. During drying, board surfaces can become badly discolored by water stain and dried sap.

The rough surface itself can cause problems. Surface roughness is attributed to both the fibrous nature of wood and the vibrations of saw blades. These vibrations produce the characteristic patterns on the surface of rough lumber. Due to this surface roughness, the lighting needed to digitize an image of a board can cast shadows. These shadows can be misinterpreted by a computer vision system. It is also known that the extent of surface roughness can affect the color of lumber [NAK60].

All of the above mentioned problems represent surface discolorations that can be removed by planing or surfacing lumber. Thus a vision system for inspecting surfaced lumber does not need to cope with any of these variations. Therefore developing a vision system for examining rough lumber is more difficult than developing a vision system for examining surfaced lumber.

7.1.4 Vision System Requirements

A computer vision system for automated hardwood lumber grading must meet certain design requirements.

1. The vision system must be able to locate and identify grading defects. Grading defects include knots, splits/checks, wane, hole, decay, stain, and pith. The output of the vision system must include an accurate boundary for each grading defect present and a label that identifies the type of defect present in each of the outlined areas.
2. The vision system must be able to perform the above task on boards coming from a variety of species. Hardwood species vary significantly in their appearance and in the way grading defects manifest themselves. The most frequently used hardwood species include red oak, cherry, hard maple, and yellow-poplar. These species are the species used in making most of the

wood household furniture made in the U.S.A. However, other important species include walnut, basswood, ash, hackberry, white oak, black gum, red gum, soft maple, beech, and birch.

3. The vision system must be able to process high spatial resolution image data since grading depends on detecting small grading defects. A spatial resolution of up to 64 points/inch resolution might be needed to detect small splits and checks. The longest hardwood board is approximately 16 feet 9 inches long, with the average board length being in the 10 feet to 12 feet range. The average width of a hardwood board is 7 inches, but boards can be two to three times this value. At 64 points/inch spatial resolution the amount of data that has to be processed is approximately 10 Mbytes.
4. Any automatic grading must be at least as fast as a skilled human grader. This means that the grading system must be able to process lumber at a rate of at least two linear feet per second, i.e., so it can grade a 16 foot board in 8 seconds. Hence, the computer vision system must be able to process a board image at a rate that will allow a total system performance of at least two linear feet per second.

7.1.5 Previous Work

While some investigators [CON83, CON89, KOI88, KOI89, PAU88] have reported progress on developing computer vision systems for locating and identifying defects on surfaced lumber, little if any work has been done on locating and identifying defects on rough lumber. The reason for this is that rough lumber processing is arguably more complex than the surfaced lumber problem, as described previously. Thus, this review is concentrated on vision systems for inspecting surfaced lumber.

Conners, et al. [CON83] first divided the image of a board into a number of disjoint rectangular regions and classified each independently using a pairwise Bayesian classifier. In the classification, both tonal measures and textural measures were used. There are some problems with this approach. First, it cannot find exact

boundaries of defects because each region to be classified is rectangular. Second, it is too time-consuming since the texture measures are calculated in every rectangular region.

In 1986, Koivo and Kim [KOI86] used a similar approach to that used in [CON83] except that a tree classifier was used instead of a Bayesian classifier. However, it still suffers from almost the same problems as those of [CON83].

In 1989, Koivo and Kim [KOI88, KOI89] used causal autoregressive (CAR) models to characterizes gray levels of the images of wood boards. The parameters of the model equations are used to form a feature vector that characterizes the board images. The feature vector is input to a hierarchical tree classifier. While accurate classification results were reported, it should be noted that the sample images that were used in estimating model parameters were used again to test the classifier. Also, it has the same disadvantages as those of [CON83].

Finally, Kim and Koivo [KIM90] used a hierarchical recognition procedure to improve the resolution of the defect detection and to reduce the computation time for classification. However, this approach also has a number of major weaknesses. First, it employes a histogram-based thresholding method using a single threshold value. Using a single threshold may cause different adjacent defects to be considered as a single defect. This is very undesirable for defect recognition. Next, the hierarchical classification procedure used is such that once this procedure has made a wrong decision at a node, the decision is final and cannot be corrected. Consequently, a wrong decision at a node affects all nodes below the current node. This can cause disastrous errors.

In 1988, Paul, et al. [PAU88] constructed a vision system for inspecting hardwood surfaces. Its processing steps are as follows:

1) detection of pixels originating from potential defects by filtering,
2) clustering detected and adjacent pixels into the regions,
3) computing geometrical and gray level region features,
4) merging regions that are spatially close to one another, and
5) classification of defects using a hierarchical decision tree.

Although their vision system runs fast due to hardware implementation of the first three processing steps, it has some disadvantages. First, some different kinds of defects can appear at close distance; in this case those defects would be merged into a single region by step 4), which is undesirable. Second, the processing control is purely bottom-up; classification of defects is based only on region features. It is often agreed that a top-down processing component should be used to attain robustness in any real world problem.

In summary, most of the vision systems reviewed above suffer serious problems inherent in statistical pattern classification. These problems are described in detail in [NAG80]. The most serious of all these problems is that they do not utilize all the available knowledge about the various properties of defects such as defect shape, defect size, defect location, syntactic and semantic relationships that exist among defects, etc.

7.2 Implementation of the Vision System

In Chapter 6, a description of the proposed vision system for use on the web inspection problems was presented. The purpose of this subsection is to show how this general system can be used on the difficult rough lumber inspection problem.

As a consequence, this section will only describe those aspects of the proposed vision system that are required to tailor the general framework to this particular problem.

7.2.1 The Segmentation Module

To reduce the computational complexity of the first segmentation step, i.e., differentiating pixels of board from pixels of background, the color chosen for the background is blue. This choice of background color is predicated on the fact that the color of wood always contains a significant red component but almost no blue component. By choosing a blue color for the background the 256x256 red/blue plane can be used to easily segment pixels of board from pixels of background by using a simple look up table. Details of this step are found in [CON89].

To do the histogram-based segmentation of pixels of clear wood from pixels that might be a defect, a Gaussian smoothing function with $\sigma = 3$ is used. The value $\sigma = 3$ was selected experimentally and has subsequently proven to be very satisfactory.

The region properties extracted by the segmentation module for use in this problem are area, average gray level, center of mass, minimum bounding rectangle (MBR), elongatedness, perimeter, compactness, touch_background, and touch_image_boundary. (See Section 5.4 for the definitions of these properties.) The elongatedness feature is used to differentiate splits/checks from other defects. The compactness feature is useful for finding compact defects, e.g., holes. The perimeter feature is used to calculate a region's compactness. The touch_background feature, indicating how much portion of R's perimeter is touching material boundary, is a helpful feature in finding wane since wane almost always appears on the edge of a board. The

touch_image_boundary, indicating how much portion of R's perimeter is touching image frame boundary, is useful because a defect region can have an abnormal shape when it is on the image boundary.

7.2.2 The Recognition Module

Currently, this module has been taught to recognize four of the most common defect types. These are knots, holes, wane, and splits/checks. Other defect types remaining in Table 7.1 are stain, decay, bark pockets, and mineral streaks. Among these defects, only stain and decay are grading defects that are not recognized in the current implementation. For the grading purpose, bark pockets can be considered as knots. Also, mineral streaks are not grading defects. The normal surface area without defects is called clear wood.

7.2.2.1 Computation of Confidence Vectors

To compute the confidence vectors using fuzzy logic, each defect type has to be described using a set of predefined qualifiers and operators (AND, OR, NOT). The descriptions used to describe each defect are as follows:

splits/checks -- dark, long, and not thick

holes

small holes -- small, dark, less elongated, and compact

large holes -- not small, very dark, and less elongated

wane -- large, dark, not very long, thick, and touching board boundary

knots -- not small, dark, not very dark, and not very long

A fuzzy membership function for each of these terms was created. The parameters used to define each of the membership functions are all fixed except those used to define the membership functions for "dark" and "very dark". These parameters are variable and are based on the average gray level of the normal surface region. The reason for making the parameters defining these membership functions variable is that the concept of darkness is relative and varies significantly from one hardwood species to another. Based on these membership functions the elements of the confidence vector of each region were computed using intersection, union, and complement operation on fuzzy sets.

7.2.2.2 Defect Detection Procedures

Basically, each defect detection procedure performs two steps. The first step in each defect detection procedure is *an initial labeling step*. The basic strategy used in this step is to first detect regions that represent "exemplary" examples of the defect the procedure was designed to recognize. An "exemplary" region is defined as one that can be recognized alone without using context of its adjacent regions. Specialized defect detection methods using model-driven or top-down processing are applied to detect such "exemplary" defects. After all of the exemplary defects have been recognized, then regions representing more "ambiguous" examples of that defect type are considered. An "ambiguous" region is defined as one that cannot be recognized alone, but can be recognized using context of its adjacent "exemplary" regions. Labeling these regions is based on defect detection methods that gauge contextual dependency and similarity of region properties.

The next step of a defect detection procedure is *a label verification step*. Three approaches for DEFECT_OBJECT verification have been implemented and tested.

These include a rule-based approach, a *k*-nearest neighbor approach, and a neural network approach. However, tests reported later have led to the selection of the neural network approach for use in the proposed vision system.

In the k-nearest neighbor and the neural network approach, a single 5-class classifier was used to classify all four defect types (one class for normal surface). A single 5-class classifier was chosen over the method using five 2-class classifiers due to much simpler training. 10 input features of the classifier are extracted for each DEFECT_OBJECT to be verified. Defect detection procedure for recognizing defect type *i* will verify the label it has assigned to a DEFECT_OBJECT only if the classifier classifies this DEFECT_OBJECT as being of defect type *i*. (In the neural network approach, a 10-10-5 neural network was used. The number of hidden neurons, 10, was determined experimentally. This was the minimum number that allowed successful learning on a training set.)

The 10 features extracted include area, average gray level, elongatedness, compactness, touch_background, touch_image_boundary, a local contrast measure, gray-level variance, absolute central moment, and average gradient magnitude. Among these features, a local contrast measure, gray-level variance, absolute central moment, and average gradient magnitude are new *additional* features used to gauge local contrast and texture characteristics. The others are the *basic* features as defined in Section 5.5. To define the new features, let *g(r,c)* represent the gray-level at pixel *(r,c)*. Also let *area(D)* and *av_gl(D)* denote the area and the average gray-level of *D*, respectively. The new features defined are:

- *Local contrast (LC)*,

$$LC = \left(\frac{1}{area(R')} \sum_{(r,c) \in R'} g(r,c) \right) - av\_gl(D).$$

where R' is the set of pixels that lie outside of D and within the focus window of D, which is slightly (e.g., 1.1 times) larger than the MBR of D.

- *Gray-level variance*, variance of all the gray levels of the pixels in a DEFECT_OBJECT.

- *Absolute central moment (ACM)*,

$$ACM = \frac{1}{area(D)} \sum_{(r,c) \in D} | g(r,c) - av\_gl(D) | .$$

- *Average gradient magnitude (AGM)*,

$$AGM = \frac{1}{area(D)} \sum_{(r,c) \in D} gradm(r,c)$$

where *gradm(r,c)* is the gradient magnitude of the Sobel edge operator [PAV77] at pixel *(r,c)*.

The remaining approach that was tried is the rule-based approach. The rules used in the defect detection procedure designed to identify defect type i attempt to determine whether a DEFECT_OBJECT created by this detection procedure is really a defect of type i or not really a defect of type i. As such the exact rules used vary from one defect detection procedure to another. In what follows, each of the four defect detection procedures that have been implemented will be described in detail. In the description of each, the rules used in the rule-based label verification step will be presented.

7.2.2.2.1 Split/Check Detection Procedure

This KS is applied to a region only if the region's confidence vector indicates it might be a split/check. Usually, splits/checks appear in an image as a linear structure that is elongated, thin, and dark. Figure 7.2 shows typical splits/checks. To detect regions representing "exemplary" splits/checks, a "crack" detector is applied to each

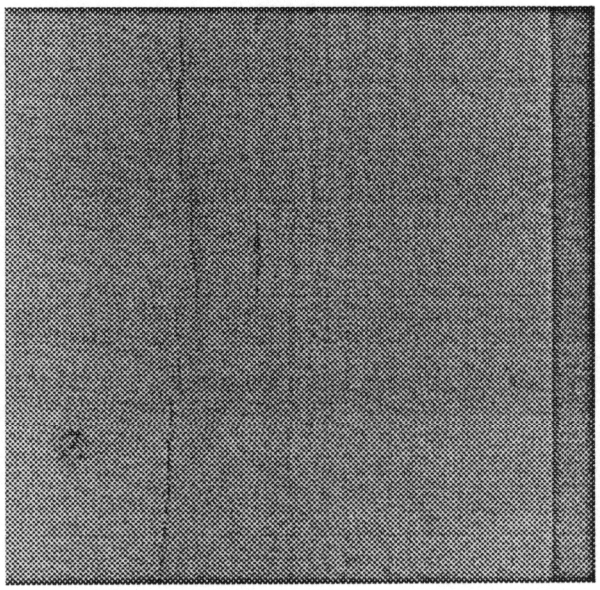


Figure 7.2. Typical splits/checks.

unlabeled region, R, whose confidence vector indicates it might be a split/check. If the crack detector successfully finds a crack within R, then R is labeled as a split/check. Note that the application of the crack detector to the original image represents a top-down, i.e., model-driven, method for determining whether the bottom-up initial labeling hypothesis as manifested in the confidence vector is true or not.

The crack detection in R is performed by 1) applying four directional masks shown in Figure 7.3 to each pixel of R, 2) counting the number of pixels where maximum response of the four masks is greater than a threshold, and 3) determining if N/Area(R) > T, where Area(R) represents the area of region R, N is the count obtained by step 2, and T is a threshold. If R satisfies step 3, then region R is considered to have a "crack" in it. Otherwise, R is considered to have no crack in it.

Next, regions representing "ambiguous" examples of splits/checks are detected. Let R be an unlabeled region. If split/check component of the R's confidence vector is low and if there is an adjacent region, R', that has been already assigned a split/check label and if the elongatedness of R merged with R' is greater than that of either R or R', then R is labeled as a split/check. This rule detects regions that have low confidence of being a split/check but are aligned with a region already recognized as being a split/check.

After all the initial labeling has been performed, DEFECT_OBJECTs labeled as splits/checks are generated and their properties are computed. Two DEFECT_OBJECTs are connected into a single DEFECT_OBJECT, if they are colinear and have a small gap between them. The reason for this merging is that a single split/check might be broken into several disconnected pieces, due to the imperfect segmentation.

4	-1	-6	-1	4
4	-1	-6	-1	4
4	-1	-6	-1	4
4	-1	-6	-1	4
4	-1	-6	-1	4

4	4	4	4	4
-1	-1	-1	-1	-1
-6	-6	-6	-6	-6
-1	-1	-1	-1	-1
4	4	4	4	4

4	4	4	-1	-6
4	-1	-1	-6	-1
4	-1	-6	-1	4
-1	-6	-1	-1	4
-6	-1	4	4	4

-6	-1	4	4	4
-1	-6	-1	-1	4
4	-1	-6	-1	4
4	-1	-1	-6	-1
4	4	4	-1	-6

Figure 7.3. Four directional masks for crack detection.

Finally, to verify that a DEFECT_OBJECT, labeled split/check is truly a split/check, the following tests are performed.

TEST 1. The DEFECT_OBJECT should not be small. (The basic feature *Area* is used in this test.)

TEST 2. The DEFECT_OBJECT should not touch the background or image frame boundary significantly. (Basic features *Touch_Background* and *Touch_Image_Boundary* are used.)

TEST 3. The DEFECT_OBJECT should have a "crack" inside it. (The crack detection algorithm described above is used in this test.)

TEST 4. The DEFECT_OBJECT should be long enough. (The basic feature *Elongatedness* is used in this test.)

Each test generates confidence values that the DEFECT_OBJECT is / is not a split/check based on its criterion. Confidence values generated by those tests are combined to yield a final confidence or belief value using Dempster's rule of combination. Only if the final confidence value that the DEFECT_OBJECT is a split/check is greater than 0.5, is the DEFECT_OBJECT's split/check label verified.

7.2.2.2.2 Hole Detection Procedure

This KS is applied to a region only if the region's confidence vector indicates it might be a hole. There are two types of holes : small and large holes. Usually, birds and worms make small holes which appear circular and very dark. Large holes are caused by the dislocation of knots. They appear very dark with the shape of either a circle or an ellipse. Figure 7.4 shows typical small holes and a large hole.

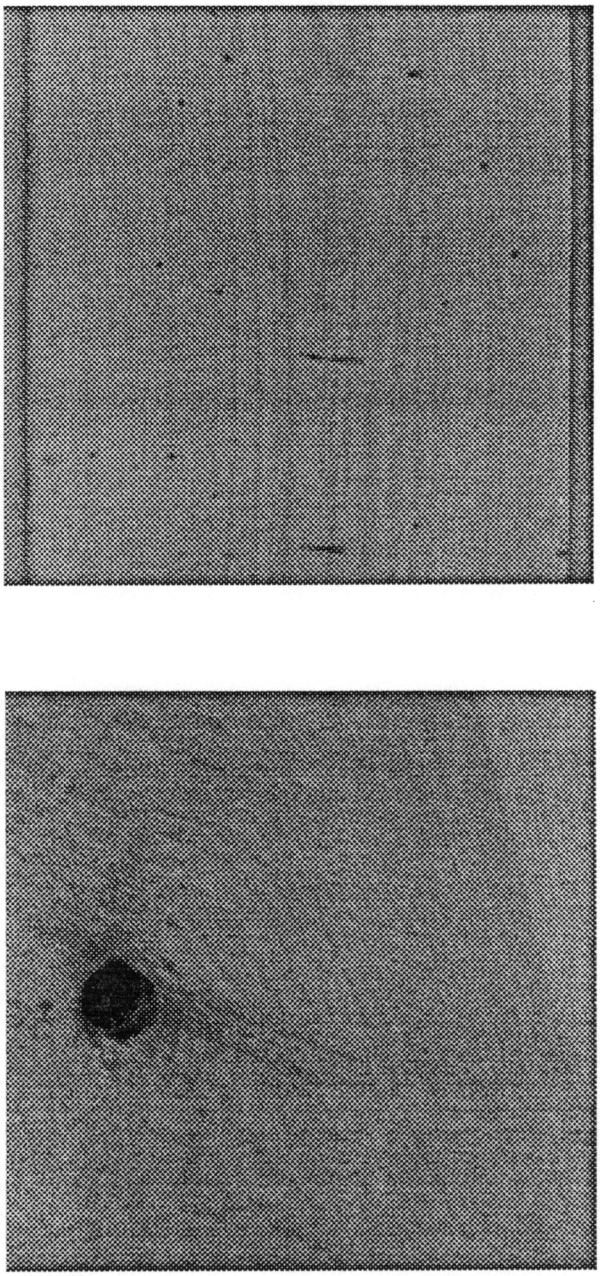


Figure 7.4. Typical holes.

To detect regions representing "exemplary" holes, all regions that have a positive "hole" component in their vector are considered. Let R be such an unlabeled region. If the area of R is small, the elongatedness of R low, and R much darker than its adjacent regions, then R is assigned the label hole. (This rule is based on the way small holes typically manifest themselves in images. Basic features *Area*, *Elongatedness*, and new feature *Local contrast* are used in this rule.) If the area of R is large and R is "very dark," then R is also assigned the label hole. (This rule is based on the way large holes typically manifest themselves. Features *Area* and *Average gray level* are used.)

Next, regions representing "ambiguous" examples of holes are labeled using the following rule. Let R be an unlabeled region. If an adjacent region R' has been labeled as a hole and if R and R' have similar average gray values, then R is assigned the label hole. (The basic feature *Average gray level* is used in this rule.)

Once the initial labeling process has been performed, DEFECT_OBJECTs labeled as holes are generated and their properties are computed. Finally, to verify that a DEFECT_OBJECT, labeled hole is truly a hole, the following tests are performed.

TEST 1. The DEFECT_OBJECT should have a large confidence for holes (computed using fuzzy logic).

TEST 2. The DEFECT_OBJECT should not be too small. (The basic feature *Area* is used in this test.)

TEST 3. The DEFECT_OBJECT should not be long. (The basic feature *Elongatedness* is used in this test.)

TEST 4. The DEFECT_OBJECT should be compact. (The basic feature *Compactness* is used in this test.)

TEST 5. If the DEFECT_OBJECT is small, it should have a *concave* shape

to its gray-level spatial distribution. (A DEFECT_OBJECT is called *concave* if N1 = N2 = 1 and the minimum gray level of the region is very low, where N1 (N2) is the number of valleys in gray levels of pixels lying on the horizontal (vertical) line passing through the centroid of the region.)

TEST 6. If the DEFECT_OBJECT is not small, then it should have low average gradient magnitude. (The basic feature *Area* and new feature *Average gradient magnitude* are used in this test.)

TEST 7. If the DEFECT_OBJECT is not small, it should have a low gray-level variance. (The basic feature *Area* and a new feature *Gray-level variance* are used in this test. Note that the use of new features in the various tests represents additional top-down processing employed to verify label assignment.)

Each test generates confidence values that the DEFECT_OBJECT is / is not a hole based on its criterion. Confidence values generated by those tests are combined to yield a final confidence or belief value using Dempster's rule of combination. Only if this confidence value is high, is the DEFECT_OBJECT's label hole verified.

7.2.2.2.3 Wane Detection Procedure

This KS is applied to a region only if the region's confidence vector indicates it might be wane. Wane is defined to be the absence of wood. Wane results because round trees are sawn into rectangular boards. Wane manifests itself in two ways. One way is when bark remains attached to the board. When this happens there are usually two types of bark comprising the wane area. One type is external bark. External bark is very dark and is on the edge of the board. The other type of bark is internal bark. Internal bark is lighter in color than external bark and hence is usually separated from external bark during segmentation. Internal bark can be

recognized because it lies between external bark and clear wood and because it is highly elongated. Typical examples of external and internal bark are shown in Figure 7.5.

If the bark does not remain attached to the board, the area of wane still typically has a dark color. The dark color is the result of cambium which for hardwood is always dark. While the color of the cambium is much lighter than the color of either external or internal bark, it is dark enough to be easily separated from clear wood during segmentation. Because of this, this can be treated in the same manner as external bark by this defect detection procedure.

External bark is first detected and then internal bark is detected, if it exists. To detect external bark, 1) label regions of high confidence as wane, and then 2) label a region as wane if it is surrounded by regions already labeled as wane.

Internal bark is detected using the following steps. 1) Label a region as wane if it is dark, highly elongated, and shares a significant portion of common boundary with regions already recognized as wane. (*Average gray level* and *Elongatedness* are used in this rule.) 2) Label a region as wane if it is surrounded by regions already labeled as wane.

Once initial labeling has been performed, DEFECT_OBJECTs labeled as wane are generated and their properties are computed. Finally, to verify that a DEFECT_OBJECT, labeled wane is truly wane, the following tests are performed.

TEST 1. The DEFECT_OBJECT should touch the background considerably. (The basic feature *Touch_Background* is used in this test.)

TEST 2. The DEFECT_OBJECT should not be too long. (The basic feature *Elongatedness* is used in this test.)

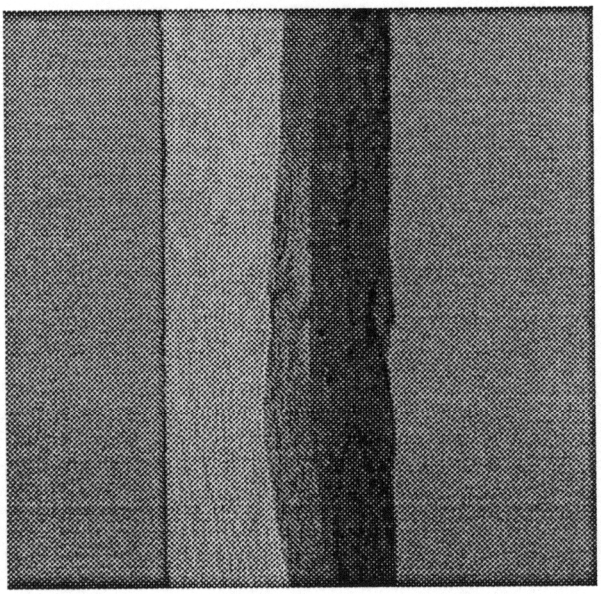


Figure 7.5. Typical wane.

TEST_3. The DEFECT_OBJECT should be much darker than the adjacent regions around its boundary. (A new feature *Local contrast* is used in this test. Again the use of an additional feature represents additional top-down processing used in this label verification.)

Each test generates confidence values that the DEFECT_OBJECT is / is not a wane based on its criterion. Confidence values generated by those tests are combined to yield a final confidence or belief value using Dempster's rule of combination. Only if this confidence value is greater than 0.5, is the DEFECT_OBJECT's label knot verified.

7.2.2.2.4 Knot Detection Procedure

This KS is applied to a region only if the region's confidence vector indicates it might be a knot. It is very difficult to characterize knots because they appear quite different in gray level, shape, and size. One can have both a very small knot and very large knot. Knots can be circular or elliptical. Knots are usually dark, but average gray value difference between knots can be large. Some typical knots are shown in Figure 7.6.

In order to detect regions, representing "exemplary" knots, all regions that have high confidence level are labeled as knots. To detect regions representing "ambiguous" knots, a test is applied to each unlabeled region R that has some degree of confidence of being a knot. If R shares a significant portion of its region boundary with regions that have already been labeled as knots, then R is assigned a knot label. This rule combines many regions formed by the segmentation but that go together to form a single whole knot structure.

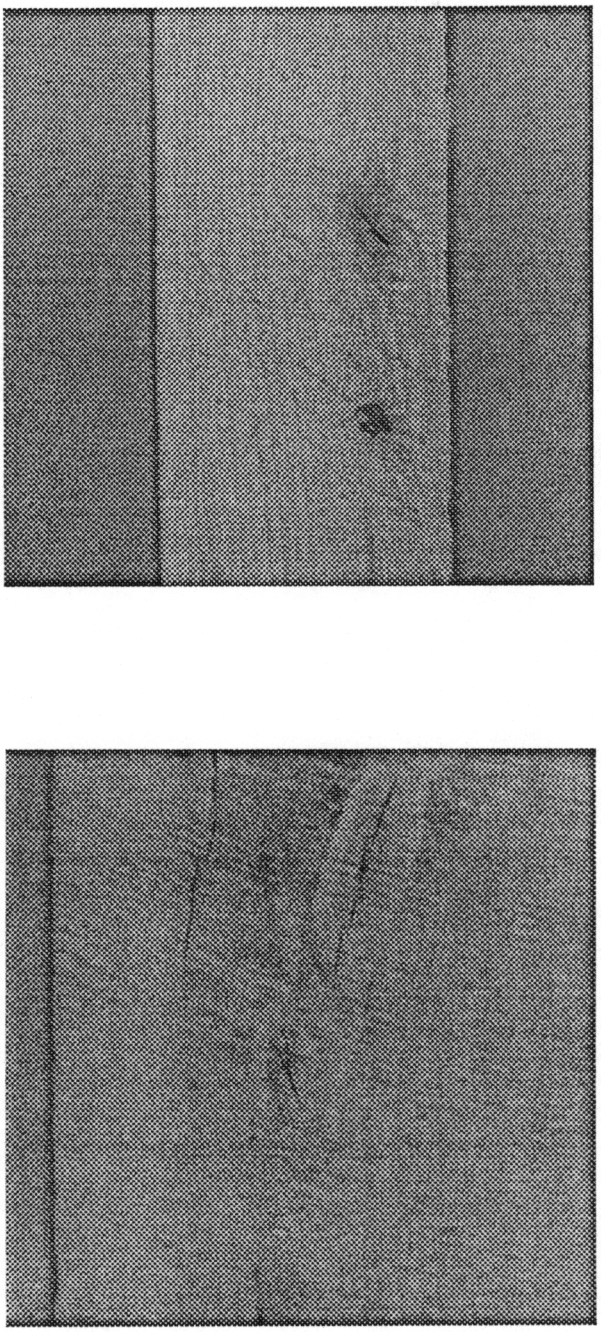


Figure 7.6. Typical knots.

Once initial labeling has been performed, DEFECT_OBJECTs labeled as knots are generated and their properties are computed. To verify that a DEFECT_OBJECT, labeled knot is truly a knot, the following tests are performed.

TEST 1. If the DEFECT_OBJECT is small, it should not touch the background. (Basic features *Area* and *Touch_Background* are used in this test.)

TEST 2. The DEFECT_OBJECT should be much darker than its adjacent regions. (A new feature *Local contrast* is used in this test.)

TEST 3. The DEFECT_OBJECT should have a high average gradient magnitude by the Sobel operator. (A new feature *Average gradient magnitude* is used in this test.)

TEST 4. The DEFECT_OBJECT should have large gray-level variance. (A new feature *Gray-level variance* is used in this test.)

Each test generates confidence values that the DEFECT_OBJECT is / is not a knot based on its criterion. Confidence values generated by those tests are combined to yield a final confidence or belief value using the Dempster's rule of combination. Only if this confidence value is greater than 0.5, is the DEFECT_OBJECT's label knot verified.

7.2.2.3 A KS for Accurate Detection of Knot Boundary

Some dried and sound knots are very difficult to detect during segmentation. These knots have either the same color or almost the same color as clear wood. For such knots the segmentation procedure completely fails to detect their presence. However, some of these knots have a split in their center. The segmentation procedure can and does detect the presence of the split while not detecting anything

else. These splits when present are usually not aligned along the usual wood grain direction, as a normal split/check would be and hence can be differentiated from normal splits/checks.

KS KNOT_BND is a top-down processing procedure that attempts to find accurate boundaries for these knots that have a split in them. KNOT_BND is invoked when a split/check detected by the split/check detection procedure has an orientation greater than some threshold value away from the normal wood grain direction. For the images used in this study, normal wood grain is vertically oriented. Hence this procedure is invoked if a split/check is found that has an orientation significantly different than vertical. In this KS, a priori knowledge is used that typical knots have an elliptical shape. Thus, the problem is to find an ellipse of the knot boundary based on the detected split/check. In general, an ellipse is defined by the center point (x_c,y_c), orientation θ, the length of major axis a, and the length of the minor axis b. These parameters are illustrated in Figure 7.7.

The processing steps performed by KS KNOT_BND are as follows.

Step 1. Find the center point, (x_c,y_c), of the ellipse. The point (x_c,y_c) is assumed to be the center point of (x_1,y_1) and (x_2,y_2), which are two end points of the detected split/check.

Step 2. Find θ, the orientation of the major axis of the ellipse. It is assumed that this orientation is identical to the orientation of the split/check. The orientation of the split/check is given by the slant of the line passing through the two points (x_1,y_1) and (x_2,y_2).

Step 3. Compute the major axis length, a, of the ellipse. It is assumed that $2a$ is slightly longer than the length of the split/check. The length of the split/check is defined by the distance between the points (x_1,y_1) and (x_2,y_2).

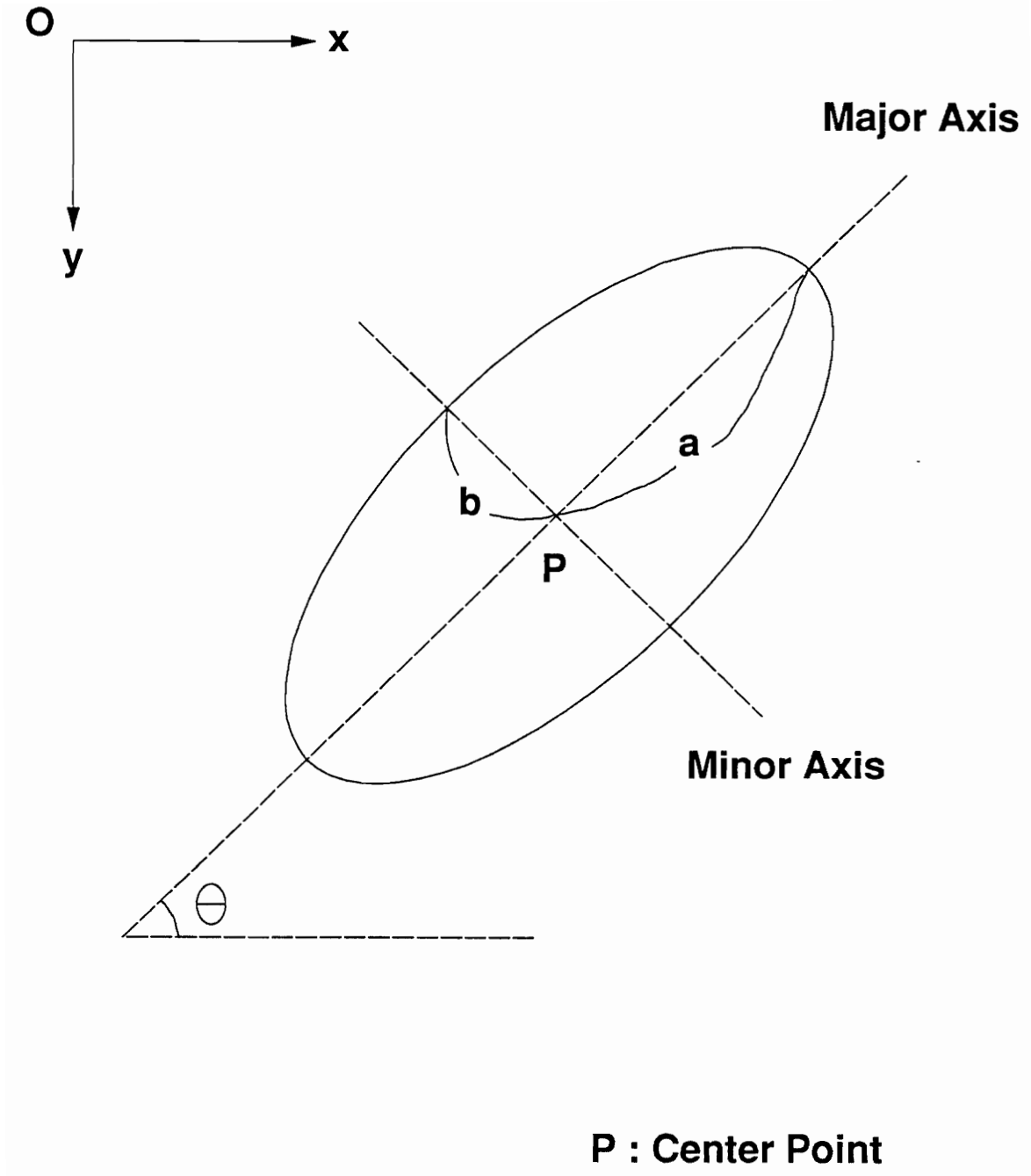


Figure 7.7. Notations for parameters of an ellipse.

Step 4. Compute the minor axis length, b, of the ellipse.

4.1 Apply a Canny edge detection operator [CAN86], which is sensitive to the major axis orientation of the ellipse. The edge detection operator is applied only to the area which is slightly larger than the minimum bounding rectangle of the split/check.

4.2 Consider only pairs of the edge points that are almost symmetrical around the major axis. (Edge points resulting from the split/check are not considered.) Among these pairs of edge points, find the pair farthest from the major axis. This pair of edge points is assumed to lie in the ellipse and the minor axis. Then, the minor axis length is given by the average of distances from this pair of edge points to the major axis. If no pair of edge points is found, then it is considered that no knot exists around the split/check, and return.

Step 5. Assign a defect label, knot, to all pixels that are inside the ellipse.

7.2.2.4 Verification Using Spatial Constraints

As described earlier, a KS VRFDFC is activated for verifying defects using spatial constraints, once all defect detection procedures have been run and finished. For each remaining DEFECT_OBJECT, O, to be verified, none of the following conditions can hold:

1) LABEL(O)=split/check and ENCLOSED_BY(O,X), where LABEL(X)=wane
2) LABEL(O)=split/check and ADJACENT(O,X), where LABEL(X)=hole
3) LABEL(O)=knot and ADJACENT(O,X), where LABEL(X)=hole
4) LABEL(O)=knot and ADJACENT(O,X), where LABEL(X)=wane

In the above, LABEL(O) denotes a defect label of DEFECT_OBJECT O, and both ENCLOSED_BY and ADJACENT are relational predicates. ENCLOSED_BY(O,X) is true if O is enclosed by X, and ADJACENT(O,X) is true if O is adjacent to X.

If the validation procedure fails to verify the label assigned to a DEFECT_OBJECT, the DEFECT_OBJECT is removed from further considerations.

7.3 Experimental Results

7.3.1 Experimental Setup

The software of the machine vision system for lumber grading was implemented in FORTRAN 77 and C on the VAX 11/785 computer system in the Spatial Data Analysis Laboratory at Virginia Tech. The above described system was trained and tested on an image data base of rough lumber. The data base was created by digitizing a number of rough hardwood lumber boards (over 160) from 4 species, i.e., cherry, red oak, yellow poplar, and maple. These species were selected because approximately 50 percent of all household wood furniture is made from these species. Hence having a system that would only grade these species would still be a viable commercial product. 41 cherry boards, 52 red oak boards, 26 yellow poplar boards, and 48 maple boards were selected. The samples were chosen to reflect the variation in appearance of each of the four defects, knots, holes, wane, and splits/checks

An 8 inch by 8 inch area of each sample was selected and scanned. This area was digitized using a 480x512x8 bits resolution black and white camera. The spatial resolution of these is approximately 64 points per inch. To obtain the desired color images color filters were used. Tungsten halogen bulbs were used to approximate afternoon sun viewing conditions. To normalize for the difference in spectral

sensitivity of the silicon/tungsten halogen combination neutral density filters were used. The same lighting conditions, camera settings, and viewing angle were employed in creating the color image of each sample. Each color image was shading corrected to remove any nonuniformities in lighting or sensitivity across the camera's imaging array [SAW77].

7.3.2 Illustration of Each Processing Step

To illustrate the results obtained from the various processing steps one sample board image will be used. This example is shown in Figure 7.8. This sample is a rough yellow poplar board that contains a large knot. Figure 7.9 shows the gray-level histogram of the board image shown in Figure 7.8. The intervals between consecutive thresholds are each given a unique gray-level. Figure 7.10 shows the results obtained from the initial segmentation operation on the image of Figure 7.8. Figure 7.11 shows the boundaries of connected regions of the segmented image of Figure 7.10. Figure 7.12 shows the results obtained after eliminating small regions with area less than six pixels. Figure 7.13 shows the region boundaries remaining after the region merging step. Figure 7.14 shows the final output of the results obtained from the recognition module using the neural network approach. The knot was correctly recognized by the system. The number of regions at each step after initial segmentation is shown below.

The number of regions before small region elimination : 337
The number of regions after eliminating 1-pixel regions : 139
The number of regions after small region elimination : 45
The number of regions after region merging : 16

Notice that the number of regions has been significantly reduced after small region elimination. Thus, only a small number of regions are input to the region merging step that performs more computationally intensive operations than the small region elimination operation. Although these numbers varied depending on the input image, small region elimination followed by region merging always markedly reduces the number of regions.

To show the necessity of the label verification step after the initial labeling step in the high-level processing, four boards from different species were processed. These boards all contain defects that were perfectly recognized by the vision system. Table 7.2 presents the number of DEFECT_OBJECTs verified in the label verification step among DEFECT_OBJECTS generated in the initial labeling step for each board. Each entry *i/j* in this table means that *i* DEFECT_OBJECTs have been verified among *j* DEFECT_OBJECTs generated. Note that many of DEFECT_OBJECTs generated in the initial labeling step were not verified in the label verification. Without the label verification step, a number of DEFECT_OBJECTs which are actually clear wood would be misclassified as a defect.

Table 7.3 shows the CPU time involved in each processing step for the above sample image shown in Figure 7.8. Notice that low-level processing is computationally more intensive than high-level processing. However, low-level processing steps can be implemented in high speed pipeline architectures. For example, connected component labeling can be implemented in a pipeline architecture as in [SAN88]. Also, defect detection procedures can be run concurrently because they are completely independent of each other. This advantage could save a significant amount of processing time in the systems which support parallel concurrent processing. A tightly coupled MIMD architecture machine would seem the best architecture for

TABLE 7.2

The Number of DEFECT_OBJECTS Verified in the Label Verification among DEFECT_OBJECTs Generated in the Initial Labeling Step

	Defect Detection Procedures				
	Splits/Checks	Holes	Wane	Knots	Overall
A cherry board	0/2	1/4	1/1	0/4	2/11
A Red Oak Board	0/0	0/2	1/1	1/2	2/5
A Yellow Poplar Board	0/1	0/1	0/0	1/1	1/3
A Maple Board	1/3	0/3	0/0	1/3	2/9

the high speed implementation of the recognition system.

A processing example that illustrates the operation of KNOT_BND is shown in Figure 7.15. This figure shows an image of a rough cherry sample containing a knot and boundaries of segmented regions. Note that the knot was barely extracted by the segmentation module. However, once the split/check detection procedure has detected the split/check inside the knot whose orientation is larger than the threshold angle, KS KNOT_BND is invoked to search for an accurate boundary of the knot. Figure 7.16 shows a subimage of Figure 7.15, highlighting the area containing the knot. Figure 7.17 shows the results of applying the Canny's edge detection operator to a focus window around the split/check, which is slightly larger than the minimum bounding rectangle of the split/check. The rectangle area shown indicates the area of the focus window, to which the edge operator is applied. Figure 7.18 shows the elliptical boundary of the knot estimated by the KNOT_BND. Notice that this is almost the correct boundary of the knot.

7.3.3 More Examples

Three processing examples for each of four species using the neural network label verification method are presented to demonstrate capabilities of the current system. They are shown in Figure 7.19 - Figure 7.30. Each of these figures has three parts. Part (a) of each shows a black and white image of a sample. Part (b) shows the results of the segmentation and Part (c) shows the results of the scene analysis operation. In Part (c) each defect is assigned a unique gray level with its name. (White area represents clear wood.) The system was also tested on surfaced boards from oak, pine, and walnut. The recognition results were satisfactory. Figure 7.31 - Figure 7.34 show the processing results on some surfaced boards.

TABLE 7.3

CPU Time for Each Processing Step :

A 480x512 image

Processing Step	CPU Time (Sec)
Background Extraction	7.98
Segmentation	5.39
Connected Component Labeling	15.28
Small region Elimination	5.62
Region Merging	4.46
Region Property Extraction	1.19
High-Level Recognition	8.21
TOTAL	48.13

Recently the Spatial Data Analysis Laboratory has started collecting images of 4-foot long boards from a prototype system using a color line scan camera. (For details of this prototype system, see [CON90].) The size of the 4-foot long board image is 1536x512 at the resolution of 64 pixels per inch in the cross board direction and 32 pixels per inch in the down board direction. These images are much more similar in their characteristics to real images that would have to be handled by the commercial computer vision system. Some of these images were used to test the adaptability of the vision system to images of a larger size. The preliminary results show that the system is able to detect all major defects. Two examples for this processing are presented in Figure 7.35 - Figure 7.40.

To show the effectiveness of the small region elimination and region merging step, the number of regions at each step after segmentation of Figure 7.35 is given below.

The number of regions before small region elimination :	1427
The number of regions after eliminating 1-pixel regions :	544
The number of regions after small region elimination :	107
The number of regions after region merging :	24

Note that the small region elimination step removed more than 90 percent of the original regions after segmentation. After region merging, only 24 regions are left. We can also notice that the numbers of regions after these steps are proportional to the area of the image. (Compare these numbers of regions with those of a 480x512 image.)

Table 7.4 shows CPU time for each processing step taken to process a 4-foot long red oak board. The total CPU time taken to process the 4-foot long board is approximately 3.4 times the total CPU time of a 512x480 image shown in Table

TABLE 7.4

CPU Time for Each Processing Step :

A 1536x512 Image of a 4-Foot Long Board

Processing Step	CPU Time (Sec)
Background Extraction	35.12
Segmentation	18.55
Connected Component Labeling	59.31
Small region Elimination	21.37
Region Merging	11.66
Region Property Extraction	0.33
High-Level Recognition	18.31
TOTAL	164.68

7.4. As in the 480x512 image, CPU time for high-level recognition is much less than that for low-level processing. In general, the CPU processing time is proportional to the area of the input image. Thus, it is feasible to roughly estimate the CPU time for processing the full length board (typically, 9-12 foot long). For the 12-foot long board, it would take about 495 seconds for the whole processing of one side of a board including 55 seconds for high-level processing. (Taking into account that two sides must be inspected for grading, about 160 million instructions per second (MIPS) computing power is required in order to grade a full length board at the speed of a human grader, i.e., 2 linear feet per second.) This amount of time could be dramatically reduced because the low-level processing module can be implemented in a high speed pipeline architecture and each defect detection procedure can be run concurrently in a parallel computing system. Therefore, a machine vision system for lumber grading could be built which can run in real-time.

7.3.4 Performance Evaluation

The performances of the vision system using the three approaches (neural network, k-nearest neighbor, and rule-based) in the label verification were evaluated and compared using an extensive image data base of rough hardwood lumber. First, the image data base was partitioned into two sets. One set of board images was used to construct a training set that is used in the neural network and k-nearest classification. The training set consisting of 100 samples was carefully constructed so that it includes most variations of each defect type. Each sample is a feature vector of a DEFECT_OBJECT verified in the label verification step. Ten features are used as elements of the feature vector. Each sample is known to belong to one of five classes: clear wood, splits/checks, holes, wane, and knots. There are 20 samples for each class. Samples for clear wood are collected using

DEFECT_OBJECTs that are actually clear wood. The other set of the image data base was used to test the performance of the system implemented using each approach.

Table 7.5 shows the number of boards of each species used in the training phase and the testing phase. Most boards used in the testing were different from those used in the training. Three cherry boards and two maple boards were used in both the training and the testing to obtain more test samples for holes and wane, respectively. However, the DEFECT_OBJECTs used in the training set was excluded in the performance evaluation. This ensured the complete independence between training and testing. In this table, the number of segmentation failures represents the number of boards whose images caused the complete failure in segmentation process. Table 7.6 shows the number of defects which appear on the boards that were used in the testing. (Of course, the defects used in the training phase were not counted in this table.)

The performances of the vision system using the three approaches in the label verification on the testing set were compared in two aspects. First, the capabilities of differentiating each defect type from clear wood are shown in Table 7.7, 7.8, and 7.9. The number of defects for each defect type in these tables is the number of *true* defects. If more than half of a true defect area is recognized as one or several defect types, then this defect is assigned the most dominant defect type among defect types associated. If no clear wood area in a testing board is misclassified as a defect type, then it is considered as the correct classification of the class clear wood. If at least one clear wood area is misclassified as a defect type, then it is considered as the classification of the class clear wood as the most dominant defect type among these falsely detected areas.

Note that the *k*-nearest approach is better than the neural network approach in detecting defects except knots. However, the *k*-nearest neighbor classifier has a higher false alarm rate than the neural network, i.e., the *k*-nearest neighbor classifier misclassifies clear wood as a defect more often than the neural network. Particularly, many of clear wood areas were confused with knots. The parameters or thresholds used in the rule-based approach were determined based on subjective criteria, not based on the training set. This might be the reason for the low detection accuracy of holes and wane. The detection of these defect types would be improved by fine tuning the thresholds used by each test or rule in the label verification step of the corresponding procedures. However, this process would be tedious and time-consuming. On the other hand, parameters or weights in the neural network can be obtained automatically by the learning algorithm once a proper training set is provided.

Second, the capabilities of differentiating clear wood from defects are shown in Table 7.10, 7.11, and 7.12. As expected, the overall correct classification rate was improved in each approach because it is a correct classification that a defect of a defect type is classified as a different defect type.

Some comments should be made concerning the training phase of each approach. Usually, successful training of a neural network requires a large number of epochs, i.e., presentations of the training set. To train the neural network used above it took about 1500 epochs on average. (The exact number of epochs required for training depends on the assignment of the initial weights of the network.) However, once the neural network has been trained, classification can be performed very fast. The straightforward *k*-nearest neighbor classifier requires storing all samples of the training set. To classify each test sample, the k-nearest neighbor classifier needs

to compute distances between the test sample and each stored training sample. This computation is proportional to the number of the training samples. By storing only samples around the decision boundary [DEV82], a significant amount of storage space and computation time can be saved with little degradation in performance. The thresholds or parameters used in the rule-based approach must be fine-tuned for optimum performance based on a training set. This takes a significantly longer development time than either the neural network or the k-nearest neighbor approach.

TABLE 7.5

The Number of Rough Boards of Each Species Used in Training and Testing

	Cherry	Red Oak	Yellow Poplar	Maple	Total
# boards in the data base	41	52	26	48	167
# segmentation failure	2	4	1	3	10
# boards used in training	19	16	5	25	65
# boards used in testing	23	32	20	21	96

TABLE 7.6

The Number of Defects on Boards of Each Species Used in Testing

	Cherry	Red Oak	Yellow Poplar	Maple	Total
Split/Check	9	11	5	2	27
Hole	28	4	0	1	33
Wane	1	8	5	3	17
Knot	24	25	22	18	89

TABLE 7.7

Differentiation of Each Class

Using the Neural Network Approach

		Assigned Class						
		CW	SP	HL	WN	KN	Total	% Correct
True Class	CW	86	3	1	0	6	96	89.6 %
	SP	5	21	0	0	1	27	77.7 %
	HL	8	1	21	0	3	33	63.6 %
	WN	4	0	0	13	0	17	76.5 %
	KN	14	2	0	2	71	89	79.8 %
	Total	117	27	22	15	81	262	
Overall Correct Classification (%)								80.9 %

TABLE 7.8

Differentiation of Each Class

Using the *k*-Nearest Neighbor Approach (*k*=5)

		Assigned Class						
		CW	SP	HL	WN	KN	Total	% Correct
	CW	56	2	1	0	37	96	58.3 %
True	SP	4	22	1	0	0	27	81.5 %
Class	HL	4	0	27	0	2	33	81.8 %
	WN	1	0	0	16	0	17	94.1 %
	KN	15	3	4	3	64	89	71.9 %
	Total	80	27	33	19	103	262	
		Overall Correct Classification (%)						70.6 %

TABLE 7.9

Differentiation of Each Class

Using the Rule-Based Approach

		Assigned Class						
		CW	SP	HL	WN	KN	Total	% Correct
	CW	72	7	5	0	12	96	75.0 %
True	SP	2	20	0	0	5	27	74.1 %
Class	HL	6	0	22	0	5	33	66.7 %
	WN	4	0	0	11	2	17	64.7 %
	KN	20	0	2	0	67	89	75.2 %
	Total	104	27	29	11	91	262	
		Overall Correct Classification (%)						73.3 %

TABLE 7.10

Differentiation of Clear Wood and Defects Using the Neural Network Approach

		Assigned Class			
		Clear Wood	Defects	Total	% Correct
True Class	Clear Wood	86	10	96	89.6 %
	Defects	31	135	166	81.3 %
	Total	117	145	262	
		Overall Correct Classification (%)			84.4 %

TABLE 7.11

Differentiation of Clear Wood and Defects Using the *k*-Nearest Neighbor Approach

		Assigned Class			
		Clear Wood	Defects	Total	% Correct
True Class	Clear Wood	56	40	96	58.3 %
	Defects	24	142	166	85.5 %
	Total	80	182	262	
		Overall Correct Classification (%)			75.6 %

TABLE 7.12

Differentiation of Clear Wood and Defects Using the Rule-Based Approach

		Assigned Class			
		Clear Wood	Defects	Total	% Correct
True Class	Clear Wood	72	24	96	75.0 %
	Defects	32	134	166	80.7 %
	Total	104	158	262	
		Overall Correct Classification (%)			78.6 %

7.4 Discussion

The research described here is aimed at developing a computer vision system to be used as part of an automated lumber grading system for rough hardwood lumber. The purpose of the computer vision system is to locate and identify surface defects that affect the grade of boards. The current system can detect four of the most common types of defects : knots, holes, wane, and splits/checks. The output of the vision system is used as input to a program that establishes the grade of a board. The system has been tested on a number of boards from four hardwood species. The results of the extensive tests suggest that species independent methods can be found for accomplishing the required tasks.

However, further research is required. The future research should be directed toward the following items.

1) Expanding the list of defects that can be identified by the system. Decay and stain must be added to the current list of defects because they are also influential in hardwood grading. Adding a new defect detection procedure to existing ones is relatively easy once sufficient typical samples of that defect type have been collected because those defect detection procedures are completely independent of each other. Since decay is typically yellowish and stain is gray or dark blue, color information seems to be needed to detect these defects.

2) Improving the accuracy of defect recognition is also very important. There are two ways to improve the recognition accuracy of the vision system. One is to refine the capability of the segmentation module. The current segmentation method has been shown satisfactory but it has difficulty detecting light knots. An X-ray scanner might be used for better knot detection because knots are usually more

dense than any other parts of wood. To attain better detection of holes and/or wane, three-dimensional information such as range information could be used. Another way for improving the recognition accuracy is to strengthen the top-down methods in the recognition module. The current recognition module has some specialized top-down methods within each defect detection procedure. When a region (or DEFECT_OBJECT) is considered to have some confidence of being a defect type, top-down methods search for additional evidences or features supporting for the defect type. However, current implementation lacks a re-segmenting capability when a need to change the form of the region (or DEFECT_OBJECT) arises from the search results of the top-down methods.

3) If a system is to be used in industry, it is critical that it run in real-time. This requires the development of hardware and software architecture appropriate to the real-time operation. Much effort to find algorithms and data structures most appropriate for real-time processing has been made. As we described earlier, it is possible to dramatically reduce the processing time by implementing the low-level module in a pipeline hardware architecture. Furthermore, implementing defect detection procedures on a parallel computing system could also save the processing time. These works remain to be investigated more closely.

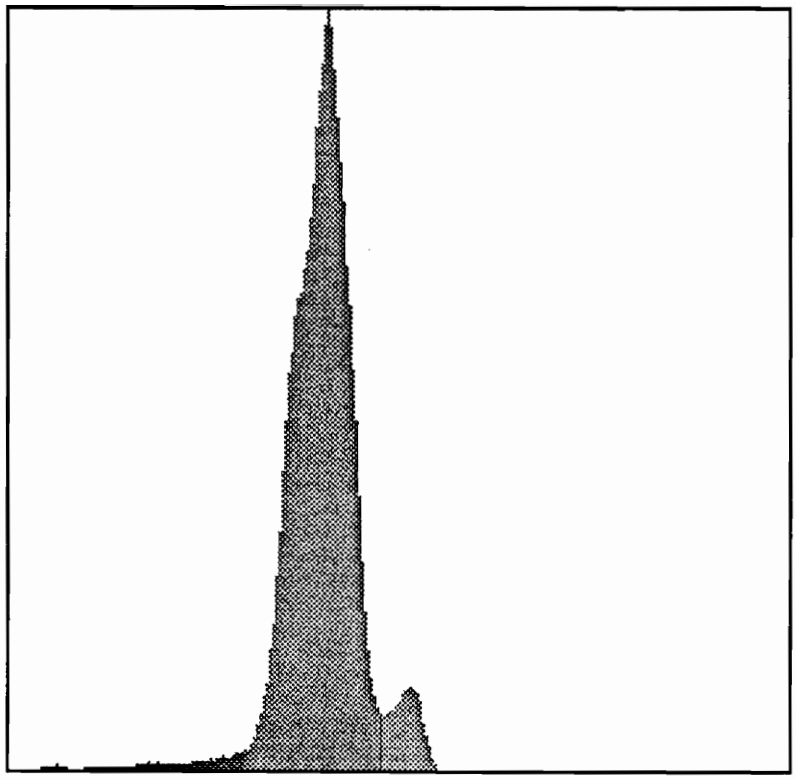


Figure 7.9. The gray-level histogram of pixels corresponding to the board area of the image in Figure 7.8. Each interval between consecutive thresholds is assigned a unique gray-level.

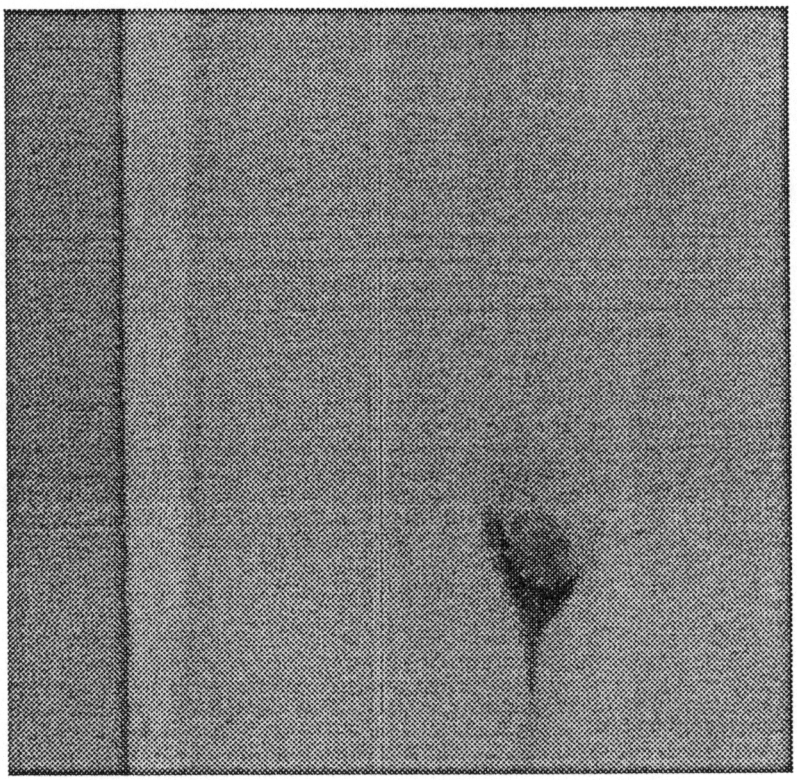


Figure 7.8. An Image of a rough yellow poplar board.

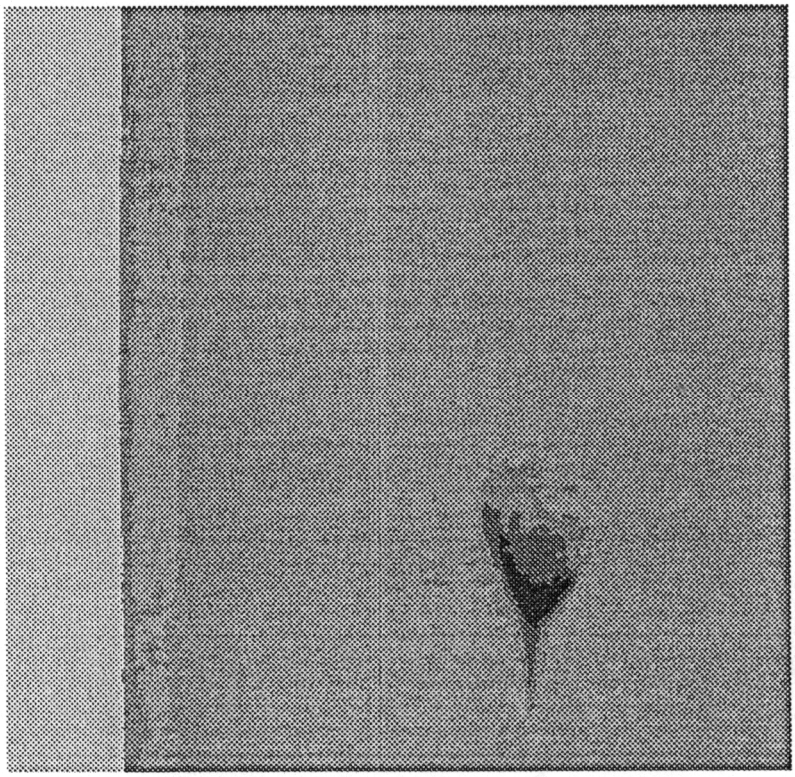


Figure 7.10. Results of initial segmentation using the thresholds shown in Figure 7.9.

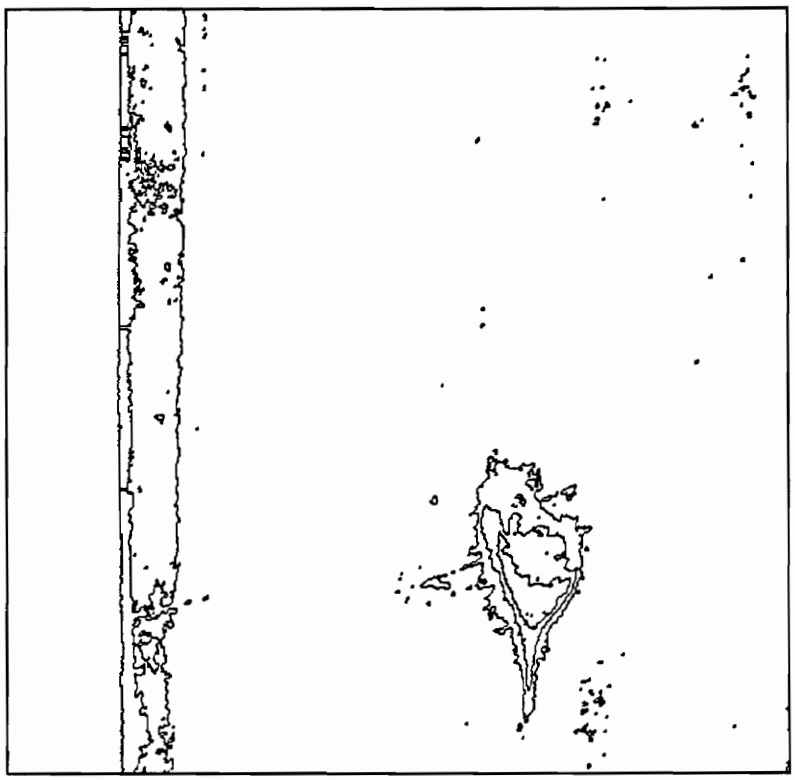


Figure 7.11. Boundaries of connected regions of the segmented image in Figure 7.10.

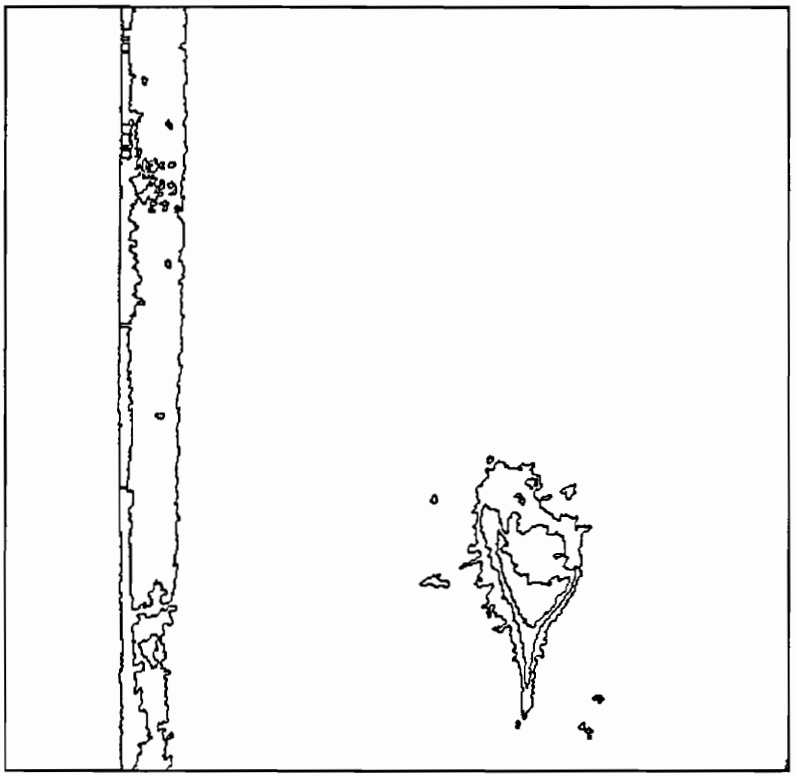


Figure 7.12. Results obtained after eliminating small regions with area less than six pixels from Figure 7.11.

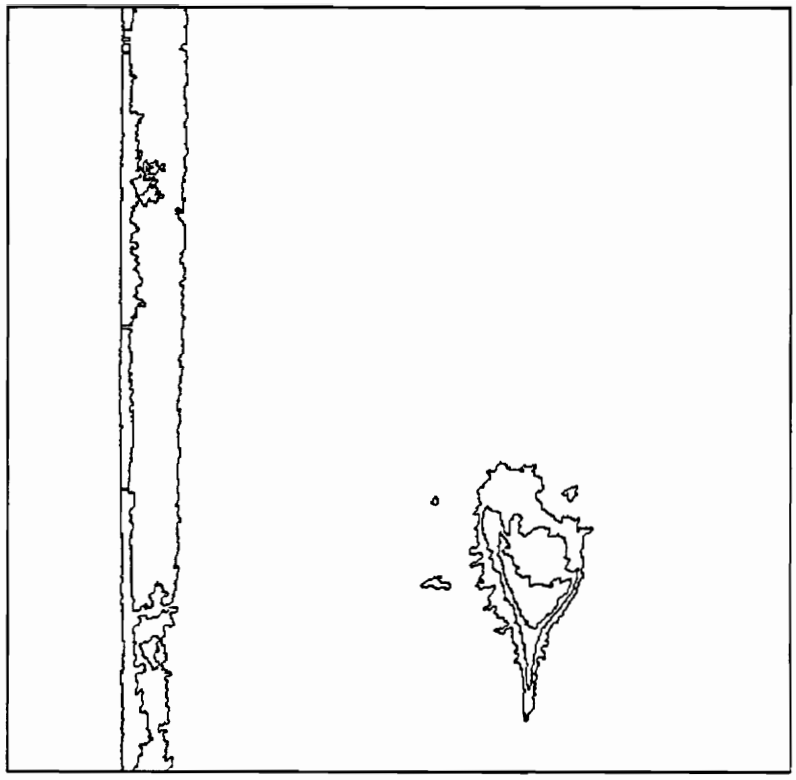


Figure 7.13. Results obtained after the region merging operation applied on Figure 7.12.

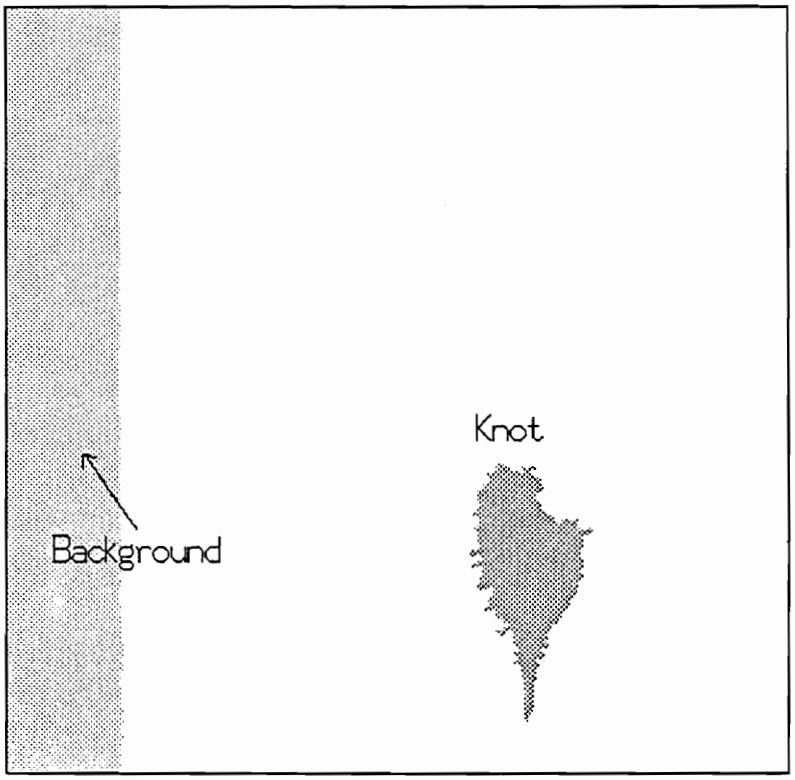
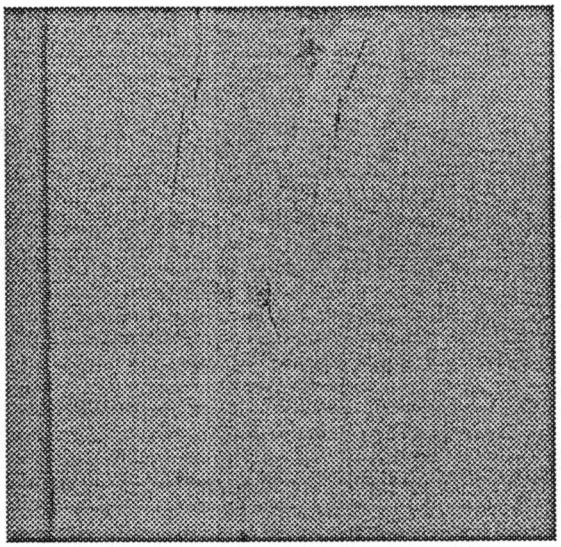
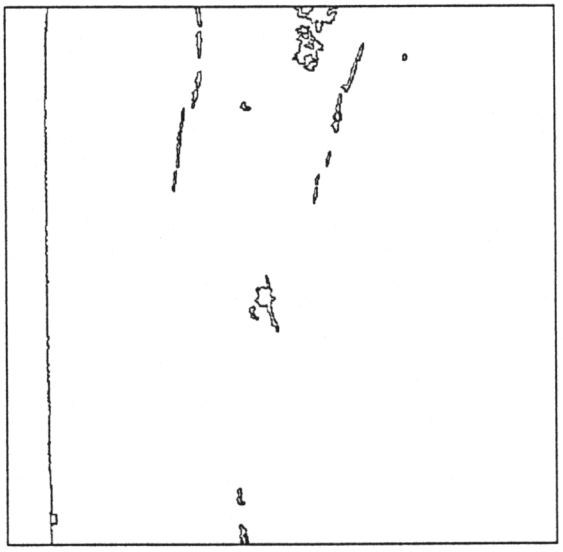


Figure 7.14. Final results of defect recognition.



(a)



(b)

Figure 7.15. A rough cherry board. (a) Original image. (b) Region boundaries after segmentation and region merging.

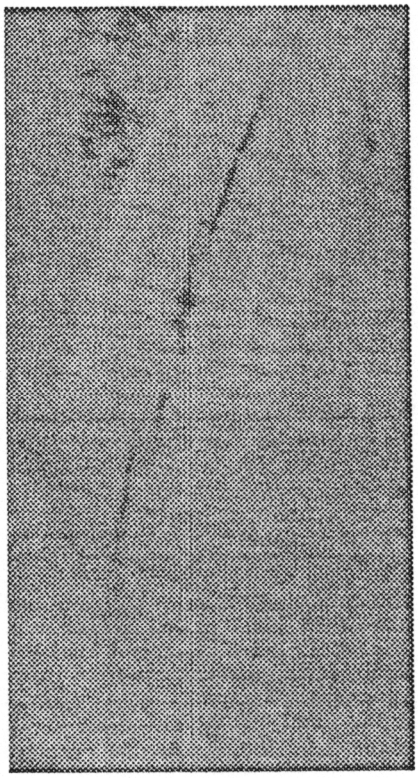


Figure 7.16. A part of the image shown in Figure 7.15. This area contains a knot around a split/check.

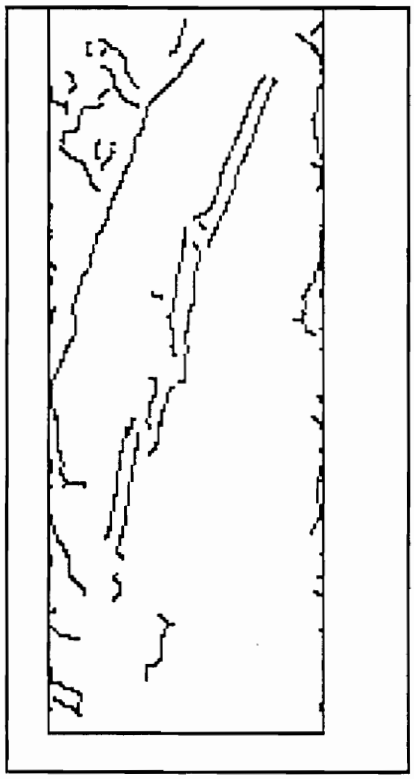


Figure 7.17. Edges obtained by applying the Canny's edge operator to the image of Figure 7.16. The smaller rectangle is the focus window to which the edge operator is applied.

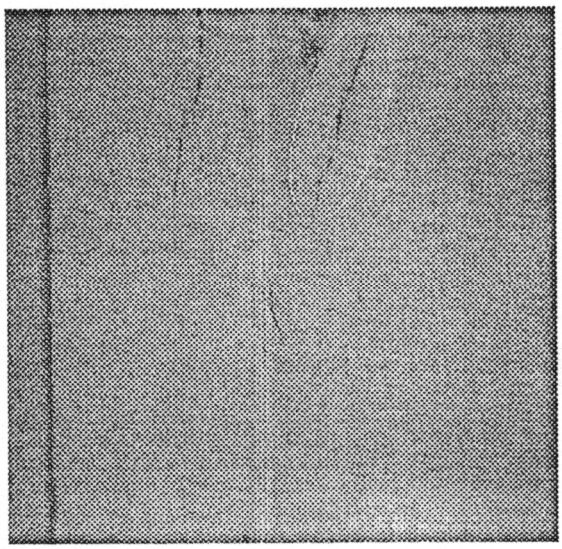
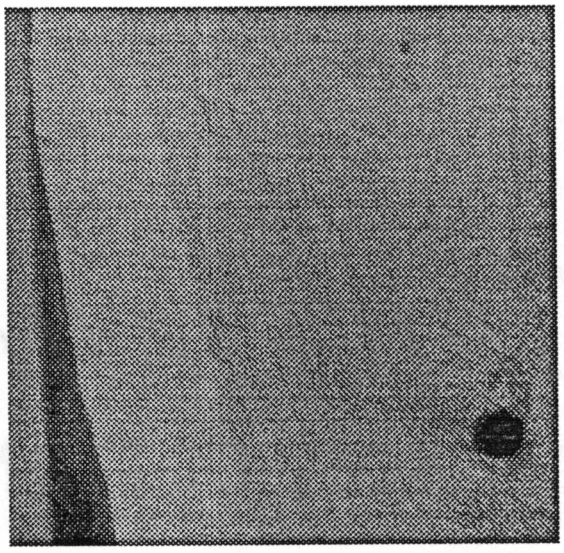
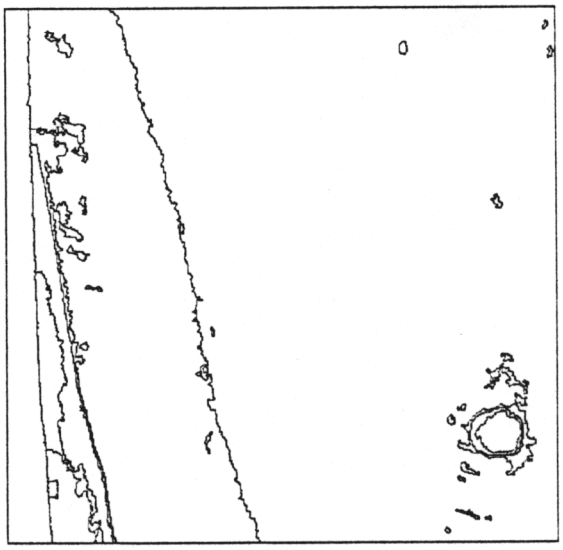


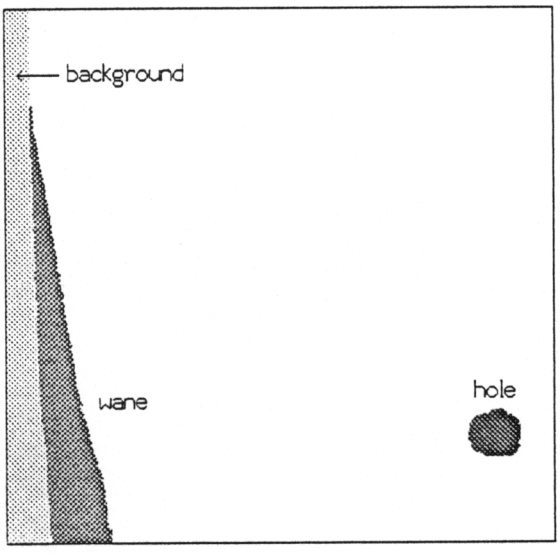
Figure 7.18. An ellipse found by KNOT_BND as an estimated boundary of the knot.



(a)

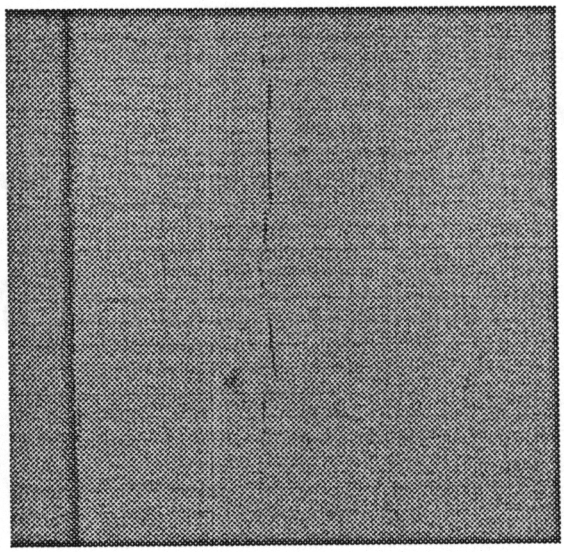


(b)

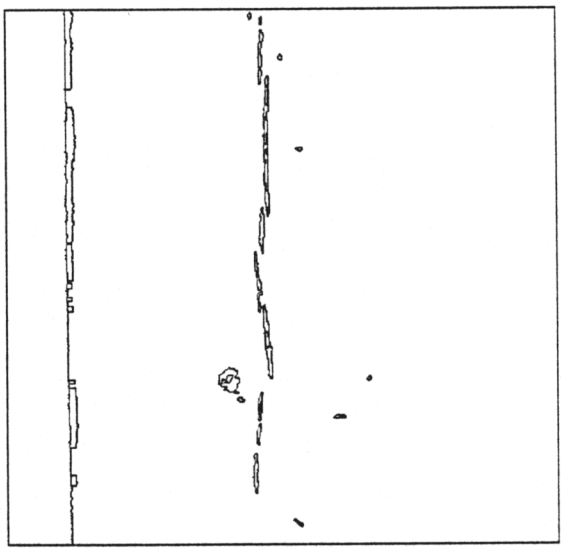


(c)

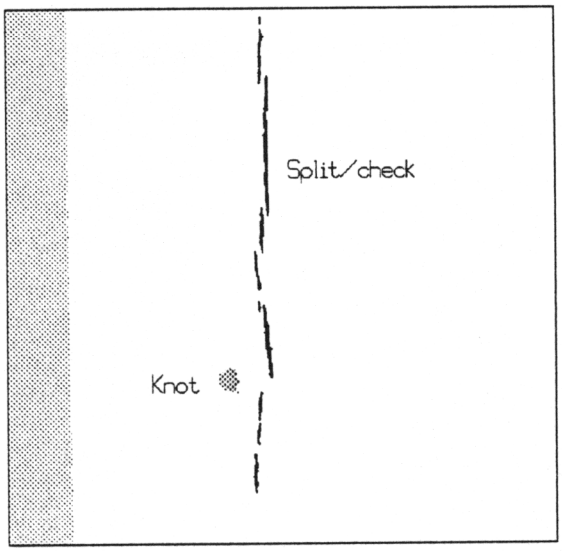
Figure 7.19. A rough cherry board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

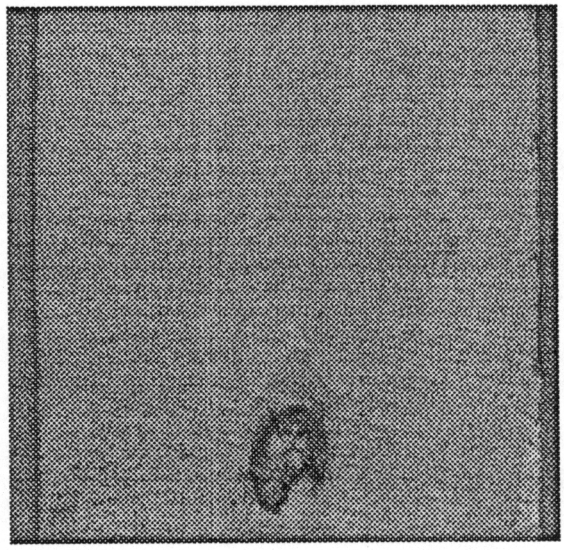


(b)

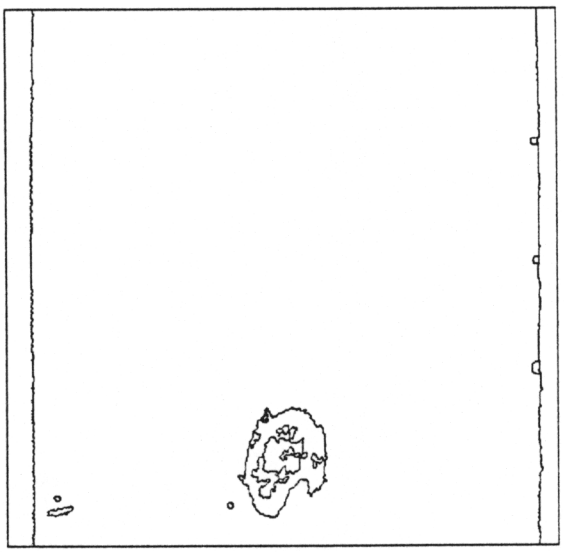


(c)

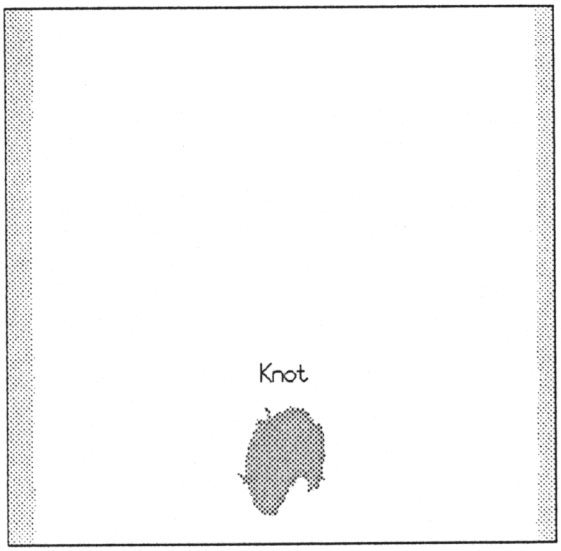
Figure 7.20. A rough cherry board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

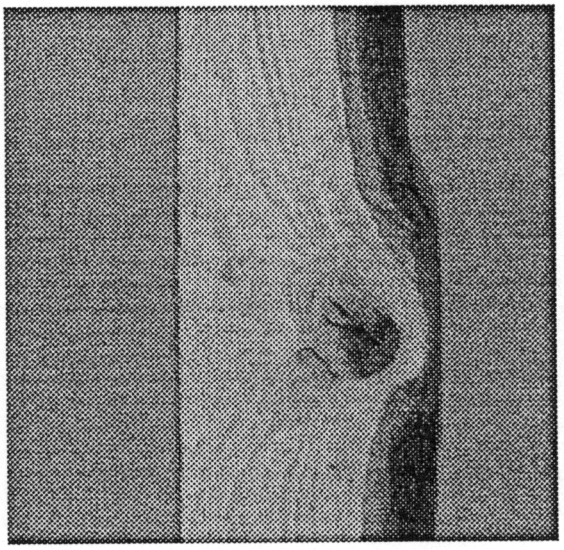


(b)

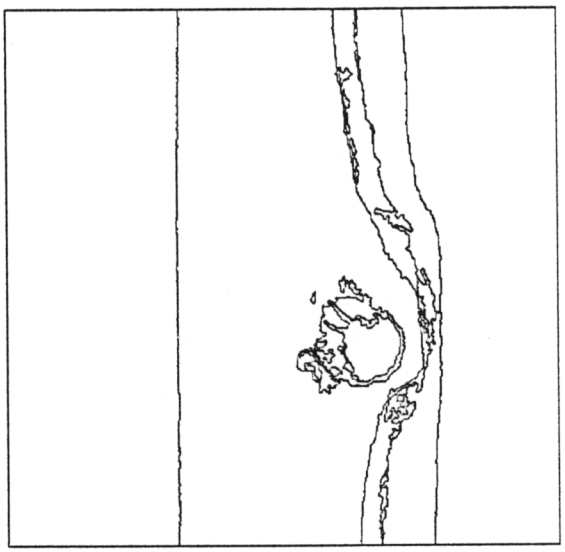


(c)

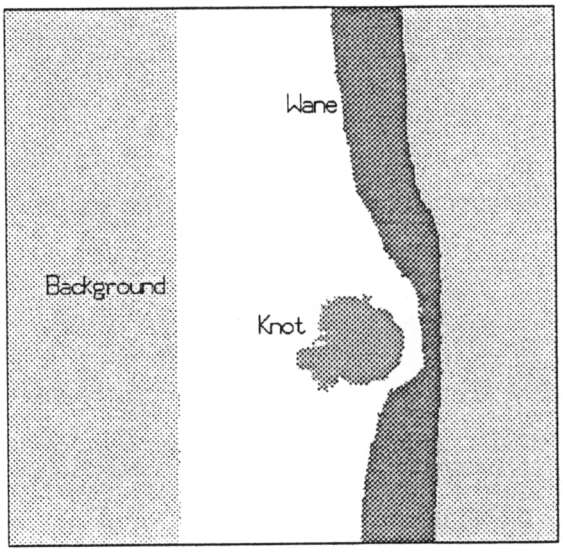
Figure 7.21. A rough cherry board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

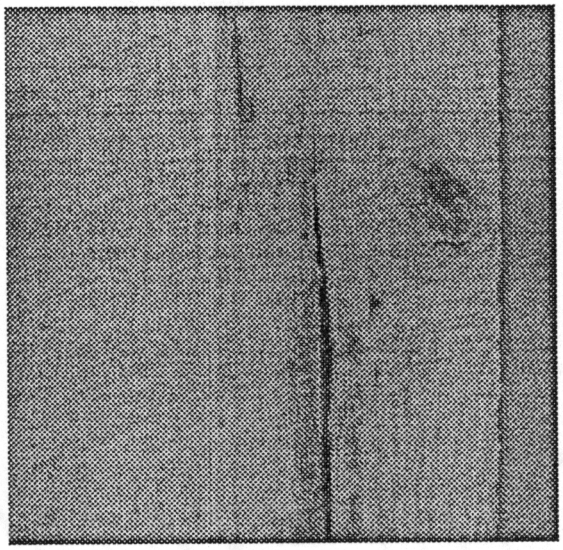


(b)

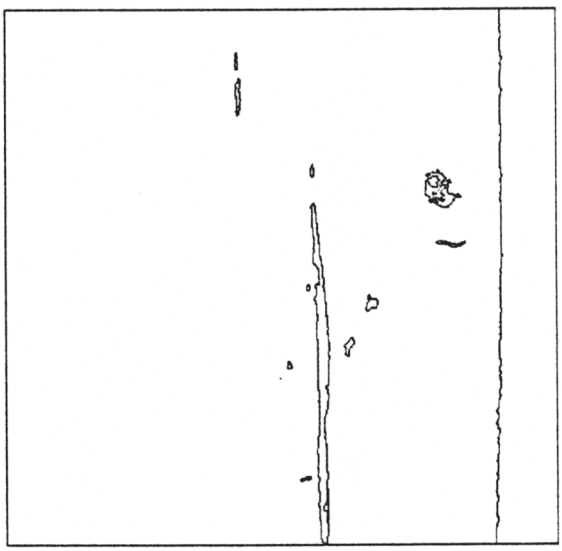


(c)

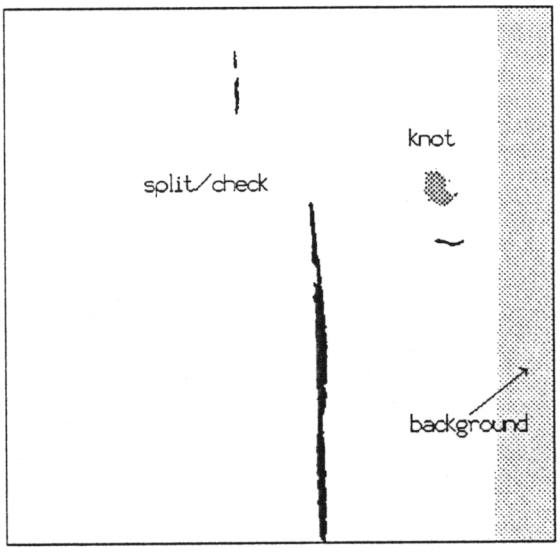
Figure 7.22. A rough oak board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

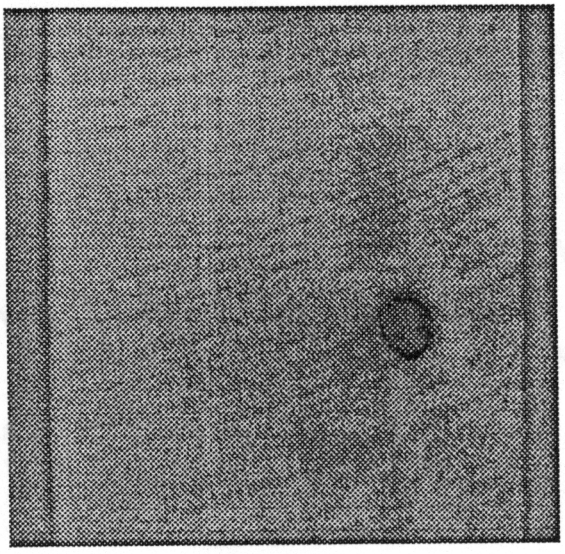


(b)

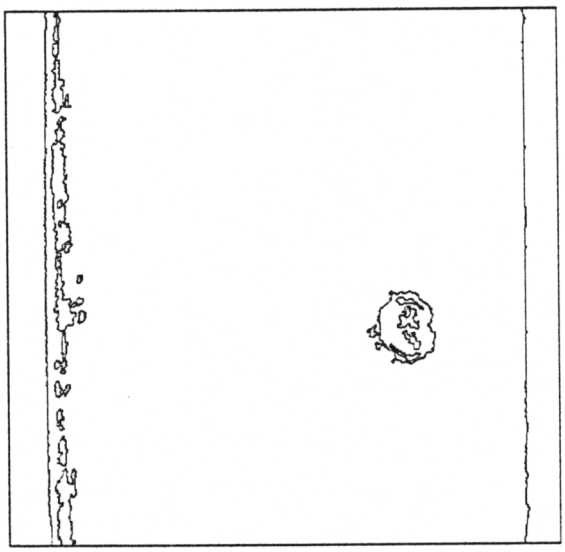


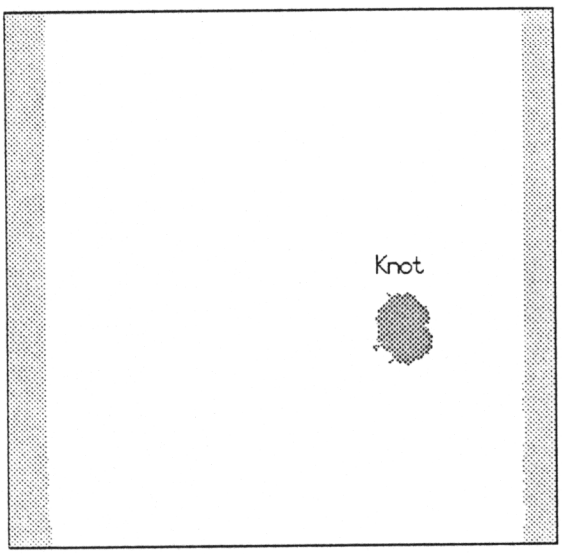
(c)

Figure 7.23. A rough oak board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

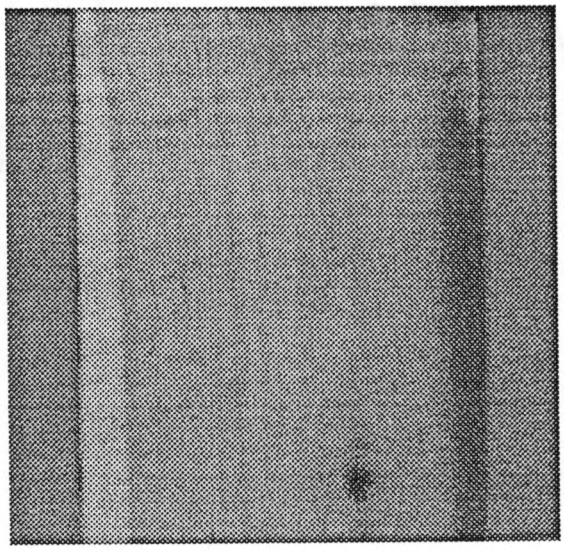




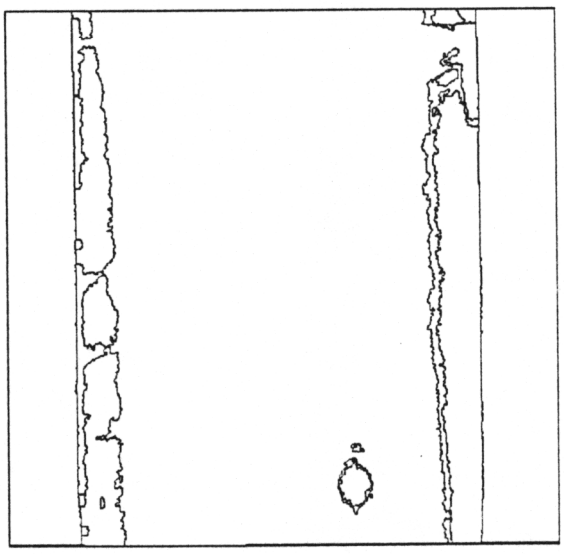
(b)

(c)

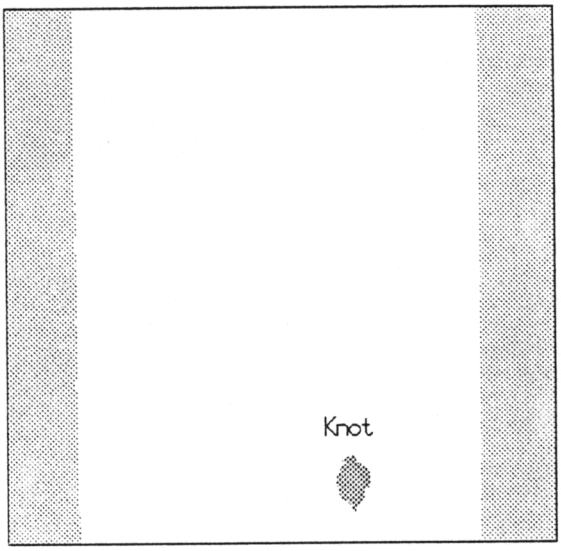
Figure 7.24. A rough oak board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

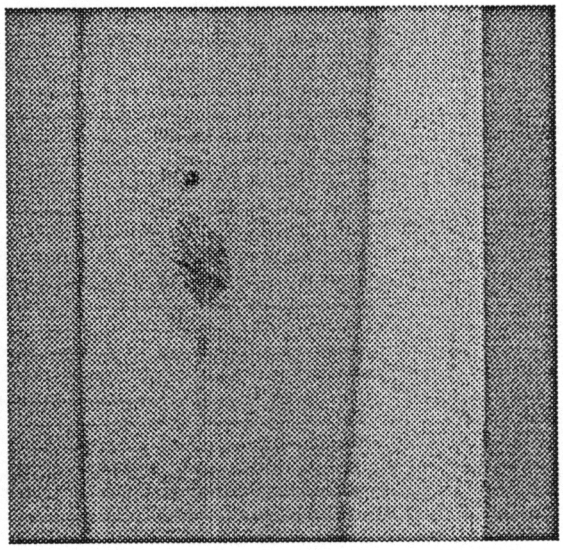


(b)

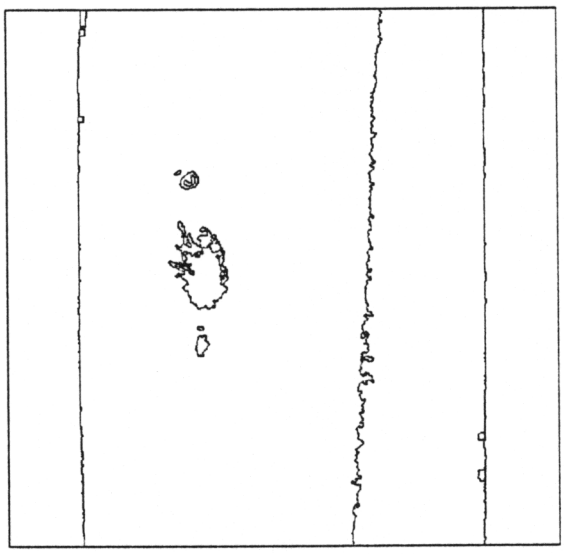


(c)

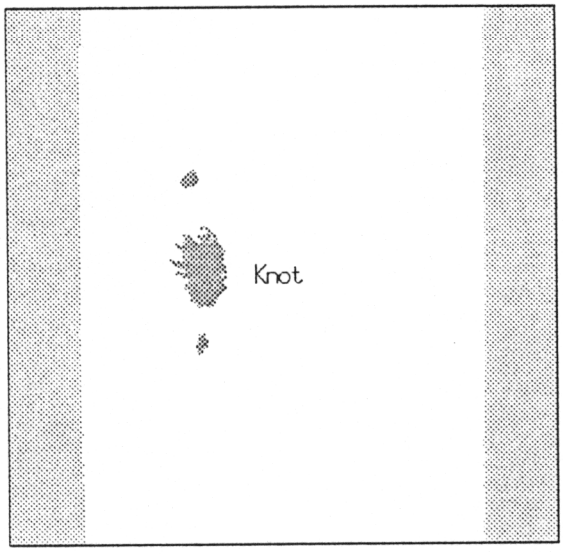
Figure 7.25. A rough yellow poplar board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

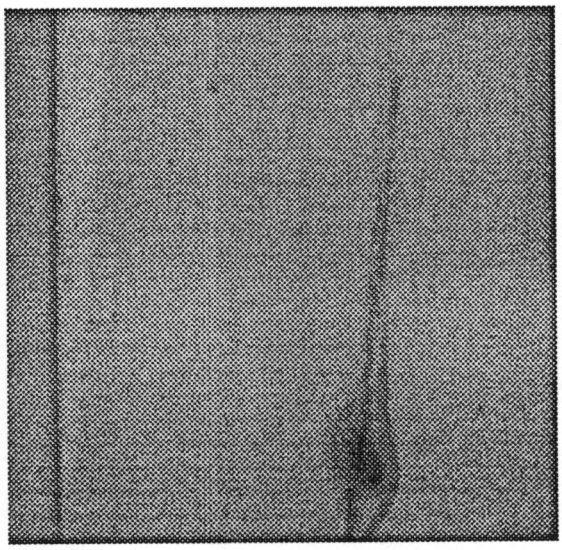


(b)

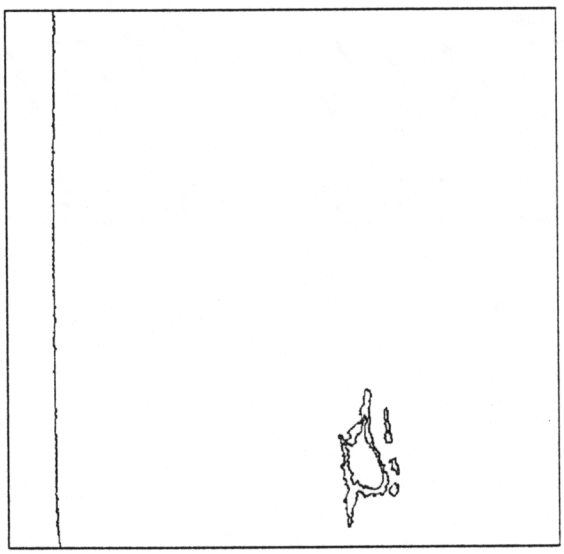


(c)

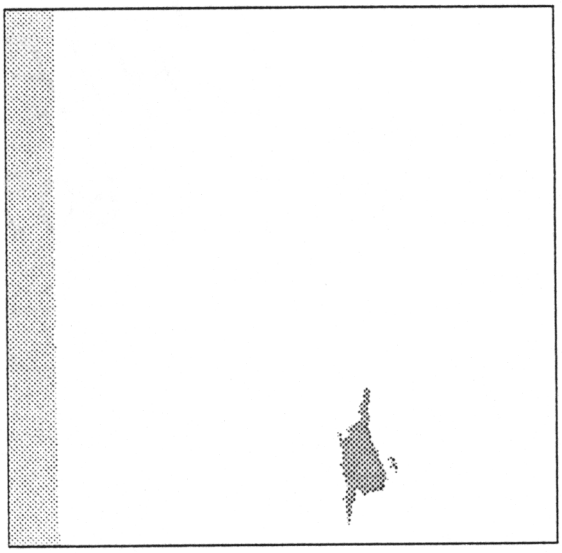
Figure 7.26. A rough yellow poplar board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

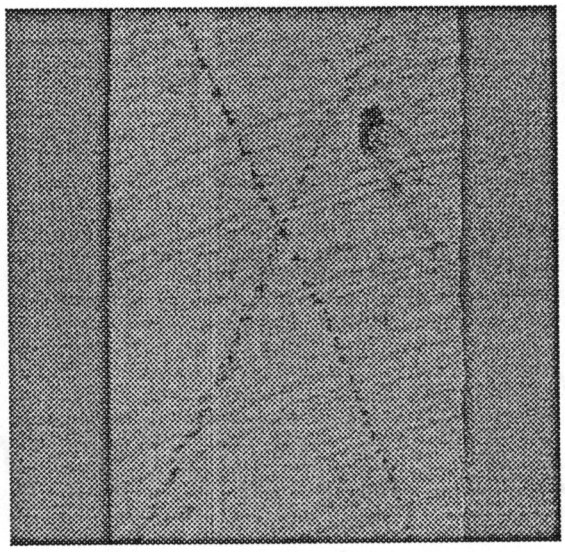


(b)

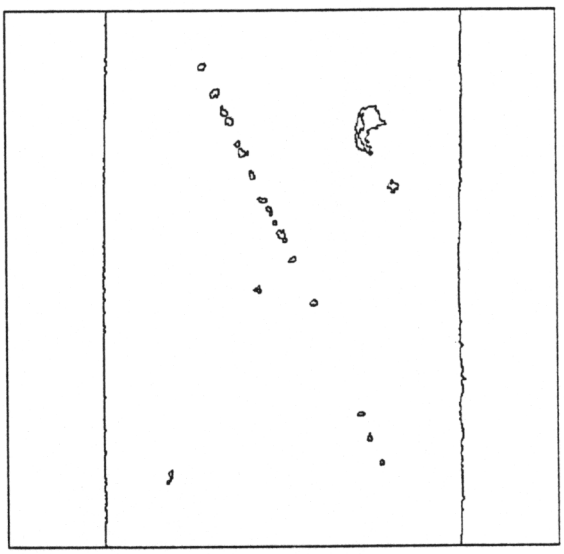


(c)

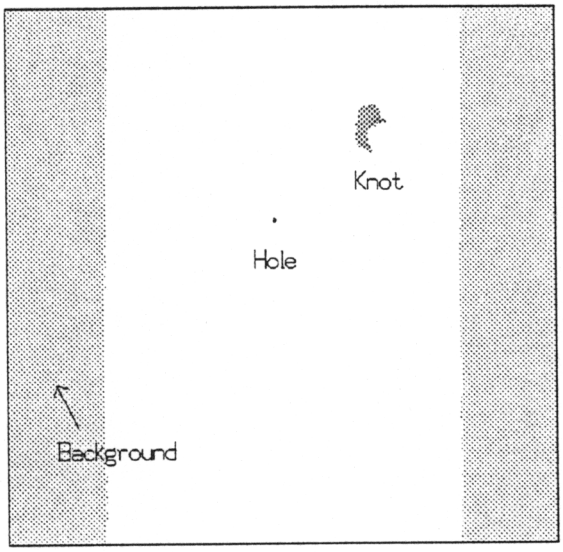
Figure 7.27. A rough yellow poplar board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

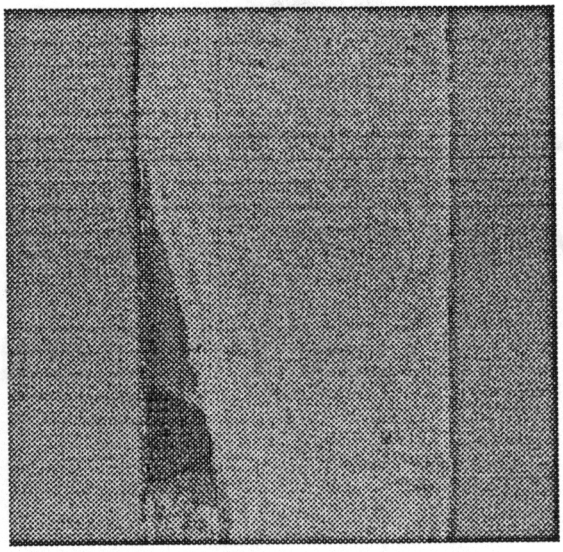


(b)

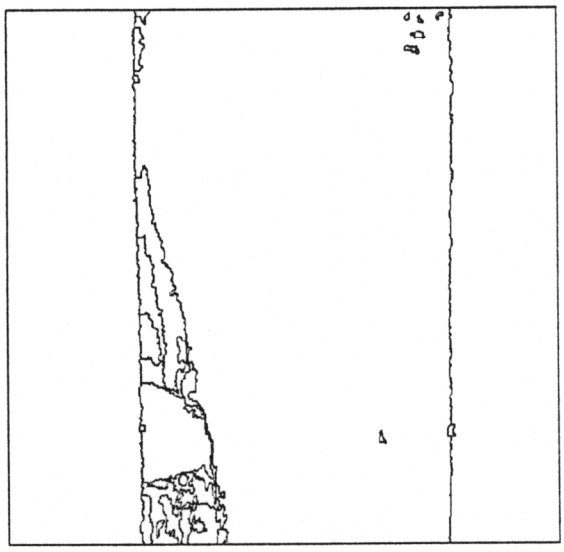


(c)

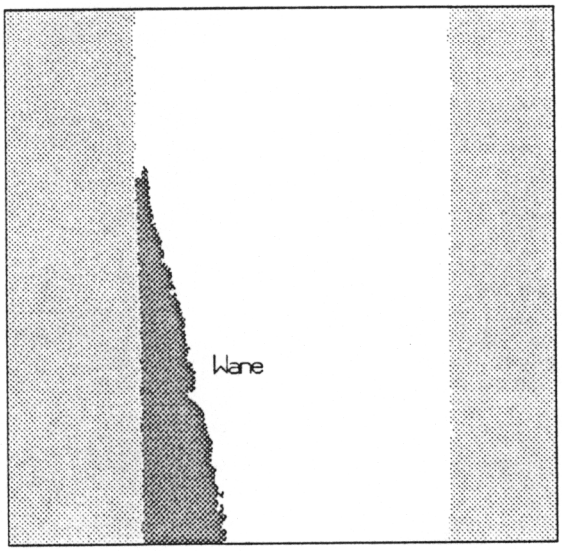
Figure 7.28. A rough maple board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

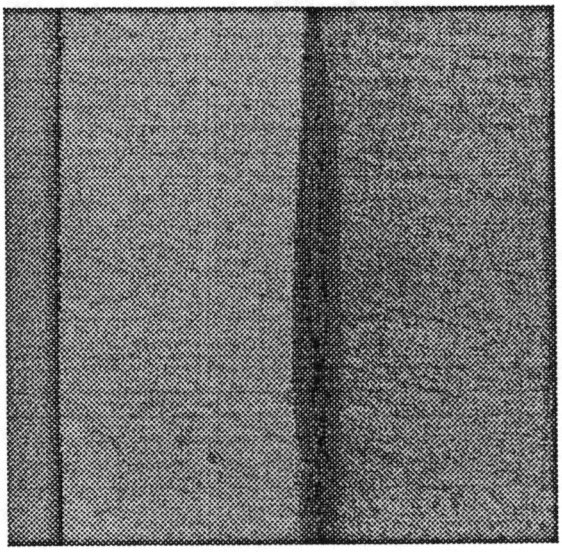


(b)

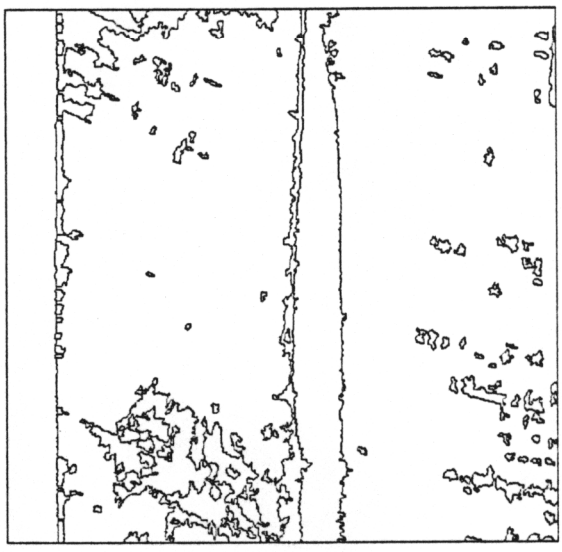


(c)

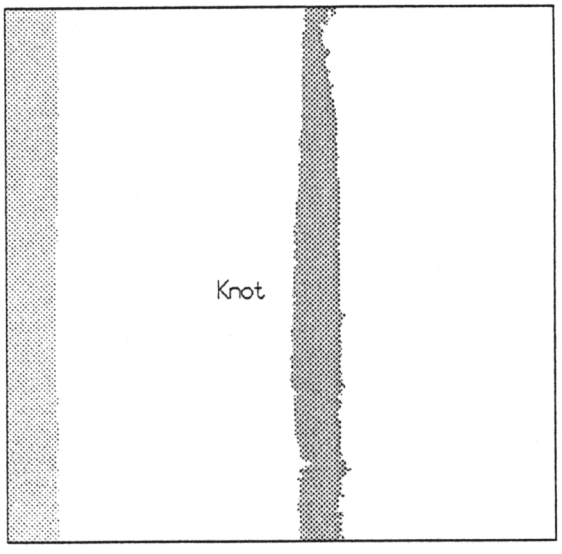
Figure 7.29. A rough maple board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

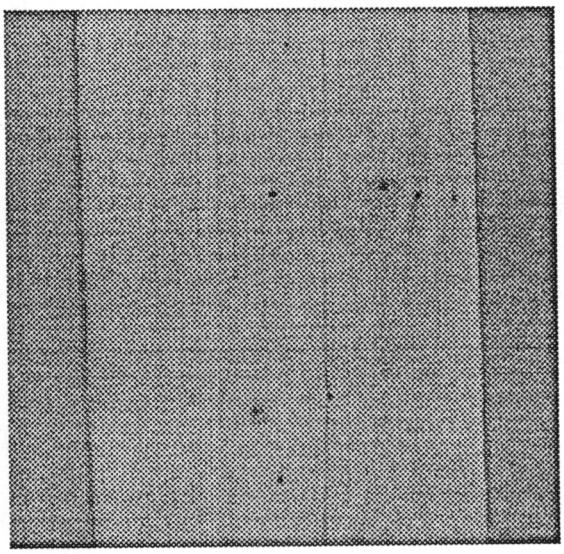


(b)

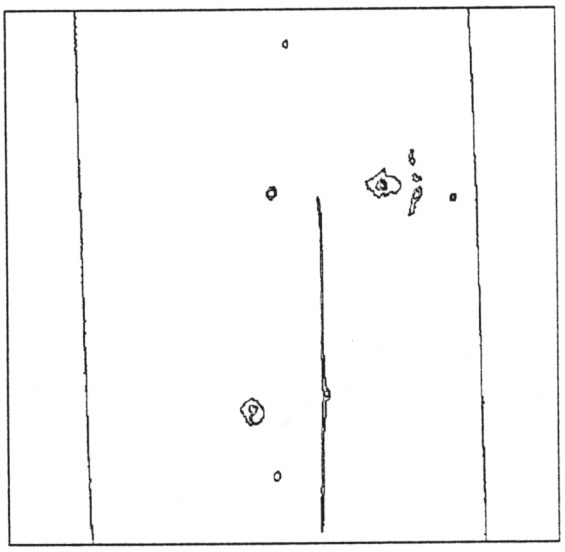


(c)

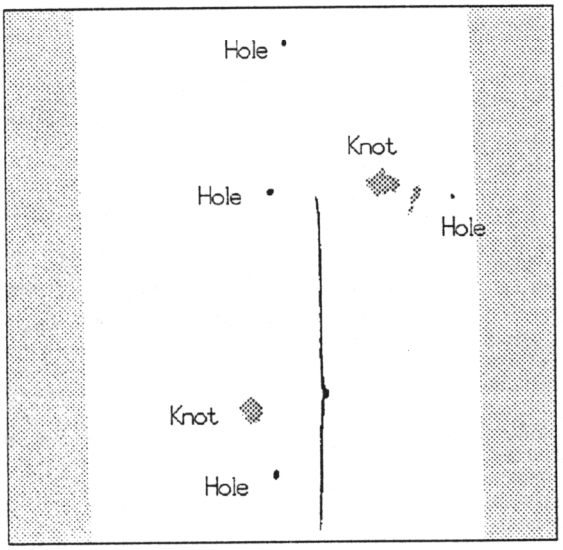
Figure 7.30. A rough maple board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

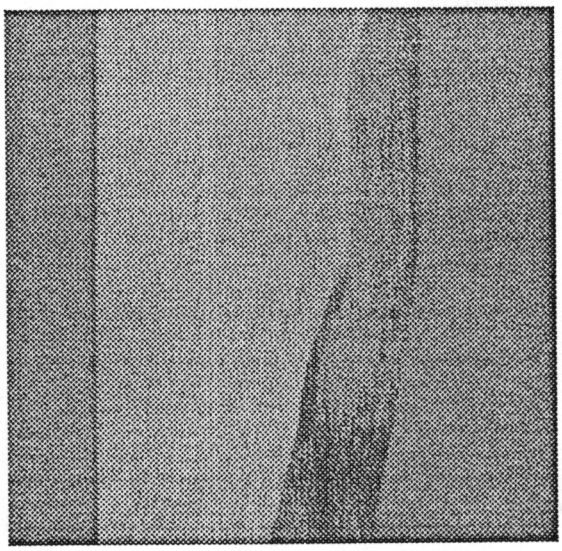


(b)

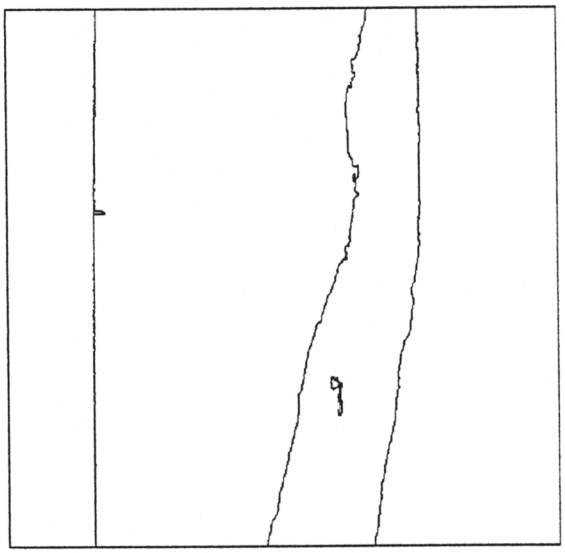


(c)

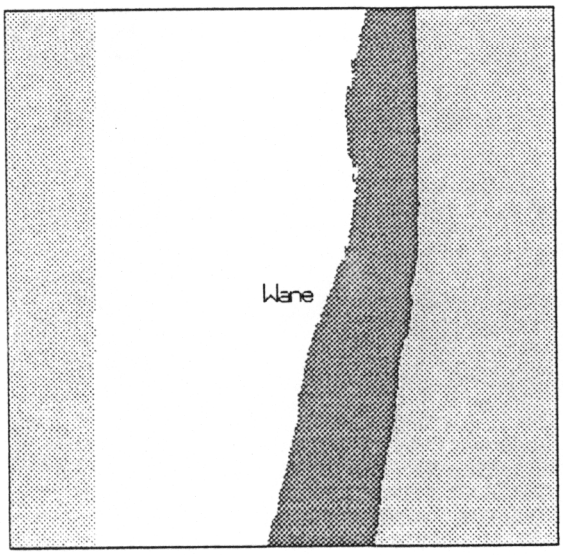
Figure 7.31. A surfaced maple board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

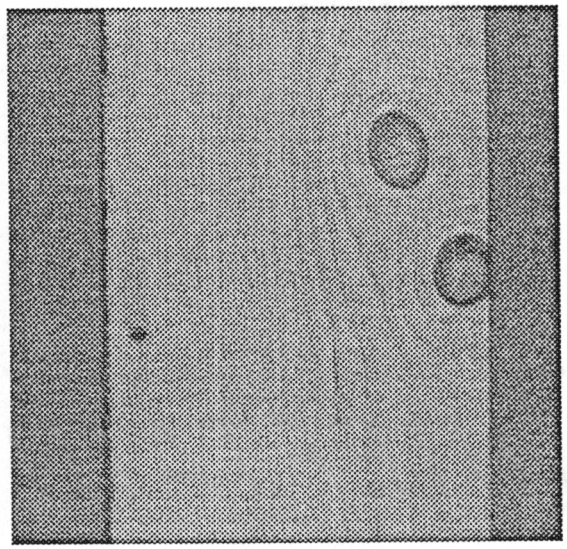


(b)

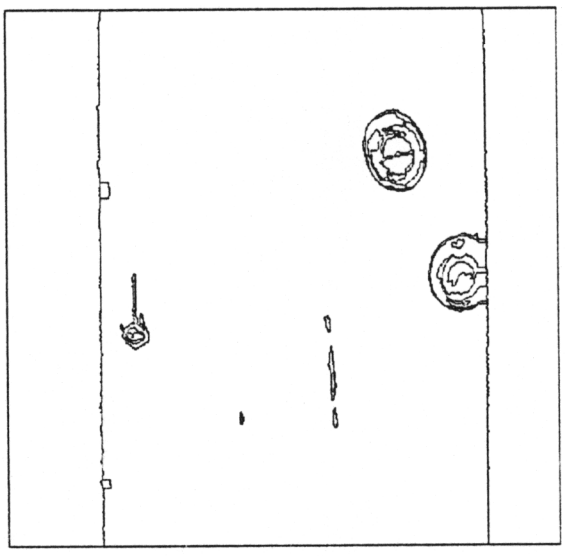


(c)

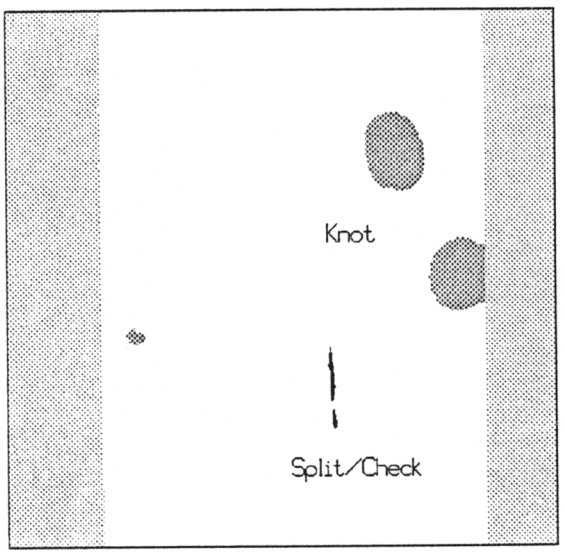
Figure 7.32. A surfaced oak board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)

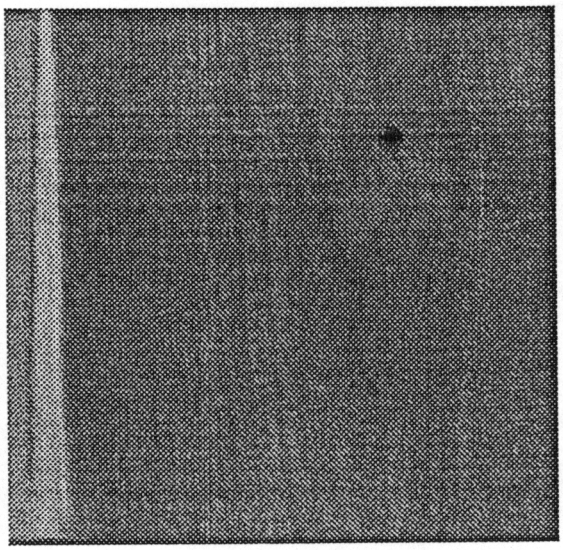


(b)

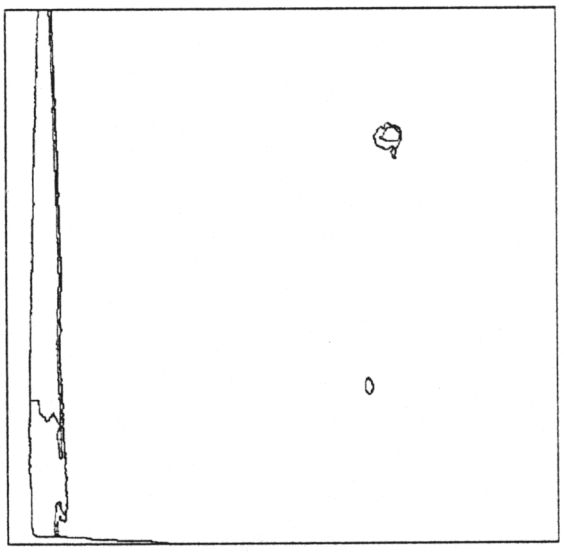


(c)

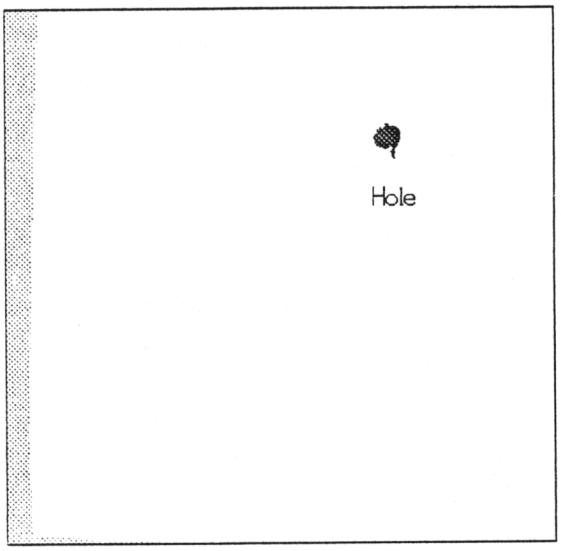
Figure 7.33. A surfaced pine board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.



(a)



(b)



(c)

Figure 7.34. A surfaced walnut board. (a) Original image. (b) Region boundaries after segmentation and region merging. (c) Results of defect recognition.

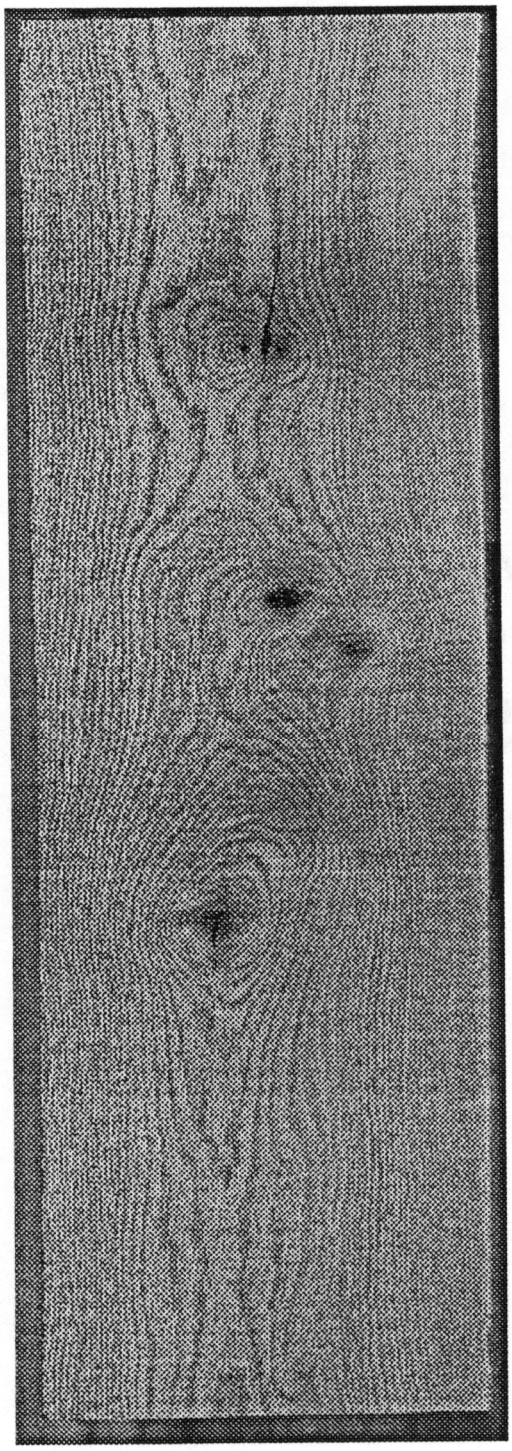


Figure 7.35. A 1536x512 image of a 4-foot long surfaced oak board.

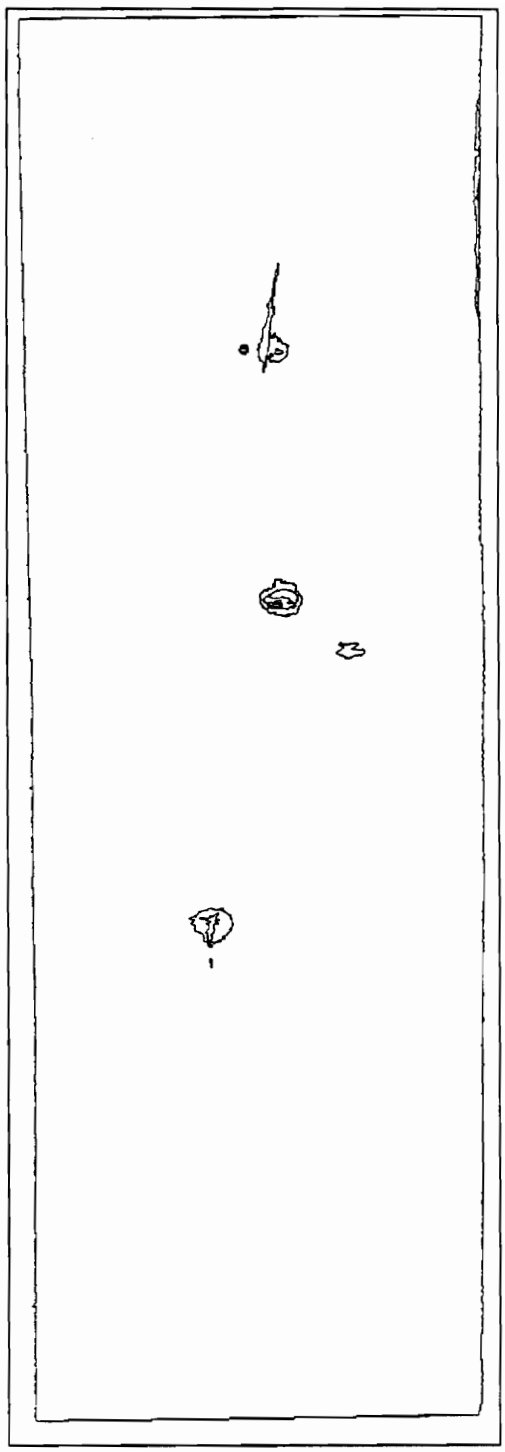


Figure 7.36. Boundaries of regions obtained after applying segmentation and region merging operation to the image of Figure 7.35.

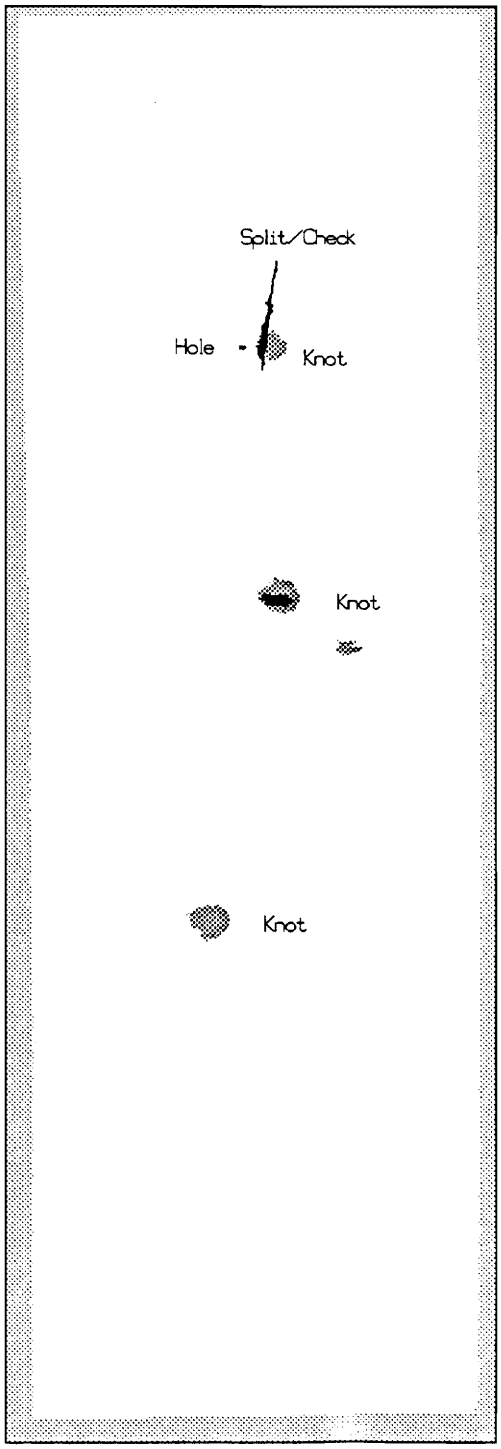


Figure 7.37. Defects recognized by the high-level recognition module for the image of Figure 7.35.

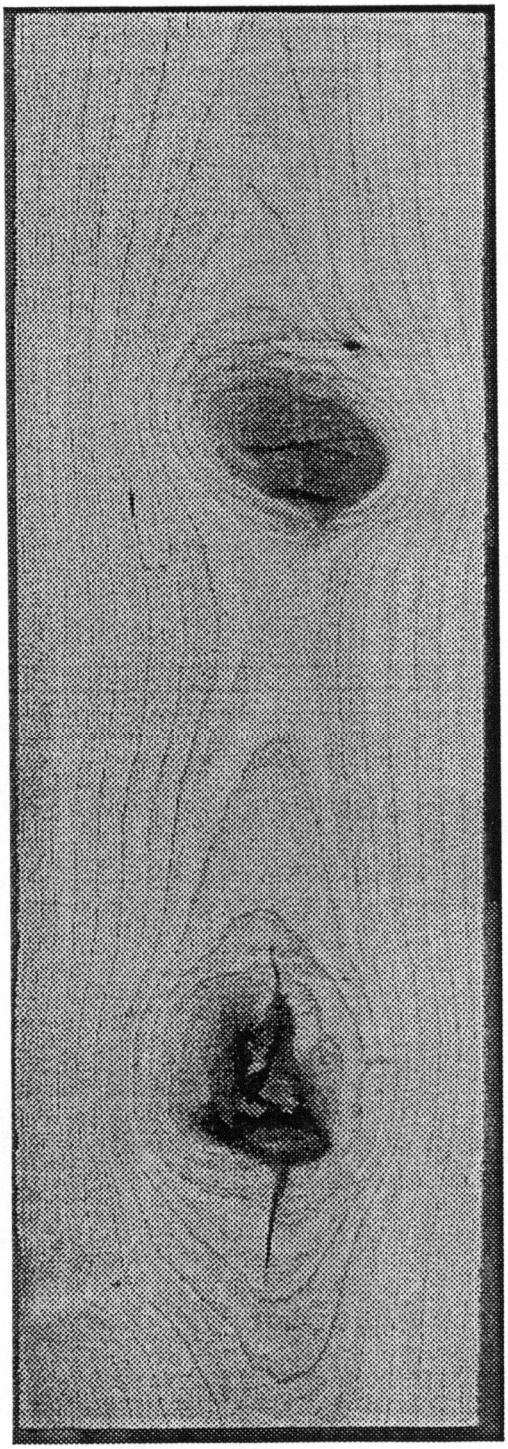


Figure 7.38. A 1536x512 image of a 4-foot long surfaced yellow poplar board.

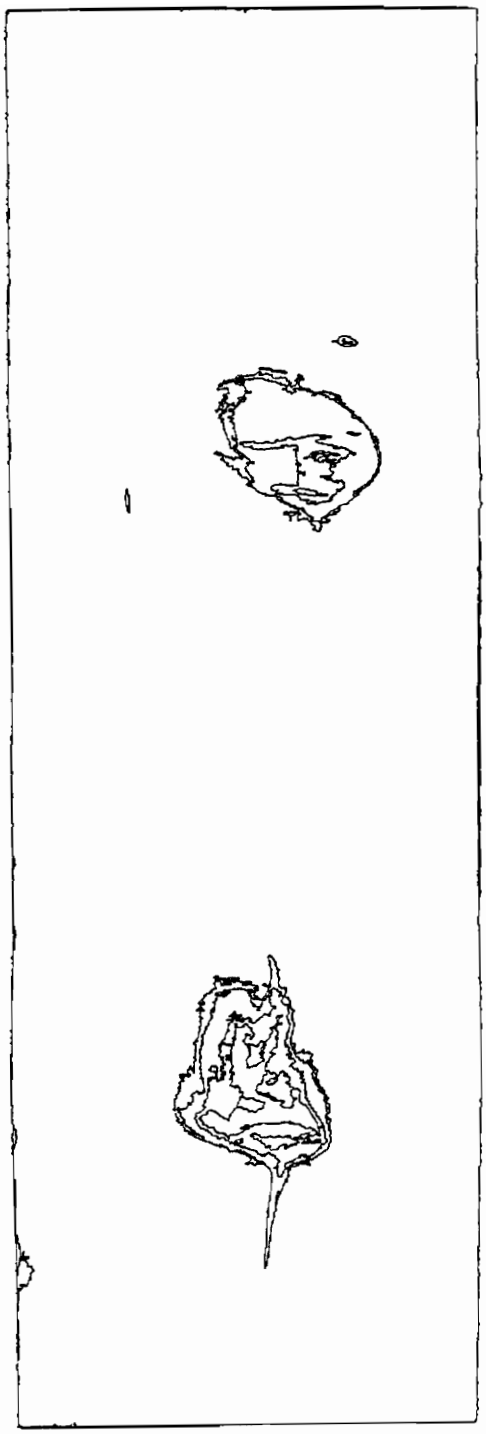


Figure 7.39. Boundaries of regions obtained after applying segmentation and region merging operation to the image of Figure 7.38.

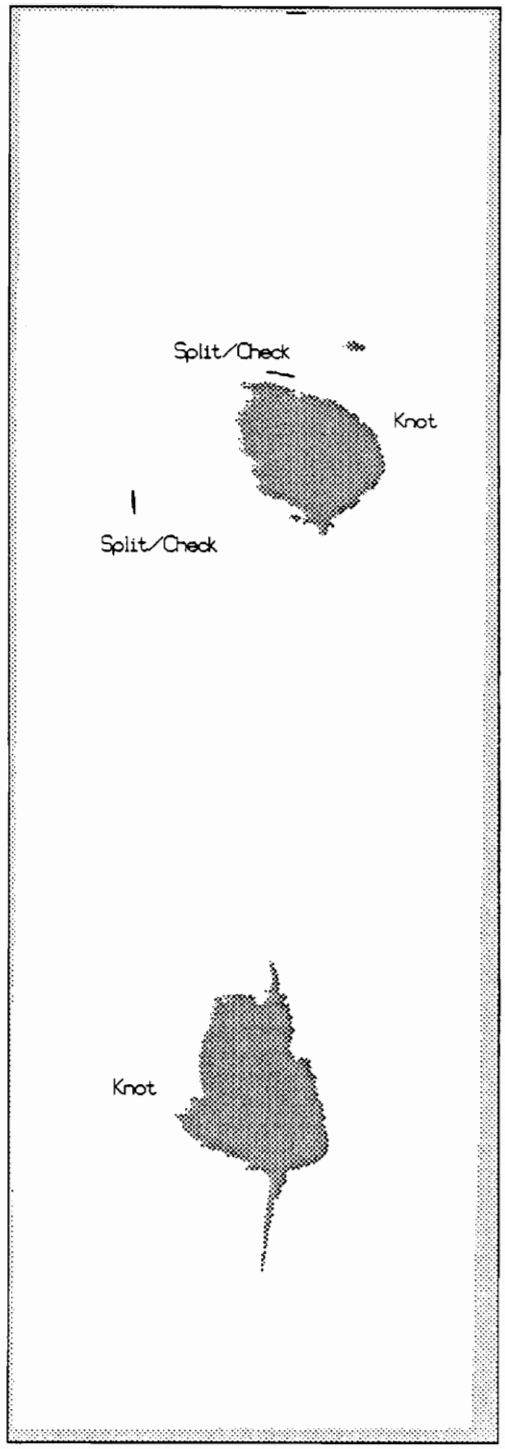


Figure 7.40. Defects recognized by the high-level recognition module for the image of Figure 7.38.

Chapter 8

CONCLUSIONS

Most vision systems that are currently in use in industry were developed with one specific task in mind. Due to the requirement of real-time operation, these were typically implemented in special purpose hardware for performing specific operations. Hence, they are typically inflexible in the sense that they are not easily adapted to other applications. However, current trends in computer technology suggest that low-cost general-purpose computers will be available in very near future that are fast enough to meet the speed requirements of many industrial inspection problems. If this low-cost computing power is to be effectively utilized on industrial inspection problems, more general-purpose vision systems must be developed, vision systems that can be easily adapted to a variety of applications. Unfortunately, little research has gone into creating such general-purpose industrial inspection systems.

In this dissertation, a general vision system framework has been developed that can be easily adapted to a variety of *industrial web inspection* problems. The objective is to automatically locate and identify "defects" on the surface of the material being inspected. This framework has the following desirable characteristics:

- The vision system framework incorporates a number of methods that should help assure robust operation.

- The vision system framework is flexible, so that it can be easily adapted to a number of different web inspection problems. Adapting the system to a particular problem domain only requires that new defect detection procedures be created and that new spatial constraints be formulated.

- The design approach used in creating the framework attempts to minimize the computational complexity of the analysis so that real-time operation at industrial speeds will be possible.

To assure robustness of the vision system, several methods were employed. 1) A combination of bottom-up (data-driven) control and top-down (model-driven) control were used to perform scene analysis. This combined or mixed control strategy is necessary to make the system robust and flexible. 2) Hierarchical object models that consider defects at various levels of representation were used to recognize the defects. 3) Methods for uncertainty management were used in the vision system to deal with inherent uncertainty in all stages of the visual processing.

To make the vision system flexible, a modular knowledge structure was used. The Blackboard framework employed provides this modularity because there is no direct communication among knowledge sources. Each knowledge source can communicate with other KSs only through the blackboard.

To minimize the computational complexity of system operation, several methods were used. 1) For segmenting images of material to be inspected, a histogram-based multi-thresholding was used due to its computational simplicity. 2) The region merging and recognition algorithms were also created so as to minimize the computational complexity of the analysis. One way used to do this was to employ a focus of attention mechanism for performing scene analysis as efficiently as possible. This focus of attention mechanism extracts with minimal computation possible candidate areas that might belong to a defect category of interest. 3) Knowledge about defects is partitioned according to defect type. This allows concurrent parallel processing by running each defect detection procedure on one processor, thus speeding up the system operation.

On the basis of the proposed vision system framework, a computer vision system for automated lumber grading has been developed. The purpose of the vision system for automated lumber grading is to locate and identify grading defects on rough hardwood lumber in a species independent manner. This problem seems to represent one of the more difficult and complex web inspection problems. No vision system has been developed to tackle this problem. The main reason for this seems to be that there are a lot of factors causing normal surface discolorations, discolorations that must be differentiated from areas that truly contain a defect. Also, some defect types are difficult to recognize due to irregular characteristics of their shapes. Hence if one can use the previously described framework to develop a vision system for grading rough lumber, this framework should be versatile enough to work on a variety of other web inspection problems. In fact, the methodologies used in the vision system for grading lumber would seem applicable to a number of other web inspection problems as well.

The system has been tested on approximately 100 boards from several different species. Parameters that are not adaptively determined during the operation were fixed throughout all the experiments. The vision system has been found to be robust. The vision system using the neural network approach performed best in overall accuracy among the three approaches in the label verification step of the defect detection procedures. An experimental comparison of these three approaches was presented.

During this research, a number of relatively general-purpose computer vision methodologies were developed. These include the following:

1. *A focus of attention mechanism using fuzzy logic in the high-level recognition stage.* Fuzzy logic is appropriate for roughly representing a concept of an object using natural language, and it is computationally simple.

2. *A robust multi-thresholding segmentation method.* The thresholds are adaptively and automatically determined based on the valleys and inflection points of the gray-level histogram of the input image.

3. *Region merging operation using the Dempster-Shafer (D-S) theory.* For the region merging step, Dempster's rule of combination is used to combine the results of multiple tests, each of which evaluates the validity of merging two adjacent regions based on this test's own criterion.

Also, a new learning algorithm for multi-layer neural networks was developed that is much faster and more stable in convergence than the standard back-propagation learning algorithm. This new algorithm uses automatic initializations of weights in the network when convergence to a solution is very slow.

However, further research remains to be done. It is suggested that the future research should be directed toward the following.

• *Refine the capability of the segmentation module.* The current segmentation method has been shown to be robust in dealing with problems where defects manifest themselves as being different from normal surface in gray shade. However, there are applications where color differences are important, differences that do not manifest themselves in gray shade. To handle such cases, a robust color segmentation method needs to be developed.

• *Augment the capability of re-segmentation.* The current recognition module has some specialized top-down methods within each defect detection procedure. When a region (or DEFECT_OBJECT) is considered to have some confidence of being a defect type, top-down methods search for additional evidence or features supporting this labeling conjecture. However, the current implementation lacks a re-segmenting

capability. There are instances when a top-down method might want to refine a region or a DEFECT_OBJECT boundary. In such cases a re-segmentation operator needs to be available for use by the top-down procedure.

• *Generalize the neural network decision rule to approximate Bayesian minimum risk decision rule.* The decision rule implemented in the neural network label verification is an approximation of the Bayesian minimum-error-rate classification. With minor modifications, this decision rule can be generalized to approximate Bayesian minimum risk decision rule.

• *Optimize the features to be extracted in the low-level module.* Region features extracted in the low-level module could be determined such that the set of selected features is an optimal set of features for differentiating defect classes.

• *Implement for real-time operation.* While every effort has been made to reduce the computational complexity of the analysis algorithms, effort should put forth to determine the best hardware architecture for implementing these algorithms so that real-time operation at high industrial flow rates can be achieved.

REFERENCES

[BAL82] D.H. Ballard and C.M. Brown, *Computer Vision*, NJ: Prentice Hall, 1982.

[BAR81a] J. Barnett, "Computational methods for a mathematical theory of evidence," *Proc. International Joint Conference on Artificial Intelligence*, pp.868-875, 1981.

[BAR81b] H.G. Barrow and J.M. Tenenbaum, "Computational vision," *Proc. IEEE*, vol.69, no.5, pp.572-595, 1981

[BAR88] S.L. Bartlett, P.J. Besl, C.L. Cole, R. Jain, D. Mukherjee, and K.D. Skifstad, "Automatic solder joint inspection," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.10, no.1, pp.31-43, 1988.

[BIN82] T.O. Binford, "Survey of model-based image analysis systems," *International Journal of Robotics Research*, vol.1, no.3, pp.18-64, 1982.

[BOU85] S. Boukharouba, J.M. Rebordao, and P.L. Wendel, "An amplitude segmentation method based on the distribution function of an image," *Comput. Vision, Graphics, Image Processing*, vol.29, pp.47-59, 1985.

[BOU90] H. Bourlard and C.J. Wellenkens, "Links between Markov models and multilayer perceptrons," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.12 , no.12, pp.1167-1178, 1990.

[BRI70] C.R. Brice and C.L. Fennema, "Scene analysis using regions," *Artificial Intelligence*, vol.1, 205-226, 1970.

[CAN86] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.8 , no.6, pp.679-698, 1986.

[CAR85] M.J. Carlotto, "Histogram analysis using a scale-space approach," *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp.334-340, 1985.

[CHI86] R.T. Chin, "Algorithms and techniques for automated visual inspection," in *Handbook of Pattern Recognition*, T.Y. Young and K.S. Fu, Eds. Academic Press, 1986, pp.587-612.

[CHI88] R.T. Chin, "SURVEY: automated visual inspection: 1981 to 1987," *Computer Vision, Graphics, and Image Processing*, vol.41, pp.346-381, 1988.

[CHO72] C.K. Chow and T. Kaneko, "Automatic boundary detection of the left ventricle from cineangiograms," *Computers and Biomedical Research*, vol.5, 1972, pp.388-510.

[CLO71] M. Clowes, "On seeing things," *Artificial Intelligence*, vol.2, no.1, pp.79-112, 1971.

[CON83] R.W. Conners, C.W. McMillin, K. Lin, and R.E. Vasquez-Espinosa, "Identifying and locating surface defect in wood: part of an automated lumber processing system," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol. PAMI-5, no.6, pp.573-583, 1983.

[CON87] R.W. Conners, "The need for a quantitative model of human preattentive vision," *Proc. SPIE, vol.786, Applications of Artificial Intelligence*, pp.211-220, 1987.

[CON89] R.W. Conners, C.T. Ng, T.H. Cho, and C.W. McMillin, "Computer vision system for locating and identifying defects in hardwood lumber," *SPIE Vol. 1095, Applications of Artificial Intelligence VII*, pp.48-63, 1989.

[CYB89] G. Cybenko, "Approximation by superposition of a sigmoid function," *Mathematics of Controls, Signals, and Systems*, vol.2, pp.303-314, 1989.

[DAR88] A.M. Darwish and A.K. Jain, "Rule based approach for visual pattern inspection," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol. PAMI-10, no.1, pp.56-68, 1988.

[DEV82] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Englewood Cliffs, NJ: Prentice Hall, 1982.

[DRA88] B.A. Draper, R.T. Collins, J. Brolio, A.R. Hanson, and E.M. Riseman, "Issues in the development of a blackboard-based schema system for image understanding," in *Blackboard Systems*, R. Engelmore and T. Morgan, Eds. MA: Addison-Wesley, 1984, pp.133-146.

[DUD73] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[ENG88] R. Engelmore and T. Morgan, Eds., *Blackboard Systems*, MA: Addison-Wesley, 1988.

[ERM80] L.D. Erman, F. Hayes-Roth, V.R. Lesser and D.R. Reddy, "The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty," *ACM Computing Surveys*, vol.12, no.2, pp.213-253, 1980.

[FAH88] S.E. Fahlman, "An empirical study of learning speed in back-propagation," Tech. Rep. CMU-CS-88-162, Comp. Sci. Dept., Carnegie-Mellon Univ., June 1988.

[FUN85] B.V. Funt and E.C. Bryant, "A computer vision system that analyses CT-scans of sawlogs," *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp.175-177, 1985.

[FUN89] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol.2, pp.183-192, 1989.

[GOR84] J. Gordon and E.H. Shortliffe, "The Dempster-Shafer theory of evidence," in *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, B.G. Buchanan and E.H. Shortliffe, Eds. MA: Addison-Wesley, 1984, pp.133-146.

[HAN78] A.R. Hanson and E.M. Riseman, "A Computer System for interpreting Scenes," in *Computer Vision Systems*, A.R. Hanson and E.M. Riseman, Eds. NY: Academic Press, 1978, pp.303-334.

[HAN88] A.R. Hanson and E.M. Riseman, "The VISIONS image understanding system," in *Advances in Computer Vision, Vol.1*, C. Brown, Ed. NJ: Lawrence Erlbaum Associates, 1988, pp.1-114.

[HAR79] R.M. Haralick, "Statistical and structural approaches to texture," Proc. IEEE, vol.67, no. 5, pp.786-804, 1979.

[HAR85] R.M. Haralick and L.G. Shapiro, "SURVEY Image segmentation techniques," *CVGIP* 29, pp.100-132, 1985.

[HAR86] R.M. Haralick, "Statistical image texture analysis," in *Handbook of Pattern Recognition*, T.Y. Young and K.S. Fu, Eds. Academic Press, 1986, pp.247-279.

[HAR88] Y. Hara, H. Doi, K. Karasaki, and T. Iida, "A system for PCB automated inspection using fluorescent light," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.10, no.1, pp.69-78, 1988.

[HEN88] S.J. Henkind and M. Harrison, "An analysis of four uncertainty calculi," *IEEE Trans. Systems, Man, and Cybernetics*, vol.18, no.5, pp.700-714, 1988.

[HOR89] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol.2, pp.359-366, 1989.

[HUA87] W.Y. Huang and R.P. Lippman, "Comparisons between neural net and conventional classifiers," *Proc. IEEE First Int. Conf. Neural Networks*, vol.IV, Piscataway, Nj: IEEE, pp.485-493, 1987.

[HUF71] D. Hufman, "Impossible objects as nonsense sentences," in *Machine Intelligence*, B. Meltzer and D. Michie (Eds.), Edinburgh University Press, Edinburgh, Scotland, pp.115-134, 1971.

[HWA86] V. Hwang, L.S. Davis, and T. Matsuyama, "Hypothesis interpretation in image understanding systems," *Computer Vision, Graphics, and Image Processing*, vol.36, pp.321-371, 1986.

[JAC88] R.A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol.1, pp.295-307, 1988.

[KIM90] C.W. Kim and A.J. Koivo, "Hierarchical classification of surfaced defects on dusty wooden boards," *Proc. 10th International Conference on Pattern Recognition*, pp.775-779, 1990.

[KLI88] P. Klinkhachorn, J.P. Franklin, C.W. McMillin, R.W. Conners, and H.A. Huber, "Automated computer grading of hardwood lumber," *Forest Products Journal*, vol.38, no.3, pp.67-69, 1988

[KLI90] E. Kline, *Personal Communication*, 1990.

[KOH81] R. Kohler, "A segmentation system based on thresholding," *Comput. Vision, Graphics, Image Processing*, vol.15, pp.319-338, 1981.

[KOI86] A.J. Koivo and C.W. Kim, "Classification of surface defects on wood boards," *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Oct. 1986, pp.1431-1436.

[KOI88] A.J. Koivo and C.W. Kim, "Parameter estimation of CAR models for classifying wood boards," *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Aug. 1988.

[KOI89] A.J. Koivo and C.W. Kim, "Robust image modeling for classification of surface defects on wood boards," *IEEE Trans. Systems, Man, and Cybernetics*, vol.19, no.6, pp.1659-1666, 1989.

[KUA88] D. Kuan, H. Shariat, K. Dutta, and P. Ransil, "A constraint-based system for interpretation of aerial imagery," *Proc. IEEE CVPR 88*, pp.601-609, 1988.

[LEE87] N.S. Lee, Y. Grize, and K. Dehnad, "Quantitative models for reasoning under uncertainty in knowledge-based expert systems," *International Journal of Intelligent Systems*, vol.II, p.15-38, 1987.

[LEV81] M.D. Levine and S.I. Shaheen, "A Modular Computer Vision System for Picture Segmentation and Interpretation," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol. PAMI-3, no.5, pp.540-556, 1981.

[LI87] Z-N. Li and L. Uhr, "Pyramid vision using key features to integrate image-driven bottom-up and model-driven top-down processes," *IEEE Trans. Systems, Man, and Cyber.*, vol. SMC-17, no.2, pp.250-263, 1987.

[LIP87] R.P. Lippman, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol.4, no.2, pp.4-22, 1987.

[MAR80] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Society London B*, vol.207, pp.187-217, 1980.

[MCK85] D.M. Mckeown, JR., W.A. Harvey, JR., and J. Mcdermott, "Rule-Based Interpretation of Aerial Images," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol. PAMI-7, no.5, pp.570-585, 1985.

[NAG80] M. Nagao and T. Matsuyama, *A Structural Analysis of Complex Aerial Photographs*, Plenum, 1980.

[NAK60] G. Nakamura and H. Takachio, "Reflection of light and roughness on sanded surface of wood," *Journal of Japan Wood Research*, vol.6, pp.237-242, 1960.

[NAT86] *Rules for the Measurement & Inspection of Hardwood & Cypress*, National Hardwood Lumber Association, Jan. 1986.

[NEV86] R. Nevatia, "Image Segmentation", in *Handbook of Pattern Recognition*, T.Y. Young and K.S. Fu, Eds. Academic Press, 1986, pp.215-245.

[NIE85] H. Niemann, H. Bunke, I. Hofmann,G. Sagerer, F. Wolf, and H.Feistel, "A knowledge based system for analysis of gated blood pool studies," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.7, no.3, pp.246-258, 1985.

[PAU88] D. Paul, W. Hattich, W. Nill, S. Tatari, and G. Winkler, "VISTA : Visual Interpretation System for Technical Application - Architecture and Use," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.10, pp.399-407, 1988.

[PAV77] T. Pavlidis, *Structural Pattern Recognition*, New York : Springer-Verlag, 1977.

[PER83] W.A. Perkins, "INSPECTOR: a computer vision system that learns to inspect parts," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol. PAMI-5, no.6, pp.584-592, 1983.

[RIS84] E.M. Riseman and A.R. Hanson, "A Methodology for the development of general knowledge-based vision systems," *Proc. IEEE Workshop on Principles of Knowledge-Based Systems*, pp.159-170, 1984.

[ROB65] L. Roberts, " Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing*, J. Tippet, et. al. (Eds.), MIT Press, Cambridge, Massachusetts, 1965.

[RUC90] D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, and B.W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. on Neural Networks*, vol.1, no.4, pp.296-298, 1990.

[RUM86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing. Exploration of the Microstructure of Cognition, vol.1: Foundations*, D.E. Rumelhart and J.L. McClelland, Eds. Cambridge, MA: MIT Press, 1986.

[SAH88] P.K. Sahoo, S. Soltani, and A.K.C. Wong, "SURVEY : A survey of thresholding techniques," *CVGIP* 41, pp.233-260, 1988

[SAN62] W. Sanderman and S. Schulumbom, "On the effect of filtered ultraviolet light on wood. Part II: kind and magnitude of color differences on wood surfaces," *Holz als Roh-und Werkstoff*, vol.20, pp.285-291, 1962.

[SAN88] J.L.C. Sanz and D. Petkovic, "Machine vision algorithms for automated inspection of thin-film disk heads," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.10, no.6, pp.830-848, 1988.

[SAW77] A.A. Sawchuk, "Real-time correction of intensity nonlinearities in imaging system," *IEEE Trans. Comp.*, vol.26, no.1, pp.34-39, 1977.

[SHA76] G. Shafer, *A Mathematical Theory of Evidence*. Priceton, NJ: Princeton Univ. Press, 1976.

[SHI87] Y. Shirai, *Three-Dimensional Computer Vision*, Springer-Verlag, 1987.

[STA86] S.A. Stansfield, "ANGY: A rule-based expert system for automatic segmentation of coronary vessels from digital subtracted angiograms," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol. PAMI-8, no.2, pp.188-199, 1986.

[SUB86] V. Subramanian, G. Biswas, and J.C. Bezdek, "Document retrieval using a fuzzy knowledge-based system," *Optical Engineering*, vol.25, no.3, pp.445-455, 1986.

[SUL67] J. Sullivan, "Color characterization of wood: color parameters of individual species," *Forest Products Journal*, vol.17, no.8, pp.25-29, 1967

[TSI74] P.P Tsiang, "Computer analysis of chest radiographs using size and shape descriptors," Ph.D. Dissertation, Univ. of Missouri-Columbia, 1974.

[WAN84] S. Wang and R.M. Haralick, "Automatic multithresholding selection," *Comput. Vision, Graphics, Image Processing*, vol.25, pp.46-67, 1984.

[WAN90] E.W. Wan, "Neural network classification: A Bayesian interpretation," *IEEE Trans. on Neural Networks*, vol.1, no.4, pp.303-305, 1990.

[WIT83] A. Witkin, "Scale space filtering," *International Joint Conference on Artificial Intelligence*, pp.1019-1021, 1983.

[YAK76] Y. Yakimovsky, "Boundary and object detection in real world images," *J. Assoc. Comput. Mach.* 23, pp.599-618, 1976.

[YOD88] H. Yoda, Y. Ohuchi, Y. Taniguchi, and M. Ejiri, "An automatic wafer inspection system using pipelined image processing techniques," *IEEE Trans. Pattern Analy. and Mach. Intelli.*, vol.10, no.1, pp.4-16, 1988.

[ZAD65] L.A. Zadeh, "Fuzzy sets," *Information Control*, vol.8, pp.338-353, 1965.

Appendix A

BASIC DEMPSTER-SHAFER THEORY OF EVIDENCE

Def. **Frame of Discernment :** A *frame of discernment* Θ is defined as a set of all possibilities which are mutually exclusive and exhaustive.

Def. **Hypothesis :** Any subset of Θ is called a *hypothesis.*

Def. **Basic Probability Assignment :** Associated with each piece of evidence is a *basic probability assignment* (*bpa* or *m*-function) which represents the impact of the evidence on the subsets of Θ. A *bpa*, denoted by m, must satisfy the following conditions:

(1) $0 \leq m(A) \leq 1 \quad for \quad \forall A \subseteq \Theta$

(2) $m(\phi) = 0$

(3) $\sum_{A \subseteq \Theta} m(A) = 1$

Def. **Belief Function :** A *belief function* for A, any subset of Θ, denoted by $Bel(A)$, is the sum of the beliefs committed exactly to every subset of A by m, i.e.,

$$Bel(A) = \sum_{B \subseteq A} m(B)$$

Def. **Plausibility Function :** A *plausibility function* for A, denoted by $Pl(A)$, is defined as

$$Pl(A) = 1 - Bel(\overline{A}) = \sum_{B \cap A \neq \phi} m(B)$$

Def. **Belief Interval :** *Belief interval*, defined as [*Bel(A),Pl(A)*] represents the amount of uncetainty with respect to a hypothesis A given the evidence.

If there are multiple bodies of evidences supporting a hypothesis A, belief in A can be calculated by the *Dempster's rule of combination.*

Dempster's Rule of Combination:

Let m_1 and m_2 be bpa's corresponding to two independent pieces of evidence. Then new *bpa* m can be calculated by

$$m(A) = K \sum_{X \cap Y = A} m_1(X) m_2(Y) \quad if\ A \neq \phi$$

$$= 0 \quad if\ A = \phi$$

where

$$K^{-1} = 1 - \sum_{X \cap Y = \phi} m_1(X) m_2(Y)$$

Appendix B

THE BACKPROPAGATION LEARNING ALGORITHM

The learning of multilayer neural networks is mostly accomplished using the well-known *back-propagation (BP)* learning algorithm [RUM86]. To describe the BP algorithm the following notational conventions will be used.

- t_{pi} *Target value (desired output)* for ith component of the output pattern for pattern p.
- o_{pi} ith component of the actual output pattern produced by presenting input pattern p.
- i_{pj} jth element of the input pattern p.
- w_{ij} Weight from unit j to unit i.
- $\Delta_p w_{ij}$ The change to be made to w_{ij} following presentation of pattern p.
- a_{pi} Output value of unit i.
- net_{pi} Net input to unit i produced by the presentation of pattern p.
- E_p An error measure on input/output pattern p.
- E An overall error measure.

Let

$$E_p = \frac{1}{2}\sum_i (t_{pi} - o_{pi})^2$$

and let

$$E = \sum_p E_p.$$

Note that o_{pi} is a function of weights from output node i to all input nodes.

The objective is to find a set of weights that minimizes E. One way to do so is to use a *gradient-descent* method, which is characterized by the following weight update rule:

$$\Delta_p w_{ij} = -k \frac{\partial E_p}{\partial w_{ij}}.$$

First, assume that

$$net_{pi} = \sum_j w_{ij} a_{pj}$$

and

$$a_{pi} = f(net_{pi})$$

where f is a *semilinear activation function* which is differentiable and nondecreasing.

Using a chain rule in differential calculus,

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pi}} \frac{\partial net_{pi}}{\partial w_{ij}}.$$

Let

$$\delta_{pi} = -\frac{\partial E_p}{\partial net_{pi}}.$$

Then, since

$$\frac{\partial net_{pi}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_k w_{ik} a_{pk} \right) = a_{pj},$$

it follows that

$$\frac{\partial E_p}{\partial w_{ij}} = -\delta_{pi} a_{pj}.$$

Since

$$\frac{\partial a_{pi}}{\partial net_{pi}} = f'(net_{pi}).$$

the expression

$$\delta_{pi} = -\frac{\partial E_p}{\partial net_{pi}} = -\frac{\partial E_p}{\partial a_{pi}}\frac{\partial a_{pi}}{\partial net_{pi}} = -\frac{\partial E_p}{\partial a_{pi}} f'(net_{pi})$$

for δ_{pi} can be obtained.

If unit i is an output unit, so that $a_{pi} = o_{pi}$, then

$$\frac{\partial E_p}{\partial a_{pi}} = -(t_{pi} - o_{pi}).$$

If unit i is not an output unit but a hidden unit, then

$$\frac{\partial E_p}{\partial a_{pi}} = \sum_k \frac{\partial E_p}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial a_{pi}} = \sum_k \frac{\partial E_p}{\partial net_{pk}}\frac{\partial}{\partial a_{pi}}\left(\sum_j w_{kj} a_{pj}\right)$$

$$= \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{ki} = -\sum_k \delta_{pk} w_{ki}.$$

Therefore,

$$\Delta_p w_{ij} = \eta \delta_{pi} a_{pj}.$$

When unit i is an output unit, then δ_{pi} in the above equation is given by

$$\delta_{pi} = (t_{pi} - o_{pi}) f'(net_{pi})$$

and when unit i is a hidden unit, then δ_{pi} is given by

$$\delta_{pi} = f'(net_{pi}) \sum_k \delta_{pk} w_{ki}$$

The above is called the *generalized delta rule*.

Typically, semilinear function f is given by the *logistic function*, that is,

$$f(x) = \frac{1}{(1 + e^{-x})}.$$

In this case,

$$f'(net_{pi}) = a_{pi}(1 - a_{pi}).$$

Hence for an output unit i,

$$\delta_{pi} = (t_{pi} - o_{pi})o_{pi}(1 - o_{pi}),$$

and for a hidden unit i,

$$\delta_{pi} = a_{pi}(1 - a_{pi}) \sum_k \delta_{pk} w_{ki}.$$

For faster convergence, it is recommended [RUM86] to use

$$\Delta w_{ij}(n + 1) = \eta(\delta_{pi} a_{pj}) + \alpha \Delta w_{ij}(n)$$

where n represents nth presentation, η is called the *learning rate*, and α is called the *momentum term* that determines the effect of past weight changes on the current weight changes.

Appendix C

GLOSSARY

Check : A lengthwise separation of the wood that usually extends across the rings of annual growth and commonly results from stresses set up in wood during seasoning.

Decay : The decomposition of wood substance by fungi.

Seasoning (Air-dried) : Dried by exposure to air, usually in a yard, without artificial heat.

Split : A lengthwise separation of the wood, due to the tearing apart of wood cells.

Stain : In hardwood lumber grading, the word "stain" is used to describe the initial evidences of decay.

Wane : Bark or lack of wood.

VITA

Tai-Hoon Cho was born on August 4, 1958 in Chonju, Korea. He received a B.S. degree in Electronics Engineering from Seoul National University, Seoul, Korea, in 1981, and M.S. degree in Electrical Engineering from Korea Advanced Institute of Science and Technology, Seoul, Korea, in 1983. He worked as a research engineer for the Ministry of National Defense, Korea, from 1983-1986. After studying five years at the Viginia Polytechnic Institute and State University, Blacksburg, Virginia, he received a Ph.D. degree in Electrical Engineering in 1991. During the period of Ph.D. study, he was a Teaching Assistant and a Graduate Project Assistant.