A GENERAL PURPOSE MACHINE VISION PROTOTYPER FOR INVESTIGATING THE INSPECTION OF PLANAR WEBS

by

Chong Teck Ng

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

April 1993

Blacksburg, Virginia

A GENERAL PURPOSE MACHINE VISION PROTOTYPER FOR INVESTIGATING THE INSPECTION OF PLANAR WEBS

by

Chong Teck Ng

Committee Chairman: Richard W. Conners

Electrical Engineering

(ABSTRACT)

In order for an industrial inspection system to be of utility in manufacturing it must be fast, accurate, and flexible [Chin 1986]. Current machine vision systems are very specialized and inflexible in nature. A reason for the inflexibility of current machine vision systems is the need for real-time processing of image data. Such a need has forced both the use of very specialized image processing hardware as well as the use of rather simple, very specialized computer vision algorithms to do the analysis. On the other hand, most, if not all, of today's computer vision methods are not general purpose in nature. In the absence of truly robust general purpose methods, developing satisfactory machine vision solutions will continue to involve experimenting with machine vision hardware and software components.

Given the current state of machine vision technology, it would seem that the best method for creating flexible machine vision systems is, perhaps, to define a subclass of inspection problems where all the problems within the subclass have a number of common features about them. Such a subclass must be of interest to a number of manufacturers. It must also be "reasonable" to solve, given the current state of the art. Once the subclass has been selected, the next logical step would seem to be to create a device that makes performing all the needed experiments on the various problems within the class easy to perform.

Based on the above line of reasoning, this work has four major objectives. The first objective is to define a meaningful subclass of inspection problems that are a) of interest to a number of manufacturers, and b) represent inspection tasks that seem "reasonable" within the current state-of-the-art of computer vision. The subclass of inspection problems selected for this work is the longitudinal planar web inspection problem under the two-dimensional imaging restriction.

The second objective of this work is to create a vehicle that will allow the types of experimentation usually associated with the development of machine vision systems to be facilitated. This vehicle created is called a "machine vision prototyper."

The third objective of this work is to use the machine vision prototyper system to attack a particular planar web applications problem. The application considered is the problem of locating and identifying surface defects in surfaced hardwood lumber in a species independent manner.

The fourth objective of this research is to indicate how the prototyper system can be used to attack a second planar web application problem. This application problem is the inspection of hardwood parts coming out of a molder.

The utility of the machine vision prototyper system as an experimental tool is demonstrated on two of the three possible types of longitudinal planar web inspection problems. The results include the development of a machine vision system for a hardwood surfaced lumber surface feature detection problem, and a discussion of how the prototyper can be used to attack the problem of inspecting hardwood parts coming out of a molder.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Richard W. Conners, for his guidance, patience, and support throughout the course of my Ph.D. studies. I have learnt much during the many years I have worked with him. This dissertation would not have been finished without his help. I am also deeply grateful to Dr. Conners for the continuous financial support I received as a graduate student.

I would also like to thank Dr. A. Lynn Abbott, Dr. Dong S. Ha, Dr. Joseph G. Tront, and Dr. Fred M. Lamb for their help in the preparation of this dissertation, and for the time they have spent as members of my committee. I would also like to further express my appreciation to Dr. Lamb for the many hours he spent introducing me to the world of defects in hardwood lumber.

Financial support during my studies was mainly provided by the U.S. Forest Service and is greatly appreciated. I am very grateful to Philip A. Araman for making this support possible.

Special thanks must be given to my parents for their love and understanding. They were the source of my motivation and endurance. Throughout the many years I have spent as a student, they never lost faith in me. I am also deeply grateful to my wife, Bee Leng, for her love and patience in going the most difficult "last mile" with me. This dissertation is dedicated to her.

Ultimately, this dissertation would not have been possible without the Lord's help. He understood my dependence on miracles. To Him be the glory.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Motivation

Machine vision systems for automated industrial inspection have been developed for a number of manufacturing applications (Chin (1982), Kent (1986), Chin (1988)). In order for an industrial inspection system to be of utility in manufacturing it must satisfy the following requirements (Chin (1986)):

1. Its speed of operation must match production line speeds.

2. Its accuracy in defect detection must be acceptable.

3. It must be flexible in order to accommodate changes in products.

Current machine vision systems are very specialized in nature. Each represents a very specialized set of algorithms implemented on very specialized image processing hardware. Each system's purpose is to solve, in real-time, a very specific industrial inspection problem. As such current systems are very inflexible in that they cannot be easily adapted to meet the changing requirements associated with the inspection of a particular product, nor can they be easily modified to run on inspection problems other than the one each was explicitly defined to solve. As Chin (1986) has stated:

> "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."

The inflexibility of current machine vision systems has markedly limited the utility of these systems in manufacturing and has kept these systems from being very widely used. A number of manufacturers who attempted to employ machine vision technology in the early to mid 1980s have removed these systems from their productions lines and have not replaced them with any newer technology. If machine vision is to be extensively used in manufacturing it must be "flexible."

Part of the reason for the inflexibility of current machine vision systems comes from the need for real-time processing of image data. As Chin pointed out, for a machine vision system to be of utility it must be able to match industrial production speeds. This means that the system must be able to process images in real-time. The need for real-time processing forces both the use of very specialized image processing hardware as well as the use of rather simple, very specialized computer vision algorithms to do the analysis.

The Intel Corporation predicts that by the year 2000 a benchmark microprocessor system will be able to execute 2 billion instructions per second. The availability of microcomputer systems capable of executing this number of instructions per second should alleviate the problem of creating machine vision systems that can match production line speeds. The availability of these systems should mean that the speed issue is no longer of concern in the majority of inspection problems encountered in manufacturing. General purpose microcomputers with this level of computational capability should allow even sophisticated computer vision algorithms to be executed in real-time without the need for specialized image processing hardware. The availability of such microcomputers should also mean that future machine vision systems should be more flexible. This follows from the fact that more computationally complex methods are typically more robust, i.e., more applicable to a wider range of problems and from the fact that the general purpose computer, i.e., the high speed microcomputer, is typically much easier to reprogram to address a new or modified problem than the specialized image processing hardware it will replace. Further, the availability of such systems should mean that future machine vision systems should be more accurate. Logic suggests that computationally complex algorithms should be more accurate than less sophisticated simple methods. Finally, the availability of such systems should also markedly reduce the complexity of developing a machine vision system since developers are no longer confined to using computationally simple computer vision algorithms and are freed from the task of having to design or select special purpose image processing hardware. As such, the availability of computationally very capable microcomputer systems would seem to offer the possibility of markedly expanding the role machine vision technology can play in automating manufacturing processes.

Unfortunately, the existence of such microcomputer systems does not, within itself, guarantee the expanded use of machine vision technology in manufacturing. Perhaps the most troubling problem that limits the expanded role of machine vision technology in manufacturing is the current state-of-the-art in computer vision algorithms. After nearly thirty years of research the development of truly robust, general purpose image analysis and scene analysis methods has proven to be a very, very difficult task. Irrespective of computational complexity, most if not all of today's computer vision methods are not general purpose in nature. Each seems to work well on only a fairly limited class of problems. In fact, the lack of progress in creating general purpose computer vision algorithms capable of at least matching some level of human visual performance has disheartened some investigators. What has resulted after these thirty years of research is a huge collection of computer vision algorithms without any common unifying thread. This frustration is well expressed by Zavidovique et al. (1991):

> "Nowhere are existing theories and algorithms characterized in a way that permits us to thoroughly understand image-processing semantics, regardless of implementation or approach. Instead, this vast know-how is scattered in conference proceedings, books, and lecture notes, which makes it harder for users to learn the fundamentals. Because there is no unifying formalism, the study of algorithm design has suffered. Instead, there is a neverending production of algorithms that resembles a creative art more than a well-established science."

Given the record over the last thirty years, it seems prudent to assume this inability to create truly general purpose methods will persist well into the future. This observation has a number of significant ramifications. First, it implies that there is going to be some definite limit as to how flexible machine vision systems can be made to be. It means that a "flexible" system might only be able to handle a limited subclass of problems. Obviously for a system to be of utility the subclass of problems it can handle must be ones that are of interest to a number of manufacturers. And this "flexible" system must be flexible enough to address the differing needs of the various manufacturers interested in this subclass of inspection problems.

The above observation about the current and foreseeable state of computer vision technology has one other important ramification. This is that creating automatic methods for solving a particular industrial visual inspection problem is, and will remain for some time, an art and not a science. Since developing machine vision systems is an art, one cannot be assured that given a particular applications problem a solution yielding the required accuracies can be found. Since developing machine vision systems is an art, one is even more hard pressed to arrive at a "flexible" solution to a subclass of inspection problems.

In the absence of truly robust general purpose methods, developing satisfactory machine vision solutions will continue to involve fixing the viewing geometry, using special lighting, and/or employing special sensors, so that the computer vision algorithm development/selection problem is markedly simplified. Therefore, developing machine vision systems requires and will continue to require a good deal of experimentation, experimentation with a variety of different computer vision algorithms, experimentation with various viewing geometries, experimentation with various types of lighting, experimentation with various types of sensors, and experimentation with various filters, reflectors, lenses, etc. A good deal of experimentation is typically needed to arrive at an acceptable solution and even more experimentation needed to arrive at a "flexible" solution for a subclass of problems. And, of course, it is the machine vision research community's ability to create flexible solutions that is going to determine the extent to which machine vision systems are going to be used in manufacturing.

1.2 Underlying Assumptions

It would seem that few things in science and engineering are revolutionary.

Rather most are evolutionary. Hence, the best method for creating flexible machine vision systems is, perhaps, to define a subclass of inspection problems where all the problems within the subclass have a number of common features about them. The subclass chosen must satisfy the following criteria: 1. The subclass of problems selected must be such that a number of manufacturers are interested in inspection problems within this subclass. Just as important is the fact that: 2. The inspection problems within the subclass seem "reasonable" to solve given the current state-of-the-art of computer vision.

Once the subclass has been selected, the next logical step would seem to be to create a device that makes performing all the needed experiments on the various problems within the class easy to perform. Reducing the difficulty in performing the needed experiments should reduce the time needed to find a satisfactory solution for each of the problems in the subclass. Hence, it should also reduce the time needed to find a satisfactory "flexible" solution to all the problems within the subclass.

Once the experimental tool has been created, the next logical step is to select a problem from within the subclass, and, using the experimental tool, attempt to find a satisfactory solution for this problem. Then, select another problem from within the subclass and using the experimental tool, attempt to find a solution to this problem. Obviously, to generate a solution for the second problem, some of the methods used in solving the first method should be tried. If these methods do not work on the second problem the reasons for the failures should be documented. If the methods used on the first problem can be modified so that they will now work on both problems one and two, these modifications should be made. If not, new methods must be created. The desire in creating the new methods is to develop algorithms that will work on both problems one and two. New methods must also be developed should the second problem require a different solution. This evolutionary process continues with the selection of further problems to be solved. The methods to be explored involve not only algorithm development but also hardware considerations such a lighting conditions, viewing angles, sensors, etc.

It is assumed that this particular line of attack represents a very good method for evolving a machine vision technology that will eventually be flexible enough to address all the problems in the subclass. Hence this will be the basic line of approach used in this dissertation.

1.3 Objectives

Based on the above underlying assumptions, this work has four major objectives. The first objective is to define a meaningful subclass of inspection problems that are a) of interest to a number of manufacturers, and b) represent inspection tasks that seem "reasonable" within the current state-of-the-art of computer vision. The subclass of problems that was selected is those involving planar webs. The reasons for selecting this particular subclass are given in the next subsection.

The second objective of this work was to create a vehicle that will allow the types of experimentation usually associated with the development of machine vision systems to be facilitated. This vehicle created is called a "prototyper" system because it allows an investigator to easily and rapidly prototype both the hardware and software components of a machine vision system so that ability of various combinations of components to solve a particular applications problem can be evaluated. This machine vision prototyper system was designed, in particular, to address the planar web inspection problem.

The third objective of this work was to use the machine vision prototyper system to attack a particular planar web applications problem. The application considered is a relatively complex industrial inspection problem, the problem of locating and identifying surface defects in surfaced hardwood lumber in a species independent manner. The machine vision system being developed using the prototyper is for use in the hardwood furniture and fixture industry. The goal of this research was two fold. The first was to demonstrate the utility of the prototyper in facilitating the experimentation needed to develop machine vision systems aimed at the planar web inspection problem. The second goal was to create at least a partial solution to the surfaced lumber inspection problem. The goal was to try to create a machine vision system that is as complete as possible, given the limitations of time and funding.

The fourth objective of this research is to indicate how the prototyper system can be used to attack a second planar web application problem. This application problem is the inspection of hardwood parts coming out of a molder. The molding operation introduces machine defects in otherwise acceptable parts. The defects caused by the molder typically involve the lack of the desired smoothness on the surface of the machined part or in a deviation from a desired three-dimensional shape.

1.4 Scope and Limitations

To fulfill the first objective, the subclass of industrial inspection problems selected for study in this research is a subclass of a set of problems commonly called web inspection problems. Classically, a web is defined to be any material produced in the form of strips (Purll (1985)). The general web inspection problem consists of the inspection of continuous strips of material for the detection of undesirable defects or blemishes. Under this definition of webs, however, the inspection of any material can be categorized as a web inspection problem. To restrict the extent of the problems to be addressed to a manageable size , i.e., to make it "reasonable", a subclass of webs will be considered. The restrictions considered involve both the permissible characteristics of the web and the types of image data collected from the web. Let $\overline{v}$ be a unit vector denoting the direction of travel. Let P be a plane perpendicular to $\overline{v}$. If there exists an orientation of an object such that when the object is in this orientation and travels along $\overline{v}$, it presents a uniform cross section as it passes through P, the object will be called a web. This is further illustrated in Figure 1.1. In the figure, P denotes a plane perpendicular to $\overline{v}$ and U is the uniform cross section on plane P. Referring to Figure 1.1, U is further defined as a four-sided polygon with the sides S1 and S3 being of lengths that are appreciably smaller than the lengths of sides S2 and S4. The web will be called a **planar web.** Under this definition planar webs have constant shape characteristics with respect to the plane P.

However, even the planar web problem is still too broad to reasonably believe that a uniform set of methods can be found for attacking all such problems. Therefore further restrictions are required. Suppose the dimensions of a planar web in the plane P are small in relation to the dimension of the web along $\overline{v}$. Such planar webs will be called **longitudinal planar** webs. There would seem to be only three types of inspection tasks that can be performed on longitudinal planar webs. One type of inspection involves the analysis of the surfaces of the longitudinal web to look for surface defects that manifest themselves as variations in color,

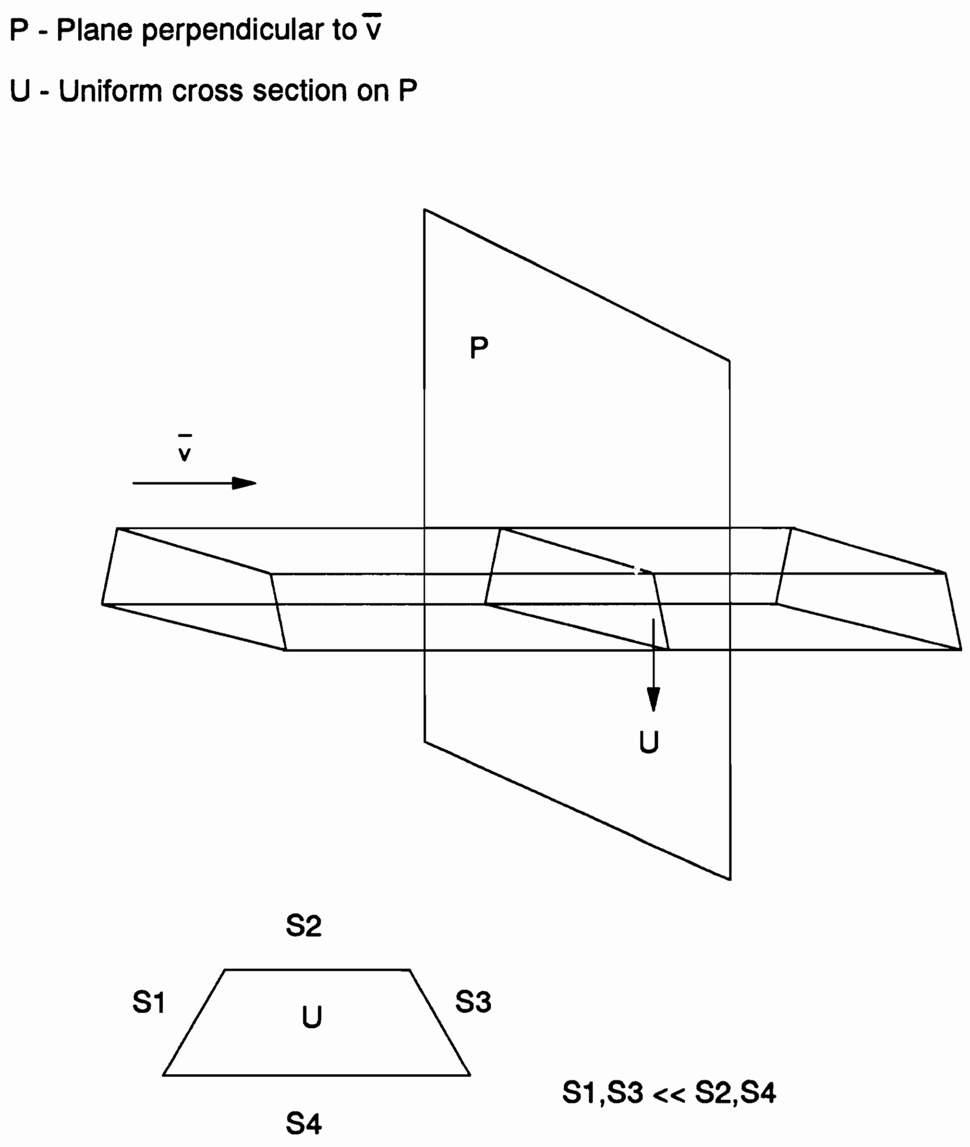
P - Plane perpendicular to $\bar{v}$
U - Uniform cross section on P
P
$\bar{v}$
U
S2
S1
U
S3
S4
S1,S3 << S2,S4

Figure 1.1. Diagram depicting the definition of a planar web.

tone, and/or texture from that of the normal web surface. A second type involves examining the surface of longitudinal planar webs to look for defects that manifest themselves by causing a variation in the cross section the web presents in the plane P as the web travels through P along $\overline{v}$. Such defects alter the three-dimensional shape characteristics of the longitudinal planar web. A third type of inspection involves examining the longitudinal web to look for internal defects in the web.

A careful examination of this last inspection type suggests that there can be a wide variation in the type of sensors that can be employed on this inspection problem. Techniques for performing this inspection operation range from simple backlighting methods in those instances when the longitudinal webs are transparent or translucent to x-ray scanning to even computer tomography, magnetic resonance imaging or ultrasonic imaging. These last three imaging modalities create three-dimensional images of the web being inspected. All the other sensors only create two-dimensional images of longitudinal planar webs. The analysis of three-dimensional images can be very different from the analysis of two-dimensional images. Hence for purposes of this research, the subclass of webs to be considered is longitudinal planar webs with the additional restriction that these webs will be imaged with sensors that create only two-dimensional images. The problem to be considered will be called the longitudinal web inspection problems under the two-dimensional imaging restriction.

This particular subproblem was chosen for a variety of different reasons. First, the longitudinal web inspection problem under the two-dimensional imaging restriction is an important subclass of industrial inspection problems in that a number of manufacturers have a potential interests in its solution (Chin (1986)). The manufacturing industries that could use machine vision systems that address this class of problems include the plastics industry, the metals industry, the textile industry, and the forest products industry. Therefore this subclass satisfies the above stated first requirement for selection.

Next, the longitudinal planar web inspection problem under the two dimensional restriction seemingly lends itself to only two basic types of imaging technology, one of which is line scan imaging technology for surface defect inspection. Also, to successfully attack the totality of longitudinal planar web inspection problems under the two dimensional imaging restriction seemingly requires the use of multiple sensors, all imaging the web simultaneously. In particular it is entirely possible that the longitudinal planar web problem will require at least one sensor to detect surface defects, at least one sensor to detect interior defects, and at least one sensor for gauging shape related defects.

Most general purpose image processing hardware is designed to accept data from only one imaging modality, typically an area array cameras. Further, general purpose systems do not lend themselves to the easy collection of data from a line scan imaging device let alone a number of these devices. Nor does most general purpose image processing hardware provide a mechanism for the control of the materials handling components needed to move objects through the field of view of the various line scan cameras used in a data collection configuration. Hence, the hardware needed to attack this subclass of problems is not generally available. It is perhaps for this reason that this subclass of problems has not been significantly investigated.

Given that the hardware needed to do the required experiments on the longitudinal planar web inspection problem is not generally available, the first task was to create a hardware system that could be used to address a spectrum of problems in this subclass. The resulting experimental testbed is called a **machine vision prototyper**. The "prototyper" is a collection of reusable hardware and software components that can be combined in a sufficient number of ways to allow one to consider different solutions to an inspection problem. The software components include computer vision algorithms, display and plotting functions, device drivers for various hardware components, and control algorithms for controlling various mechanical and electronic devices. As such, the software components of this system represent a superset of the operations normally found in a typical image processing system. The hardware components include various imaging and non imaging sensors; sensor interfaces to a computer; various types of lighting; mounting components for both the sensors and lighting; filters for selecting and/or rejecting various wavebands of electromagnetic energy; various lenses; a mounting platform that facilitates the easy relocation of components; a free standing enclosure for preventing ambient electromagnetic energy from entering the imaging areas; a materials handling system for moving objects through the imaging areas; a cooling system for removing heat generated by the illumination systems from the free standing enclosure; and a computer for collecting image data, controlling mechanical and electronic components, and for performing the image analysis and scene analysis functions. This system was created so that all the components are reusable and are easily reconfigurable, i.e., are extensible.

To obtain the significant degree of reusability and extensibility needed for the rapid prototyping of machine vision systems, the design of the prototyper is based on the concepts of object-oriented system design. Using an object oriented scheme allows the reusable hardware components that interact with the computer and the software components to be treated in the same conceptual manner. Each reusable hardware component is an object defined by the construction of the component, and the functions it performs through commands sent by the computer. Each reusable software component is defined by a set of data structures and a set of software operations it performs, or that can be performed on it. The software for the prototyper is not written in an object oriented-programming language such as C++, but in the C language. The reason that an object-oriented programming language is not used is because of the amount of executable overhead involved in supporting object-oriented mechanisms. Since the prototyper is designed to handle large high resolution images collected from multiple sensors, software speed becomes a very real concern. This is the reason C was selected. The use of C allows the programmer to custom design the object-oriented mechanisms to suit the application. For the prototyper, the object-oriented mechanisms have been designed to support a software environment that is meant to operate on image data only.

It has been argued that currently, the most robust computer visions systems are all knowledged based (Cho (1991)). Obviously, in designing the prototyper it was important that the software methodologies used support the development of knowledged based computer vision methods. The object oriented paradigm supports such development by providing a natural way of implementing a blackboard data structure (Cappellini (1990)), a data structure that is arguably the one to use in knowledge based vision systems (Nagao (1980)).

The particular longitudinal web inspection problem selected for consideration in this research was the hardwood surfaced lumber, surface defect inspection problem. The goal is to create methods that can locate, identify and measure the “important characteristics” of surface defects such as knots, split/checks, and the like in a species independent manner. This problem will be described in detail later. This particular problem was selected because it represents what is seemingly one of the more difficult longitudinal web inspection problems. The difficulty of the problem stems from the fact that hardwood species vary significantly in their appearance, and in the way surface defects manifest themselves in their lumber. Even within a species, there is significant variation in the appearance of boards and in the way surface defects manifest themselves from one tree to another.

An objective of this research is to demonstrate the feasibility of addressing this problem using machine vision methods. If one can establish the feasibility of addressing this problem, then it can be argued that the subclass of longitudinal web inspection problems may be “reasonably” solved using today’s technology or variations on this basic technology, i.e., this subclass of problems can be addressed by evolutionary methods and not revolutionary methods.

A second longitudinal web inspection problem will also be described. This second problem addresses the inspection of the three-dimensional shape characteristics of longitudinal planar webs. The sensor technology used is a black and white array camera, laser based structured lighting, and algorithms for transforming the two-dimensional images obtained from the array camera into a line of ”range” data. There are two goals in describing this inspection problem. The first is to demonstrate the utility of the prototyper system in addressing a ”variety” of longitudinal planar web inspection problems. The second goal is to further demonstrate that a spectrum of longitudinal planar web inspection problems would seem ”reasonable” given today’s computer vision technology and/or an evolutionary variation of this technology.

There are a number of limitations to the reported research. First the prototyper system was not developed to its most general form. The basic concept of the prototyper system is to have a variety of both hardware and software components readily available for use to perform any needed experiments. Limitations in both time and money has limited the number of reusable components that could be developed for future work. However, the demonstration of the usefulness of the system across two of the three general categories of inspection problems that can be performed on longitudinal planar webs hopefully demonstrates this system's capability in absence of it being in its fully developed form.

Another limitation of the work concerns the extent of the research on the surfaced lumber inspection problem. Once again limits on time and funding prevented a more complete vision technology to address this problem from being developed. Much of the work was concentrated on the low-level segmentation component of the vision system. The high-level recognition component was not developed to the extent desired. However, even the rather limited scope of the work presented does suggest that, with more research, an adequate solution to this automated industrial inspection problem can be found. In fact, the reported research provided the basis for a more complete work by Cho (1991).

Perhaps because of the above, one can argue that a final limitation of the study comes from the fact that the "reasonableness" of the longitudinal planar web inspection subclass of problems has not been completely demonstrated. And this is true. However, the goal of university research is to consider problems whose feasibility has not yet been completely demonstrated.

1.5 Hypotheses

The basic hypotheses that have been tested during the course of this research is that

1. The machine vision prototyper system can be used as an experimental tool for investigating the subclass of longitudinal planar web inspection problems under the two-dimensional imaging restriction. The hypothesis is tested on two of the three possible types of longitudinal planar web inspection problems.

2. The development of a machine vision system for solving a hardwood surfaced lumber surface defect detection problem is feasible. Such a machine vision system has been developed and tested.

1.6 Organization of the Dissertation

The next chapter introduces rapid-prototyping and machine vision system design. In the chapter, a background into the wide variety of technology necessary for designing the prototyper is provided. Chapter 3 describes machine vision prototyping and overall design of the machine vision prototyper. Chapter 4 presents a detailed description of the hardware and software development of the prototyper. Chapter 5 defines the hardwood lumber inspection problem and Chapter 6 demonstrates the utility of the prototyper in designing a machine vision system for the hardwood lumber inspection problem. Chapter 7 will demonstrate the utility of the prototyper on a second planar web inspection problem. In particular, it will describe how a structured light-based vision system can be designed using the prototyper. A summary and conclusion is provided in Chapter 8. Recommendations for future research are also given in Chapter 8.

Chapter 2

RAPID-PROTOTYPING AND MACHINE VISION SYSTEM DESIGN

The meaning of the terms machine vision system, computer vision system, and industrial inspection system has varied in the literature so much so that sometimes these expressions are used interchangeably. For purposes of this dissertation, the term **machine vision system** is taken to mean a system consisting of both hardware and software components. The hardware components of a machine vision system include the sensors, the illumination sources, the materials handling system, and a computing system. The software components of a machine vision system include a computer vision system and control software for controlling the imaging hardware and materials handling system. The term **computer vision system** refers to the software system involved in the analysis of images. Important issues in a computer vision system are the algorithms and mathematics used in the analysis process. An **industrial inspection system** is a machine vision system applied to a particular industrial inspection task.

As described in Chapter 1, the second objective of this work is to create an experimental tool, i.e., a machine vision prototyper, that will facilitate the experiments needed to create machine vision systems for various planar web inspection problems. The hope is that in considering the various planar web inspection problems, a single, flexible, machine vision technology will evolve, a technology that will be able to handle a broad cross section, if not the entire class, of planar web inspection problems.

Successful achievement of this objective requires an understanding of, not only the general planar web inspection problem, but also an understanding of computer vision technology, machine vision technology, and rapid-prototyping system design methods. This chapter provides the necessary background in these subject areas.

The objectives of this chapter are four fold. The first objective is to restate in more detail the planar web inspection problem and to discuss the basic characteristics of the problem. Designing a machine vision prototyping system for planar web inspection applications implies the need for an appropriate method of system design. The second objective is thus to introduce a suitable method of system design. As the "prototyper" is a system for designing machine vision systems, a mechanism must be provided for designing such systems. The mechanism chosen is the one associated with rapid-prototyping. Associated with the concept of rapid-prototyping is the concept of reusability. For components to be reusable, these components must correspond to functions that are common tasks performed by machine vision systems on a wide variety of inspection tasks. The third objective, then, is to study the typical components of computer and machine vision systems. The fourth and final objective is to study existing computer and machine vision systems to identify the characteristics that they have in common. In studying these systems, an objective is to show that the incorporation of knowledge based methods are a common trend and therefore, the machine vision prototyper must be able to support the use of knowledge based methods.

2.1 Planar Web Inspection

In Chapter 1, a **planar web** was defined as having a uniform cross section on a plane as it travels along an axis perpendicular to the plane. It is basically a continuous strip of material of finite length, measured along this axis. A restriction was further imposed on the dimensions of the planar web in order to make solving the problem more tractable, i.e., more "reasonable" to solve. A subclass of planar webs, **longitudinal planar webs**, was defined as planar webs with cross sectional dimensions in the plane being small in relation to the dimension of the web along the direction of travel. A final restriction is that these longitudinal planar webs will be imaged with sensors that create only two-dimensional images. Henceforth, the term **planar web** will be taken to mean **longitudinal planar web** unless it is otherwise stated.

There are basically three types of planar web inspection problems. The first is to detect undesirable defects or blemishes on the surfaces of a planar web. A **defect** or **blemish** on a surface of a planar web can be defined to be a region on the surface of the planar web that is a connected subset of the surface with properties that make it undesirable. A connected region is one in which a curve can be drawn to join any two points within the region such that the curve lies completely within the region. These undesirable properties manifest themselves as variations in color, tone, and/or texture from that of the normal web surface. The second type of planar web inspection problem involves examining the shape of the surface of the web to look for defects that manifest themselves by causing a variation in the shape of the uniform cross section on the plane perpendicular to its direction of travel. Finally, the third type of planar web inspection problem involves the location of internal defects. It should be noted that, for planar webs in general, defects should have different characteristics from defect free regions. It is also assumed that defect free regions are characterized by some locally uniform property.

Examples of planar web inspection systems in industry include those that inspect copper strips (Purll (1978)), textiles (Hashim (1984)), printed circuit boards (Benhabib (1990)), thin-film disk heads (Sanz (1988)), and hardwood lumber (Conners (1989a)).

There are also many examples of other types of webs that are not planar webs but which may be considered within the framework of planar web inspection. An example is the inspection of glass bottles (Giebel (1982)). In this application, the transparency of the bottles are an important factor. These bottles are imaged by a linescan camera and defects show up as opaque regions. Only a two-dimensional analysis of the images using texture analysis methods, is required. Another example is the inspection of apples for surface bruises (Rehkugler (1986)). In this application, the surfaces of apples are imaged by rotating these apples for imaging by a linescan camera. A simple thresholding method is then used to detect any bruises.

We have seen in Chapter 1 that there are a large number of problems that must be addressed in the automated inspection of planar webs. In order to consider a sufficiently broad range of planar webs, a great deal of flexibility is needed. Considering the breadth of the planar web inspection problem, a machine vision prototyper is proposed as an experimental tool that can be used in attacking the planar web inspection problem. The machine vision prototyper is based on a method of system design known as rapid-prototyping.

2.2 Rapid-Prototyping and Object-Oriented System Development

Prototyping is a common mechanism by which products in industry are brought into existence and tested. The work described in this dissertation takes this a step further by proposing a prototyper for designing prototypes. The design of the prototyper is based on the concepts of rapid-prototyping. A powerful means by which rapid-prototyping can be achieved is through the use of object-oriented methods (Mullin (1990)). In this section, the concepts of prototyping, rapid-prototyping, and object-oriented system design is introduced.

The term **prototyping** refers to the process of developing a scaled-down version of a system to use in building a full-scale system. The term **rapid-prototyping** refers to the process of quickly building and evaluating a series of prototypes (Tanik (1989), Luqi (1989)). Rapid-prototyping has been applied to a diversity of applications such as the design of user interfaces (Gray (1988)), VLSI computer-aided design systems (Gupta (1989)), testing vehicle control algorithms in aircraft (Duke (1989)), and the study of communication network design and behavior (Cieslak (1989)). The last two references describe systems that involve both hardware and software aspects. Cieslak, et al., in particular, describe the use of reusable hardware modules within the prototyping environment that provides the software tools to help design, test, and analyze a network (Cieslak (1989)).

The concepts of rapid-prototyping have mostly been associated with software system development. These concepts revolve around the rapid-prototyping cycle that begins with the specification of requirements followed by construction of a prototype. This prototype is then tested and a refinement of the requirements made, should it not be acceptable. This is followed by another cycle of prototyping and testing. This procedure is exemplified in great detail by Mullin (1990). The reason that such a procedure is needed is due to the fact that exact requirements for system development in industry is typically not completely known (Jacobson (1991)). As noted by Jacobson, incremental development of a system may thus be a preferred strategy that can be facilitated by rapid-prototyping.

In introducing the utility of prototyping to the analysis of user requirements, Andriole (1989) makes the following statement:

> "Where did we go wrong? Why do so many of our interactive computer-based systems fail to satisfy user requirements? The short answer is that things have changed."

Andriole later remarks that:

> "Regardless of arguments against prototyping, it has become clear over the past few years that analytical requirements simply cannot be captured via conventional 'specifications,' and that new tools, techniques, and attitudes are necessary."

There are two opposing views in design philosophy that are sometimes followed in the design of complex systems. The first consists of quickly designing a series of prototypes with each preceding prototype being discarded as a new improved version comes into existence. This method is wasteful in terms of time spent and cost of equipment lost. This method can be regarded as a revolutionary method of of system design. The second view requires a complete specification of system requirements and design parameters leading to a final prototype in a single step. Such a methodology would be based on formal design methods such as structured design. This method is usually applied only to simpler systems whose requirements can be fully specified. Such systems tend to be specialized designs and are therefore inflexible. The complete requirements of complex systems, however, may not always be known in the initial design phase. It is often the case that new requirements come into existence during the testing of a prototype perhaps due to some fact or consideration that had been overlooked. For these reasons, the middle ground based on a combination of these two opposing philosophies is often desirable. This middle ground consists of the ability to quickly design prototypes based on reusable components that can be designed using formal methods. This is the rapid-proptotyping approach to system design and is basically evolutionary in nature.

The approach to system design that requires the complete specification of user requirements before implementation has been compared to rapid-prototyping by Boehm et al. (1984) in an experiment in which four groups of software teams used the "specifying" approach and three groups used the "prototyping" approach. One of the results of the experiment was that prototyping yielded products with roughly equivalent performance, but with 40 percent less code and requiring 45 percent less effort to develop.

Among the most extensive work in the development of systems for the rapid-prototyping of software is that which has been done by Luqi. In a series of papers, the combination of a high level language known as the Prototype System Description Language (PSDL) (Luqi (1988b)), with an automated prototyping environment (Luqi (1988c)) for rapidly prototyping real-time systems (Luqi (1988a)) is described. The methodology was further extended and formalized for software systems, in general, in another study by Luqi (1989)). This study describes an integrated set of computer-aided software tools known as the Computer-Aided Prototyping System (CAPS) that supports software evolution through rapid-prototyping. This set of tools provides facilities for quickly configuring prototype designs, facilities for easily making design changes, and includes a database of reusable software components. These are all essential aspects of rapid-prototyping.

An important aspect of the work described by Luqi (1989) is the concept of object-oriented prototyping. Object-oriented prototyping is described by Luqi to be the "rapid construction of software systems using objects that encapsulate data as building blocks." The use of objects as building blocks of a system serves to localize information for reducing the complexity of a system. This aids rapid-prototyping by allowing one to make changes in a prototype through modifications on an object. Object-oriented programming has become an important aspect of software design as it gives a programmer a better understanding on how to view a programming problem, promotes reusability, and makes software modifications simpler. It has, in fact, gained considerable attention in the computer industry (Verity (1991)). Object-oriented software development has seen a wide variety of applications such as in rapid-prototyping systems (Luqi (1989)), simulation of manufacturing systems (Stein (1989)), telecommunication switching systems (Destor (1990)), database management systems (Atwood (1991)), image understanding (Lawton (1990)), and a variety of artificial intelligence applications (Alpert (1990)) such as knowledge representation (Newton (1988)), expert system shell design (Leung (1990)), and expert systems (Duff (1991)). It is also recommended as a means of rapid-prototyping machine vision software (Walsh (1990)).

The object-oriented approach to software development as defined by Booch (1986) is:

> "... an approach to software design in which the decomposition of a system is based upon the concept of an object. An object is an entity

whose behavior is characterized by the actions that it suffers and that it requires of other objects."

This approach entails the modeling of a system by a set of objects that simulates the behavior of various parts that make up the system. The advantage to this is the ability to decompose a complex problem into smaller sub-problems in a manner that seems natural and simpler to do. Note that this approach need not be limited to software design. Hardware design can also be decomposed in this manner. Object-oriented design thus provides a conceptual framework under which both hardware and software components can be treated in the same conceptual manner.

The concepts of prototyping, rapid-prototyping, and object-oriented system design provide a suitable approach to evolutionary system design. Since the design of complex machine vision systems is typically evolutionary in nature, often requiring considerable experimentation before the evolution of a final prototype, the machine vision prototyper proposed in this work will be based on these concepts. However, to design a machine vision prototyper one must first understand machine vision system design requirements. For this reason the next section will discuss the major components of a typical machine system.

2.3 Components of a Machine Vision System

The major components of a typical machine vision system include imaging subsystems, object detection systems, computer interfaces, a materials handling system, and a computing resource. Hence, any machine vision prototyper must have a number of these same components available. In this section, we will examine the choices that must be made in selecting these components for solving a machine vision problem. It will be seen that the breadth and complexity of these choices motivates the development of a machine vision prototyper. The goal of this prototyper is to provide a way of handling the complexity of making design choices.

2.3.1 The Imaging Subsystem

The imaging process is facilitated by a suitable choice of sensors, illumination sources, lenses, filters, and a suitable system for mounting these components. Setting up the various components is often a complex task. Furthermore, different planar web applications may require different setups. An important requirement is a mounting system that will allow one to experiment with different imaging geometries.

A wide variety of sensor technology exists, each providing its own output signals as a function of the physical variable that it is sensing. One of the most frequently used sensor in machine vision systems is the camera that creates images of objects in the visible light portion of the electromagnetic spectrum. Cameras are either of the electron tube or solid state variety (Galbiati (1990)). The disadvantages (Galbiati (1990)) of electron tube cameras are its relatively shorter life as compared to solid state cameras, its low speed of response resulting in image lag, and its susceptibility to image burn from over intense radiation. Electron tube cameras are also heavier, larger, and produce more heat than solid state cameras. They are affected by magnetic fields, suffer from geometric distortion, and are sensitive to vibration. They also tend to exhibit a greater spectral sensitivity towards the blue to ultraviolet portion of the spectrum. These cameras would be needed should ultraviolet imagery be desired. Solid state cameras do not have the disadvantages of electron tube cameras listed above. They tend to be light and rugged and consume little power. The spectral response of these cameras are, however, highest in the infrared region of the spectrum and may require infrared rejection filters especially when color images are desired (Chien (1975)).

Solid state cameras are either of the "linear array" or the "line scan" variety. Here we must consider the nature of the imaging problem in making the choice between camera types. Linear array cameras typically require that objects to be imaged be stationary. This is because the full object must be in view before imaging can be performed. A moving object can still be imaged with a linear array camera if the scanning speed of the camera is sufficiently fast, or an appropriate lighting method such as a stroboscopic light source is used. With strobe lighting, however, synchronization between the light source and camera adds further complexity in the imaging process.

Line scan cameras make scans of a thin line of sight from a stationary position. An object moving past this line of sight is imaged one scan line at a time. In this way, arbitrarily long objects can be imaged by line scan cameras. Furthermore, line scan cameras afford greater flexibility in determining image resolution as the resolution along a scan line can be fixed independently of the resolution in a direction perpendicular to the scan line.

An important aspect of a camera is its optical lens system. Four important parameters associated with optical lenses listed by Galbiati (1990) are its magnification, focal length, depth of field, and its mounting system. The first three parameters have to be adjusted according to the application while its mounting has to be matched to the camera being used. An important consideration in a lens is the quality of the image it forms. The quality of an image formed by a lens is affected by spherical aberration, coma, curvature of the surface being imaged, radial distortion, and astigmatism. With proper lens design, these effects can be minimized. A complete discussion of these effects is given by Price (1976).

Next to the choice of lens and camera is the choice of the illumination system. The first consideration is the choice of light sources. The types of lamps used most frequently in industrial environments are incandescent, fluorescent, mercury vapor, and sodium vapor lamps (Galbiati (1990)). An important consideration in the choice of lamps are its spectral characteristics. For example, tungsten halogen lamps that produce light in the visible and infrared portion of the electromagnetic spectrum are often used due to the ability of each lamp to sustain relatively little change in intensity for most of its life. Such lamps are very suitable for black and white imaging but will require some form of filtering should color imaging be required. An alternative to tungsten halogen lamps that may be considered are xenon arc lamps that have higher intensities, a flatter spectrum similar to the sun's, and longer lives than tungsten halogen lamps. They are, however, considerably more expensive than tungsten halogen lamps. It should be noted that the choice of lamps in machine vision systems need not be limited to lamps radiating light in the visible region of the electromagnetic spectrum. For example, the use of X-rays may be desirable for detecting internal defects of a planar web.

Filters are often used in conjunction with lenses and cameras to correct for differences in spectral response over the desired spectrum or for band-limiting the radiation. For example, for solid state cameras, due to the fact that typical silicon based solid state cameras are sensitive to infrared light, the appropriate infrared rejection filters have to be used.

Next to the choice of light sources is the type of lighting used to illuminate the desired objects. The various forms of lighting can be divided into four categories, namely, back lighting, front lighting, structured lighting and strobe lighting (Galbiati (1990)). In back lighting, an object is placed between the light source and camera. This method is often used for tasks like the detection and location of opaque foreign material in transparent objects. It can also be used to detect holes in opaque objects. In front lighting, light reflected from an object is viewed by the camera to produce an image of the object. Strobe lighting is often used for the illumination of moving objects by a short pulse of high-intensity light to freeze the motion of these objects. Finally, in structured lighting, illumination of an object is carried out with a special pattern. The three-dimensional shape of an object can be extracted based on the resulting pattern as viewed by a camera. Selecting a combination of light source, lighting method, and camera lens can be a difficult undertaking and has prompted the development of an expert system to help in making this choice (Novini (1990)). This expert system, however, is limited to only one small group of machine vision applications, that of small parts assembly verification. Even then, there were over three hundred rules in the system, a further indication of the complexity of the problem.

2.3.2 Materials Handling System

In order to image an object in an industrial inspection problem, there needs to be a system for transporting the object into the field of view of the imaging subsystems. The transport function is accomplished by a materials handling system.

Considerations such as speed of the transport system, the maximum load capacity, its accuracy in positioning an object, and the manner in which an object can be mounted to it, are important in selecting a materials handling system.

Perhaps the most important aspect of the transport system is the choice of actuators used to control it. An **actuator** can be viewed as consisting of a **transducer** and an **amplifier** (Lawrence (1988)). The transducer converts electrical energy into the required control function. In the case of electrical motors, for instance, they convert electrical energy into mechanical motion. The amplifier amplifies the control signals from a computer interface to a sufficient level for driving the transducer. In many cases, the transducer and amplifier is available as a single unit.

Electrical motors that provide mechanical motion for the transport of webs and whose velocity and position can be controlled are of two basic types, stepping motors and continuous direct current servomotors. Stepping motors apply electric pulses to their windings to achieve incremental motion. For this reason they provide precise positional control within the limits of each increment. The speed of rotation is easily controlled by determining the amount of time between successive steps of the rotor. The position can be controlled by determining the number of steps. These two factors make it easy for computers to control a stepping motor. DC servomotors on the other hand are more difficult to control as motor torque is entirely controlled by armature current. The velocity and position of these motors can only be controlled through the use of feedback. As a result, these motors are often equipped with velocity and feedback sensors.

2.3.3 Object Detection System

An important component of many industrial inspection systems is an object detection system that is used for determining the presence of a object. This is necessary to relieve the computer from continuously analyzing an image for the presence of an object. It also allows the computer to move the object close to a camera or sensor before starting to scan. The object detection system need not be elaborate. A simple combination of an infrared light source to generate a beam and a phototransistor to detect the presence or absence of the beam of light, depending on whether the object has broken the beam, is often sufficient.

2.3.4 The Computer Interface

Each sensor must have an interface for transferring its digitized images to a computer for further processing. The main functions of a computer input/output interface are twofold. Firstly, it must coordinate the transfer of data between the external device and the processor. Second, it must convert data from a form recognized by the external device, to a form recognizable by the computer. There are several ways in which the imaging system can be interfaced to the computer. For example, the interface may be made through direct memory access (DMA) methods. If the data transfer rate is not important, the interface may be made through the regular parallel or serial ports one typically finds on computers. To effect the transfer of data through the control of the interface, a device-driver must be written. A device-driver is a software entity used by an operating system to communicate with an external device through a system of signal protocols between the computer and device.

Apart from the interface for transferring digitized images to the computer, there is a need for the interfaces to the materials handling system consisting of the web transport mechanism and the object detection system for detecting the presence of a web. These interfaces usually do not have a great deal of data to be transferred at a high rate and, as such, a serial interface is often all that is required.

2.3.5 Supporting Software

Every machine vision system must have supporting software to perform a variety of functions. The supporting software includes system management software for controlling the computing system's resources, and image processing/computer vision software for analyzing the acquired images. The image processing/computer vision software will be discussed in the next two sections. This section will concentrate on the system management software.

The central controlling software on any computer is its operating system. An operating system is a collection of programs that coordinate the execution of user programs on a computer. It provides access to hardware resources to user processes and also allows these processes to communicate with each other. There exist a significant variety of operating systems ranging from simple ones like MS-DOS from Microsoft Corporation for microcomputers to complex systems like IBM's VM for large scale computers. The choice of an operating system depends on the type of computer and the applications. MS-DOS is an appropriate choice for personal computers that do not have applications that require large main-memories, or that do not require support for a multiuser, multitasking environment. IBM's VM on the other hand provides a great deal of support for these environments. There are also many operating systems suited to medium scale computing installations such as DEC's VAX VMS operating systems and AT&T's UNIX operating system.

2.3.6 Computing Resource

The next important component in a machine vision system is the computer and its supporting software. The choice of the computer is an open-ended question considering the multitude of computers in the market. Basic considerations include the CPU processing speed, the type of bus for mounting interfaces to external devices (choosing a computer with a standard bus available in a wide variety of computers enables a designer to reuse any interfaces designed for the bus, when upgrading or porting from one system to another computer), the amount of main memory, and so on. The availability of standard software components such as compilers, word processing software, etc., is also important as these may contribute considerably to the costs of purchasing the computing resource.

When making the choice of a computer, the nature of the inspection problem is usually an important factor. For example, long planar webs of up to several feet in length requiring high spatial resolution imaging may require CPU speeds of several hundred million instructions per second (mips). Furthermore, the size of the web and the required image resolution will determine the amount of main memory necessary. If a number of these images are to be stored for further offline analysis, this would also necessitate the availability of a large disk storage capacity.

Finally, the computer should be a general purpose computer as special purpose machines do not provide the flexibility required for rapid prototyping. Special purpose machines, by definition, have hardware specially designed to implement specific algorithms. As the number of computer vision algorithms is very large without any unifying theory, and since developing computer vision algorithms is more an art than a science, the general utility of a specialized implementation of any set of computer vision algorithms, is questionable.

2.4 Computer Vision Systems: A Theoretical Review

An important aspect of machine vision systems is the computer vision software for the analysis of images. In this section we review some computer vision methodologies and show that the selection of suitable techniques can be quite difficult given the plethora of available techniques.

Computer vision systems can be divided into two basic components. The first component, which will be called the image analysis component, basically segments an image into uniform areas and extracts important information or features from these segmented areas for further processing. The second component, called the scene analysis component uses information provided to it by the image analysis component to interpret the scene within the image. Pavlidis (1977) states that:

> "We shall use the term 'scene analysis' to denote the assignment of labels or interpretations to the results of a segmentation algorithm. For example, we may decide to call a region the 'right lung' in a chest x-ray picture, or a set of regions a 'highway' in a satellite photograph."

2.4.1 Image Analysis Methods

The purpose of an image analysis system is to extract features from a digital image and then to segment this image into regions which are uniform with respect to one or more of the extracted features. These features are obtained by applying operators to the image. As the scene analysis system depends on information provided by the image analysis system, it is important that the segmented image be as noise free as is possible. Image analysis strategies that perform image segmentation can be divided into three types of strategies. The three types of strategies are called the local-to-global, middle-out, and global-to-local strategies. These names are used in order to be consistent with terminology used by perceptual psychologists.

Local-to-global processing involves the formation of regions by combining together small areas which have similar properties. There are two basic ways we can go about doing this. The first involves the grouping together of points within the image that have the greatest difference in gray levels. The second method involves merging together points that have similar gray levels. Both these methods imply the use of operators whose domain of definition are small areas. Much of the earliest work in computer vision made use of these strategies. For example, Roberts (1965) in his block's world environment employed a simple gradient-based edge detector to segment simple scenes into lines. Edge detectors correspond to the first method of local-to-global analysis. In 1970, Brice et al. (1970), introduced the concept of region growing which is basically the grouping of similar points into regions. Local-to-global methods were, however, found to produce noisy outputs and, therefore, are difficult to use.

At the other extreme of image analysis strategies lies the category of global-to-local methods. In global-to-local processing, large global regions are formed first followed by the formation, if necessary, of smaller regions within these global regions. An example of such a strategy is the pure-split procedure (Pavlidis (1977)).

The initial large area in this case is the whole image. Midway between the local-to-global and global-to-local strategies are the middle-out strategies with "medium" sized regions being formed first. An example of this strategy is the split-and-merge procedure described by Pavlidis (1977).

These last two categories of image analysis strategies were found to be relatively less sensitive to noise than in the local-to-global strategies. However, because their domain of definition are over relative large areas, a rich descriptive vocabulary is needed for describing the greater number of characteristics within a large region as compared to small regions. The need for a richer descriptions of segmented regions, irrespective of the image analysis strategy used, has been recognized for some time (Conners (1989b)).

Regardless of the strategy used, there are two basic approaches used to implement these strategies. The two basic segmentation approaches are edge-based and region-based. In edge-based approaches, local discontinuities are detected to form boundaries around regions while in region-based approaches, homogeneous areas are first found which then give us the boundaries (Nevatia (1986)). Several problems are inherent to edge-based approaches. Edge operators that are able to find most edges also respond to noise while edge operators that are noise-insensitive often miss some crucial edges (Ballard (1982)). Furthermore, building scene descriptions from edge images of complex scenes can be a nightmarish task.

A good survey of region-based image segmentation techniques is given by Haralick and Shapiro (1985). Region-based techniques include the various forms of spatial clustering and measurement space guided spatial clustering. Among the most popular and simple to implement procedures are ones listed under measurement space guided spatial clustering. In this category, each pixel in an image is assigned a label representing a point in measurement space. The various points in measurement space are then clustered to separate points belonging to different objects. An example is the simple method of histogram thresholding. In this method, a histogram (a probability density distribution) is made of the gray levels of pixels within an image. Peaks in the histogram represent objects and thresholds are set in the valleys between peaks. A good survey of thresholding techniques using different measurement spaces is given by Sahoo et al. (1988). Measurement space guided techniques are basically global-to-local segmentation techniques in which the global properties of pixels are analyzed in measurement space and used to guide the segmentation of the image.

An aspect of region based segmentation are the texture based segmentation techniques. A texture can be regarded as a generalization of a region where the region need not necessarily be characterized by uniformity in gray level values. There are basically two approaches to texture analysis, namely, the structural and statistical approaches (Haralick (1979)). The structural approach regards texture as a global pattern arising from a repetition of local subpatterns arranged together according to certain placement rules. The structural approach to texture analysis is based on locating and describing these local subpatterns and determining the rules for their placement. The statistical approach describes texture through statistical models that describe the spatial distribution and relation of gray levels. The structural approach is more suited to the analysis of regions that have a well characterized structure such as a periodic pattern. The statistical approach is more suited to the analysis of regions that are more random in nature especially when the textural elements or primitives are relatively small. The textural approach (whether structural or statistical) to image analysis is a powerful but very computationally intensive approach. As such, texture based methods may not be suitable to many real-time applications.

Unfortunately, all the image analysis methods suggested to date have difficulty segmenting images into visually meaningful regions. In some cases they may over fragment an object, i.e., an object may be fragmented into several different regions, regions which, to a human observer, do not correspond to any visually recognizable components. In other cases, regions that contain separate objects may be merged into a single region. On some scenes, region based operators work better. In other scenes, edge-based methods work better. It is difficult to know which technique might work best in any given situation. And, as might be suspected, the results obtainable by the image analysis system markedly affects the design complexity and accuracy of the higher level scene analysis methods.

Given the importance of the scene segmentation, two basic approaches have been taken to assure that the highest quality segmentation is achievable on a particular application. The first approach that was considered by the research community was to develop general purpose image-processing systems. These systems provide a relatively convenient method for exhaustively trying a number of image analysis methods and for displaying the results at various stages of processing so that the results can be evaluated. A number of such systems have been created and includes GIPSY (Garland (1986)), RIPS Perceptics (1986)), and ELAS (Beverly (1989)).

A second approach has been to attempt to unify the various methods available into an expert system. This expert system uses various heuristics to determine which algorithm should be applied to various scenes or to various parts of a scene. To date two such expert systems have been developed (Nazif (1984), Meisels (1988)).

The expert system for low-level segmentation described in (Nazif (1984)) and (Levine (1985)) unifies the processes of edge detection and region growing with their various heuristics into a single system. In the system, the input image, segmentation data and at the end of processing, the output, are stored in the short term memory (STM). The long term memory (LTM) contains the knowledge about low-level segmentation as well as control strategy. Various system processes such as a line analyzer, region analyzer, focus of attention, etc., match rules in the LTM against the data stored in the STM. When a match occurs, the rule fires. This triggers an action that is usually performed on the data. A weakness of the above system is that it was designed as a general purpose segmentation system without the use of any scene dependent contextual information. The system of Meisels and Bergman (1988) seeks to alleviate such a weakness by the incorporation of scene dependent contextual information. It does so by keeping a list of simple "geometric" objects that it can look for in a particular type of scene.

Unfortunately, expert systems for algorithm selection can be very large and complex themselves, depending on the breadth of applications they attempt to address. There is also a concern about the ability of heuristics to make the best selection of algorithm to employ. If truly robust heuristics were available, they could be used by an investigator without the need of an expert system.

2.4.2 Scene Analysis Methods

In this dissertation, *scene analysis* refers to the process of assigning labels or interpretations to the results of a segmentation (Pavlidis (1977)). Unlike the image analysis strategies discussed in the previous sub-section, scene analysis depends a great deal on domain specific information. For example, in order to label the different objects in a scene, we need to have knowledge of these objects such as a list of expected objects, a model of each type of object and expected locations of the various types of objects.

Scene analysis strategies can be classified into top-down, bottom-up or combination categories. Top-down strategies are basically goal-directed and contain no data-directed procedures. Such strategies involves making hypotheses as to what objects are in the scene being analyzed. Scene analysis operators are used to verify if a hypothesis is true. If found to be false, a new hypothesis is generated, independent of previous processing. The advantage of such a strategy is that known characteristics of the various objects can be used to generate heuristics that can help locate the objects in noisy segmented images. Such heuristics are more easily found in, and usually only applicable to, highly structured scenes.

Bottom-up procedures on the other hand, starts with the input data. Operators are used to extract information from the data to form a richer description to be used at the next level of analysis. Finally, when a rich enough description has been reached, these descriptions are matched to models of known objects. The advantage of bottom-up strategies is that no prior knowledge of the scene is required and is thus applicable to any type of image. The disadvantage is that any noise introduced at any step along the analysis propagates on to subsequent steps.

Combination strategies can combine the best characteristics of both bottom-up and top-down strategies. Such strategies usually begin by using bottom-up procedures to generate a list of possible objects. Top-down procedures could then use this list as a set of hypotheses to be verified. The object of this strategy is to limit the number of hypotheses that needs to be verified and thus to reduce the computations involved. A further discussion of all three strategies can be found in (Ballard (1982)).

An important aspect of scene analysis methods is the form of knowledge about scene objects. It is this knowledge about scene objects that allow the recognition process in scene analysis to take place. While simple scenes do not require much knowledge to be stored, complex scenes often require extensive knowledge about the objects to be recognized in a scene. When this is the case, knowledge representation schemes become an important issue. Examples of knowledge representation schemes are semantic nets, frames (a type of object-oriented representation scheme), production systems, and first-order logic (Rao (1988)). Semantic nets basically represent objects and relationships between objects in the structure of graphs of nodes and arcs (Ballard (1982)). These scheme is well suited to the object-oriented view of systems. Frames represent knowledge in the form of data structures containing information about a situation or object. Attached to each frame is a quantum of knowledge about the situation or object. Frames can be designed around an object-oriented scheme where each frame represents an object. They can also be combined into a semantic network representation where each node is a frame. Another way in which knowledge can be stored is in the form of production rules as can be found in production systems. These rules often take the form of a **rewriting rule** such as $A \rightarrow BC$ which means "given A, rewrite A as BC." These rules can be implemented in computer programs in the form of an **IF-THEN** statement. First-order logic represent knowledge in terms of predicates. A predicate is a function that returns a true or false value and is basically used to describe some property of an object. First-order logic, however, suffers from the fact that it cannot represent imprecise or fuzzy knowledge. With the increasing complexity of inspection tasks being performed systems that incorporate some form of knowledge base are becoming necessary.

2.5 Existing Computer and Machine Vision Systems

Designing machine vision systems has been shown to be quite a complex task. To provide greater insight into the machine vision design process this section will discuss some existing computer and machine vision systems. The systems to be discussed cover a wide variety of applications such as industrial inspection, autonomous vehicle navigation, medical image analysis, aerial image interpretation, and hardwood lumber inspection. While some of these systems are fully integrated hardware and software systems, many concentrate on the computer vision aspect, deemphazing the hardware aspects such as the image acquisition methodology. Those that are purely computer vision systems are usually non-real time systems. They attempt to be more robust through the use of knowledge-based methods.

There are two goals in studying these computer and machine vision systems. The first goal is to show that while most existing industrial inspection systems are specialized in nature, and not well suited to complex tasks or changes in requirements in these tasks, there is a trend towards incorporating knowledge-based methods in these systems to increase their robustness. These systems, however, do not incorporate very sophisticated knowledge-based methods. As such, a consideration of computer and machine vision systems in other areas of application are considered. The intent is to show that knowledge-based methods can be found in many areas of application of computer and machine vision systems. This being the case, it is important that the machine vision prototyper is able to support the design of knowledge-based systems.

The second goal is to demonstrate that, in spite of the large selection of available methodologies, these computer and machine vision systems have many common characteristics. These common characteristics are the ones associated with the different aspects of machine vision system design described in sections 2.3 and 2.4. Knowledge of these common characteristics is useful in designing the machine vision prototyper as it provides a general framework on which to develop the prototyper.

2.5.1 Industrial Inspection Systems

There have been a great number of automated visual inspection systems designed in the past twenty years (Chin (1982), Chin (1988)) that perform such tasks as the inspection of printed circuit boards (Benhabib (1990)) to the detection of surface defects in wood surfaces (Conners (1989a)), cast metals (Okawa (1984)), and apples (Rehkugler (1986)). The goal of automatic inspection systems is to locate and identify objects within image data. Once these objects are detected, functions such as sorting or detection of anomalies within the objects, are performed. One place object specific knowledge is used is in the object models used by a matching process during object identification. Another way object specific knowledge is used is in the direct detection of defects by subtraction where a suspect image is subtracted from a perfect image (Chin (1986)). Most automatic inspection systems are designed for operation in well-controlled environments with simple tasks to perform. They are usually high-speed, special-purpose systems which are specially designed to perform a specific inspection task (Chin (1988)). They typically represent *ad hoc* approaches to the solution of a single, very specific, problem (Chin (1986).

As was discussed in Chapter 1, flexibility and robustness are important requirements for vision systems (Chin (1986)). There have been recent attempts to improve the robustness of automatic inspection systems through the incorporation of rule-based systems in the inspection process. Bartlett et al. (1988) and Darwish et al. (1988) for example, apply the use of rule-base expert systems to printed-circuit-board inspection. One system (Bartlett (1988)) is a simple rule-based system that is still in developmental phase. In their approach, a manual registration of solder-joint images is performed and perfect registration is assumed. Features are extracted from a subimage and defect detection is performed by two approaches, a statistical pattern recognition approach, and an expert systems approach. This can hardly be considered a fully automatic system due to the manual registration of images that is required. Another system (Darwish (1988)), uses a rule-based system in which knowledge is stored in a form based on semantic networks and frames. Their approach consists of binary thresholding, conversion of the binary image into a line drawing, and a semantic network constructed from the line drawing. This semantic network is then compared with a stored model by a rule-based system to detect defects. One of its weaknesses is that binary thresholding is used in its segmentation, a weakness that is common to many industrial inspection systems (Chin (1986)). Another example of the incorporation of a rule-based system into an industrial inspection is the automated inspection of thin film disk heads reported by Sanz and Petkovic (1988). In this system, images of thin-film disk heads are first segmented by histogram based thresholding, followed by a connected component labeling and feature extraction. Finally, a rule-based system is used to identify defects based on these features that have been extracted.

The common characteristics demonstrated by all three systems are:

1) Image acquisition is first performed.

2) Segmentation and feature extraction follows image acquisition.

3) A high-level knowledge based scene analysis system is used to detect defects based on the segmentation and features extracted.

All three systems, however, are still application specific and are therefore not flexible.

It would seem that the trend is to use knowledge based intelligent automatic inspection to solve inspection problems. The reason for incorporating intelligence is to provide the flexibility and robustness for solving complex problems encountered in inspection tasks. The use of knowledge based methods, however, is not limited to the computer vision system in a machine vision system. For example, in one printed circuit board inspection system (Benhabib (1990)), an adaptive illumination system is employed. This system is able to automatically change the intensity of its illumination sources, for example, to suit a wide variety of printed circuit boards.

2.5.2 Autonomous Vehicle Navigation

Machine vision systems for the navigation of autonomous vehicles are fine examples of the complete integration of hardware and software systems. Two of these systems have characteristics relevant to the design of the machine vision prototyper. Dickinson and Davis (1988) describe an expert vision system for Autonomous Land Vehicle (ALV) road following. The system uses a top-down object hypothesis generation and bottom-up object hypothesis verification methodology in accordance with the hypothesize-and-test paradigm. Initially a scene model is built from images acquired from a camera. This is done by a subsystem called the Planner which generates object hypotheses with the help of a road map and sends these hypotheses to another subsystem called the Verifier which then returns verified objects to the Planner. Objects (such as roads and telephone poles) are represented as networks of frames constructed by a modular system of communicating blackboards. Each blackboard, implemented by a production system, controls the definition of a single object class. Based on the scene model the ALV plans a course and moves through the environment following the course plan. This system demonstrates the utility of a mixed top-down and bottom-up control strategy in a scene understanding system with all the advantages described in Section 2.4.2. The system is still being improved upon and the authors do not provide any information as to the speeds with which their ALV can move.

VITS (Turk (1988)) builds symbolic descriptions of objects in the environment using both video and range sensors. The vision system basically operates in a bottom-up manner with some aspects of the vision process being top-down in nature. The vision system first performs a segmentation of the scene. Color images are segmented using based on clusters formed in a two-dimensional histogram computed from the red and blue color bands. A scene model is then constructed. The scene model is a description of the observed road. The scene model is then sent to a reasoning subsystem whose job it is to evaluate the scene model and to navigate, based on a plan script received from a human test conductor. Another function of the reasoning subsystem is to determine if a visually identifiable feature is within the field of view based on the information stored in a knowledge base. These visual cues, are used to help determine the form of vision processing.

Though these ALV systems are also application specific they again demonstrate the common characteristics of a complete machine vision system in that 1) they are composed of an image acquisition system, a low-level segmentation and feature extraction system, and a high level scene analysis system, and 2) they employ knowledge based methods in their scene analysis system.

2.5.3 Medical Image Analysis

A knowledge-based system for obtaining a complete diagnostic description of an image sequence taken in nuclear medicine from the human heart is described by Niemann et al. (1985). In this system, images are first smoothed and organ contours extracted. The result is used as the input for high level scene analysis system. Knowledge in the form of both declarative and procedural knowledge is stored in a semantic network. Conclusions are made by a production rule system. This is basically a bottom-up system.

Again, we notice the partitioning of the system into an image acquisition system, a low-level image analysis system, and a high level knowledge based system. This system is, however, a non-real time computer vision system. The image acquisition system is not directly coupled to the vision system.

2.5.4 Aerial Image Interpretation

The next three computer vision systems again demonstrate the use of knowledge based methods in their scene analysis subsystems. The first, a knowledge based vision system for the analysis of aerial images is reported by Nagao and Matsuyama (1980). The system begins analysis by applying edge-preserving smoothing to remove noise and then segmenting the image based on multispectral properties. Following this, regions with prominent features (called characteristic regions) are extracted. A set of object detection systems analyzes specific local areas made of some characteristic regions and checks for the existence of specific objects. The detection of objects by the object detection systems are performed by both data-driven and model-driven analysis. All information on recognized and unrecognized objects, characteristic regions and the elementary regions from which these characteristic regions are made up of, are stored in a common database known as a blackboard. Each object detection system checks the blackboard for their activation and enters the results of any analysis they perform. The object detection systems communicate with each other through the blackboard. Binford (1982) lists several weaknesses with the system. The smoothing operation tends to erode thin lines and small regions. Furthermore, assumptions made in the interpretation are not generally useful. For example, water is assumed to be darker then adjacent land. This is not always true as water can be brighter or darker than surrounding land. These weaknesses demonstrate the need for evolutionary system design. When such weaknesses are found, machine vision systems must be allowed to evolve to overcome these weaknesses. Note also that this system has a low level segmentation system and a high level scene analysis system.

ACRONYM (Brooks (1981), Brooks (1983)), is a model-based vision system that uses viewpoint independent models of objects. These objects are represented as generalized cylinders. An edge detector is used to derive ribbons and ellipses from input images. These ribbons and ellipses represent a two-dimensional projection of the boundaries of the three-dimensional generalized cylinder. Object recognition is performed by matching these ribbons and ellipses with object models. This is done in a top-down manner. The main weakness in this system is its dependence on noisy edge-detection based segmentation that sometimes results in spurious ribbons being extracted. Furthermore, it would not be able to handle highly textured regions very well due to its dependence on edge-detection.

SIGMA (Hwang (1986)) is a framed-based image understanding system. Object models are represented in a semantic network with each node described by frames. This system uses both top-down and bottom-up inferencing control to achieve a consistent match between an input image and an underlying scene model. The system begins analysis by first performing an initial segmentation of the image to form a set of image structures. The high-level system of SIGMA then constructs a partial interpretation based on the initial segmentation. During this process the high-level system may direct the low-level system to find more image structures as needed. A query-answering model is then used for selecting "good interpretations" from the set of partial interpretations, and to display the reasoning used to achieve these interpretations. As is the case with most aerial image understanding systems, objects are fairly well-defined and relatively easy to model. Such systems based on well-defined object models are, however, not suitable for many web inspection problems where defect do not have a consistent shape or size nor do they always have definite relational constraints imposed on their location.

All three of these systems demonstrate a partitioning of the computer vision system into a low level image analysis system, and a high level scene analysis system. Furthermore, they incorporate sophisticated knowledge based methods within their high level scene analysis system. These systems were designed without the ability to adjust image acquisition parameters. This is in fact a major weakness of all computer vision systems. Without the ability to control the image acquisition, the computer vision system typically has to use much of its computing power to preprocess images to make them suitable for analysis. While this may be unavoidable with aerial interpretation systems, for example, real-time systems for solving complex inspection problems do not have this luxury.

2.6 Summary

In this chapter, the basic concept of rapid-prototyping was introduced as a method of system design to be utilized by the machine vision prototyper. Rapid-prototyping provides the prototyper with a mechanism for designing machine vision systems. Given a suitable system design mechanism, the next step is to provide the necessary components needed by machine vision systems. To design such components, a knowledge of machine vision system technology is necessary. As such, a discussion of the major components of a typical machine vision system was provided. To further acquaint the reader with machine vision system technology, a brief discussion of some existing machine vision system was given. It was found that existing machine vision systems for industrial inspection were by nature rather simple and inappropriate to the task of inspecting more complex planar webs. For this reason a discussion of some knowledge based machine vision systems was given to demonstrate the utility of the knowledge-based approach for solving complex problems. The next chapter will provide an overview of the machine vision prototyper.

Chapter 3

OVERVIEW OF THE THE MACHINE VISION PROTOTYPER

Chapter 2 provided a survey of the extensiveness and complexity of current machine vision technology. From the survey, it is clear that there is currently no consistent theory available for designing machine vision systems. Rather, a great deal of experimentation is needed when designing such systems. Therefore, an important objective of this research was to create a vehicle that will allow the types of experimentation usually associated with the development of machine vision systems, to be facilitated. This vehicle is called the "machine vision prototyper." This machine vision prototyper was designed to address planar web inspection problems. The objective of this chapter is to provide an overview of the "machine vision prototyper", and to describe the fundamental methodologies involved in the utilization of the prototyper.

3.1 Rapid-Prototyping and Machine Vision System Design

Chapter 2 lists 1) an imaging subsystem, 2) a materials handling system, 3) an object detection system, 4) computer interfaces, 5) supporting software, 6) a computing resource, as the major components of a machine vision system. To design a machine vision system, all of these components must be considered in the design. Unfortunately, as has been discussed in Chapter 1, choosing some of these components is a difficult task, a task that may require some experimentation. To design a "flexible" machine vision system for attacking a number of problems, the choice is made even more difficult as considerable experimentation may be needed to achieve an acceptable solution.

A way in which this experimentation can be facilitated is to construct partial implementations with each implementation providing components that can be reused in the next implementation. A typical procedure that can be followed to perform this method of experimentation is shown in Figure 3.1. This flowchart does not necessarily represent a procedure that has to be followed. In the box labelled "Problem Definition" the statement of the machine vision problem together with the derivation of an initial set of requirements are formed. This set of requirements need not be complete but should at least include all the important components. The next stage takes the problem definition and configures an initial set of hardware and software components. These components may be existing hardware or software components. This is a very important stage as it would be the first opportunity to consider the extent to which hardware or software is responsible for solving different parts of a machine vision problem. Sometimes, however, some of these components are "abstracted." This means that the functions of these components are described but not initially implemented. For instance, before a sensor is chosen, the types of image data to be provided is first described. If such image data already exists, other parts of the machine vision system can first be designed to analyze these images. The choice of sensors and imaging methods can then be decided on later.

The procedure then branches into separate hardware and software prototyping procedures. In these two steps, the hardware and software units are configured

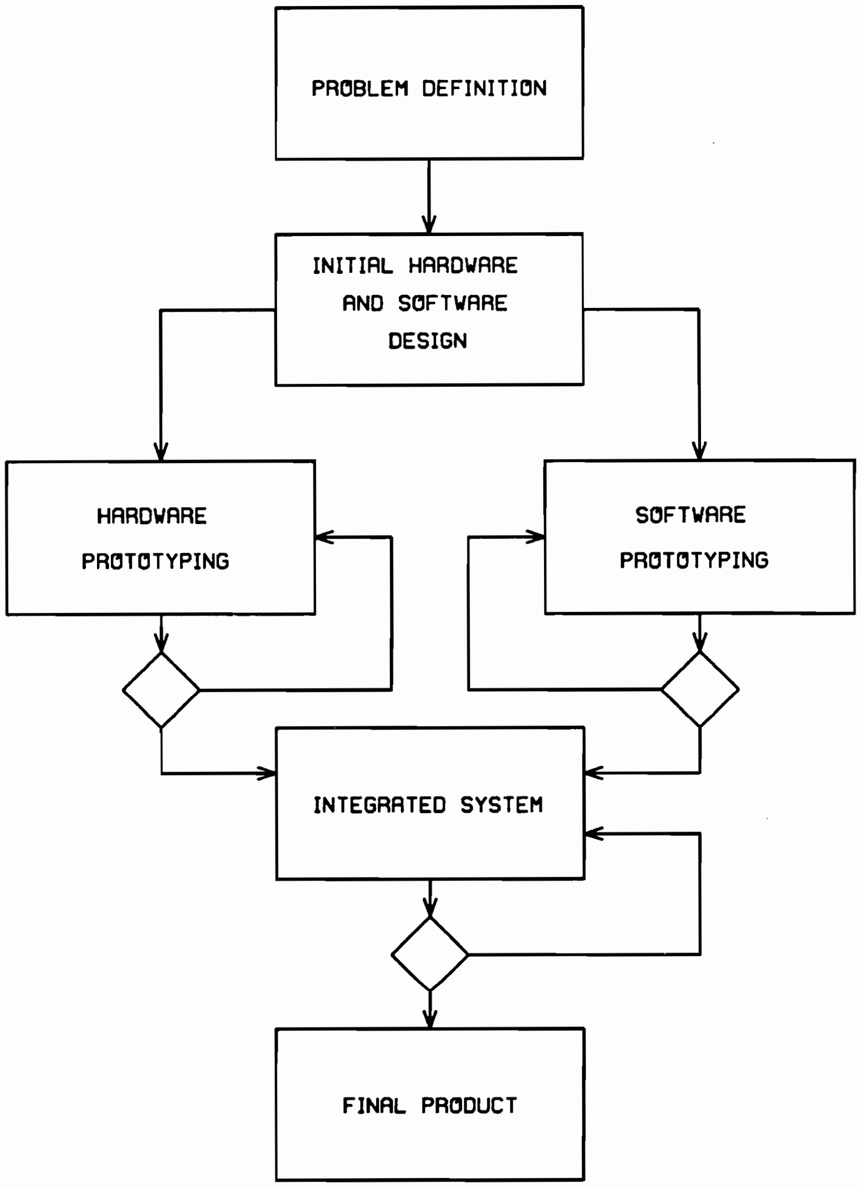

Figure 3.1. Example of a typical rapid-prototyping procedure. Each diamond shaped box represents a decision point in a prototyping cycle.

and tested to see if they perform according to the initial requirements. Any problems or considerations that surface at this stage can be used to redefine the initial requirements and cause further iterations in the hardware or software prototyping cycle. It is during the prototyping cycles that experimentation is performed. During such a cycle, a configuration of components is tested. If this implementation does not meet the requirements, reconfiguration of these components or the design of new components may be necessary. This process continues until a satisfactory implementation is found.

Though hardware and software prototyping appear to be two independent efforts, they sometimes are not. For example, if the computer vision algorithms suitable to a particular type of sensor are found to be unsuitable for attacking a problem, one might consider the use of other types of sensors. Furthermore, a temporary implementation of the hardware may be provided to facilitate the software prototyping. For example, before a complex high speed interface to a sensor is completed, a simpler parallel port interface with a slower data transfer rate may be designed and used temporarily to test the computer vision software being prototyped. When it appears that suitable choices of hardware and software components have been made, these components are integrated into a complete system for further experimentation and testing. Tests include testing the accuracy of the machine vision system, and determining the speed with which it performs it's tasks.

In addition, if it is required that each partial implementation of a machine vision system be done quickly, the number of hardware or software components that are available must be relatively large. Furthermore, these components must be reusable. It is assumed in this dissertation that the reusability of these components will evolve as they are applied to more problems.

This evolutionary form of machine vision design is based on the concepts of rapid-prototyping. In rapid-prototyping, no formal procedure exists. The main idea behind evolutionary rapid-prototyping is to first construct partial implementations of a system which meet known requirements, or a subset of these requirements. Each prototype implementation is then used to help understand the full requirements, or to identify requirements yet unknown. This process implies a great deal of experimentation. Furthermore, to be able to quickly design a new prototype, some, or all of the components of a previous prototype should be reusable. They should also be extensible, i.e., each of these components should be easily changed in order that they can evolve with usage, and easily reconfigurable in order that they be easily reused in the design of a new prototype.

For the design of "flexible" machine vision systems, support must be provided for both hardware and software prototyping. A machine vision "prototyper" must provide a platform on which to quickly configure hardware and software components, to test these components, and to quickly design new components if needed. It would also be of great help if these hardware and software components can be treated in the same uniform manner. Being able to treat hardware and software components in the same uniform manner simplifies the design process. In treating them in the same uniform manner, we have to hide the implementation details of a particular component. Not having to deal with implementation details allows a developer to concentrate on the actual functions that each component in a machine vision system has to perform. This reduces the level of effort required in understanding a complex design and thus simplifies the design process. Furthermore, some software functions may be implemented in hardware, and vice versa. Therefore, treating them in the same manner seemingly makes a great deal of sense.

3.2 Requirements of the Machine Vision Prototyper

The machine vision prototyper is a system that will be used to aid the development of machine vision systems for attacking a number of planar web inspection problems. The objective of the prototyper is to provide the experimental vehicle needed to design such systems. The machine vision prototyper can be defined to be a set of hardware and software units that can be combined in a sufficient variety of ways in order to allow the consideration of different solutions to a particular inspection problem. For such a prototyper system to be of greatest utility it should be able to derive solutions that are as close to the final solution as is possible. Instead of being limited to a fixed hardware and software configuration forcing a "particular" solution onto a problem, it should also be of sufficient generality to accomodate a variety of different solutions to a problem. For the prototyper, the goal is to facilitate rapid-prototyping in the design of machine vision systems. Note that this goal is not just confined to applying rapid-prototyping methods in designing a single machine vision system. The criteria used in designing the prototyper can be summarized as follows:

(1) The system should provide the experimental tools needed to consider different solutions to a planar web inspection problem. These experimental tools should allow one to test different solutions and to view the results of any test. In other words, the system should provide the experimental tools needed to

rapidly-prototype a machine vision system for attacking a planar web inspection problem.

(2) The system should provide the facilities to rapidly-prototype a machine vision system for new planar web inspection problems. These facilities include a database of reusable components from previous designs.

(3) The system must allow the design of machine vision components to evolve, not only with new technology, but also with the selection of new planar web inspection problems. This is an extension of rapid-prototyping. It means that rapid-prototyping should not end with the design of a final prototype.

To accomplish the first design criteria, the machine vision prototyper should support the hardware and software design activity required to rapidly-prototype a machine vision system. This activity includes the mounting of hardware components, experimenting with imaging geometry and illumination, viewing and analyzing images that have been scanned to determine their quality, and to display the results of applying different computer vision algorithms. Experimental tools must be provided to support such activities.

To accomplish the second criteria, the machine vision prototyper should provide a variety of reusable hardware and software components. The hardware components include a variety of different sensors and illumination sources. The software components include components that communicate with the hardware components, and components that perform the different computer vision tasks. Obviously, the number of these components should increase with the application of the machine vision prototyper to more inspection problems.

The support provided by the prototyper in accomplishing the first two criteria requires both hardware and software support and are summarized in section 3.3 and section 3.4.

To accomplish the third criteria, the machine vision prototyper should provide a system design environment that promotes evolutionary system design methods. Such a system design environment should provide support for the middle ground of design philosophies described in Chapter 2. This middle ground is that between designing "throwaway" prototypes, and the complete specification of system requirements followed by the design of a final prototype in a single step. Rapid-prototyping requires such an environment. Rapid-prototyping requires a database of previously designed components. Such components must be allowed to evolve to increase their utility. Furthermore, in rapid-prototyping, initial prototypes (partial implementations) evolve to prototypes that are closer to meeting the full requirements. A method of system design that allows previously designed components to easily evolve, and prototypes consist of these components to evolve is object-oriented system design. The object-oriented system design concepts utilized in the prototyper is described in section 3.5.

3.3 Hardware Components of the Machine Vision Prototyper

The basic hardware components of any machine vision system include:

1. one or more sensors,
2. imaging optics, e.g., band pass/rejection filters, lenses, etc.,
3. illumination sources,

4. interfaces to connect sensors to a computer,

5. a computer.

The goal of the machine vision prototyper is to provide a mechanism for rapidly prototyping machine vision systems. To facilitate hardware rapid-prototyping in designing machine vision systems, the prototyper should have a number of sensors, sensor interfaces, lenses, filters, and illumination sources in its "data base" of hardware. To rapid-prototype, the prototyper must have the capability to mount hardware components, and to experiment with different configurations of these components. For the planar web inspection problem, the prototyper must also provide the capability to transport planar webs, and to detect the presence of a web.

In addition, the prototyper must be able to support software development and to provide a mechanism to rapidly prototype alternative hardware solutions to a planar web inspection problem.

To support software development, the prototyper must have

1. image display capability, preferably a high resolution 24 bit full color display capability,

2. the ability to make hardcopies of images, and the results of the processing of these images,

3. and a secondary storage capability.

To provide for most of the above capabilities, the hardware components of the prototyper consists of

1. a basic platform for mounting hardware components, for transporting planar webs, and for detecting the presence of a web.

2. a database of reusable modules,

3. A color monitor for displaying images, a laser printer for obtaining hardcopies of images, and a plotter for drawing lines such as outlines of webs and outlines of defects in these webs. The laser printer is also able to draw these lines.

4. A computer.

The hardware rapid-prototyping process requires the ability to experiment with different sensors and illumination sources, experiment with different imaging geometries, and to move an object to within viewing range of a sensor. To support hardware rapid-prototyping, the machine vision prototyper must provide the means to mount hardware components, a materials transport system, and an object detection system. Chapter 4 provides a detailed description of the hardware components of the machine vision prototyper.

3.4 Software Components of the Machine Vision Prototyper

The basic software components of any machine vision system include

1. image and scene analysis software,
2. image processing software, and
3. device drivers for communicating with hardware components,
4. system software for operating image displays and hardcopy devices.

To facilitate software rapid-prototyping in designing machine vision systems, the prototyper should have a database of software components providing a variety of image and scene analysis operators, image processing operators, device drivers and other system software that provide the software services necessary for operating sensors, materials transport systems, object detection systems, image displays, and plotting devices. To rapid-prototype machine vision system software, the prototyper must provide an environment for developing the above software components and to provide the ability to experiment with different computer vision related software components, and the ability to easily configure the other software components.

The software design goals of the machine vision prototyper are thus to provide an environment for designing and testing reusable software components, and to create a database of these components. It is also the goal of the machine vision prototyper to provide a mechanism for treating hardware and software components in the same conceptual manner. The ability to treat hardware and software components in the same conceptual manner allows a software developer to treat both types of components as building blocks for configuring a machine vision system.

To achieve these three software goals the protototyper provides

1. an object-oriented design environment,

2. a database of reusable software components,

3. a kernel of support for low-level system functions and for facilitating object-oriented system design.

The software components of the machine vision prototyper is further described in Chapter 4. Understanding of these components also require a further background into the object-oriented system design concepts supported in this dissertation.

3.5 Object-Oriented System Design

To facilitate rapid-prototyping on the machine vision prototyper, an object-oriented method of system design is proposed, a method that has been found to be useful for rapid-prototyping (Mullins (1990), Luqi (1989)). In the rapid-prototyping approach to designing complex systems, one often begins by abstracting the various components and their functions. This allows the developer to concentrate on the important functional aspects of the design. This also frees a developer who wishes to utilize a previously designed object, from the responsibility of having to know how to operate on an object, allowing him to concentrate on what the object and its functions do. Object-oriented system design has been found to be useful in computer vision system design by a number of researchers (Cappellini (1990), Flickner (1990), Lawton (1990), Mohan (1988), and Zavidovique (1991)).

Object-oriented systems tend to be easy to use and portable due to the abstraction of underlying data (Stefik (1986)). In the abstraction of data, the functionality of the data is more important than the implementation details of the data. This abstraction can be extended to hardware objects. An example of this could be a sensor interface where a relatively slow 8-bit parallel port interface (found in IBM PCs and usually used as a printer port interface) to a sensor is designed. Hiding the implementation details of objects tends to reduce processing errors when the implementation is correct. The hiding of implementation details is known as

encapsulation. In object-oriented systems, data and functions (operators) are encapsulated into a single unit known as an **object**. Encapsulation has been used as a programming technique even in non object-oriented languages such as ADA (Booch (1983)). The purpose of information hiding is to make inaccessible certain details that should not affect other parts of a system (Booch (1986)). *With information hiding in an object-oriented environment, adding new program knowledge or removing unneeded facts (or changing a sensor interface) involves changing the independent object without affecting the rest of the system.* Only the response of the system to given sets of circumstances are altered. The maintenance and/or updating of the system would thus be greatly simplified. Such a system would make the re-definition of a removable defect a fairly straightforward process. Such a system also provides the ability to easily experiment with different components or algorithms.

In Chapter 2, Booch (1986) was quoted stating that the object-oriented approach to software development is an approach in which the decomposition of a system is based upon the concept of an object. System decomposition is concerned with the underlying description of a system in terms of its various components. It is an important aspect of design as it also represents the decomposition of a problem for which the system is intented to solve. Furthermore, knowledge of system decomposition and the ability to specify the decomposition can aid the system design process. It can also be used to help automate the prototyping process to whatever extent is possible.

Three different views of system decomposition have often been used. These are the data-flow view, the control-flow view and functional decomposition view (Lawrence (1988)). Though all three methods are widely used by system designers, proponents of object-oriented system decomposition are of the opinion that object-oriented system description provides an easier way of understanding system design (Coad (1990)). Variants of the three methods, however, have been used as tools for designing parts of object-oriented systems (Coad (1990), Luqi (1989)).

The basic unit of an object-oriented system decomposition is the object. Often, these objects can be grouped into different classes, and subclasses resulting in a hierarchical decomposition. A **class** can be defined as a collection of objects with one or more similar **attributes**. An **attribute** is a property or characteristic of an object. An **object** is an abstraction of an entity. Objects can be characterized by a set of data structures and operators. The state of an object is defined by its data structures which may be changed by the action of its operators on its variables. These operators are activated through a message passing mechanism (such as a function call) between objects.

A **subclass** of a class, which is also a class in itself, is a collection of objects which has some or all of the properties of its parent class of objects and also a set of properties not found in objects of its parent class. The properties of an object that are inherited from its parent class of objects are its **class inheritance**. Finally, objects may not always exist. When an object comes into existence, it is called an **instance** of an object.

Object-oriented systems are often developed through the use of object-oriented programming languages such as C++ and Smalltalk. These languages provide a high-level of support for object-oriented concepts (Cook (1986)). Object-oriented programming, however, is not limited to the use of object-oriented languages. Conventional languages have also been used, such as an object based interpreter written in the C language (Campbell (1991)) and a medical application program written in standard Pascal (Jacky (1987)). While object-oriented concepts are not strictly enforced by a conventional language, they may be enforced by the programmer. In doing so, the advantages found in object-oriented programming languages such as encapsulation (that promotes reduced-error programming), reusability, and extensibility (Micallef (1988)) may also be derived from the use of conventional programming languages. Encapsulation is the strict enforcement of information hiding, **reusability** is the ability of parts of system to be reused in the construction of new systems, and **extensibility** is the ease with which a software system may be modified in response to changes in its requirements (Micallef (1988)).

The programming language C is used for developing much of the software in the machine vision prototyper. There are several reasons for not using object-oriented languages. The complexity of some object-oriented languages may cause problems particularly in learning their use, as was found by Nielsen and Richards (1989) in learning to use the language Smalltalk. Furthermore, object-oriented languages tend to compile into slower executable code and may not be suitable in a real-time environment. Some of these languages are sometimes used only in the developmental phase of a system design. For example, Gray et al. (1988) notes that Smalltalk is often used commercially for "revolutionary" prototyping where the prototype is discarded and a final system generated from scratch using different development tools.

For the machine vision prototyper, reusable components correspond to objects. Objects may be hardware or software objects. When configured into a machine vision system, these components represent a decomposition of the machine vision system. Object-oriented systems can, however, be found in many different forms and implementations in the literature (Coad (1990), Blair (1990)). The most basic concepts that unify most of these object-oriented systems are the encapsulation of data and operators into objects, the grouping of objects into classes and subclasses, and class inheritance. In this dissertation, the term "object" applies to both hardware or software components. When specifying the decomposition of a system, no distinction need be made as the definitions may be applied to either software or hardware components, thus maintaining the generality and uniformity of descriptions. Real-world entities are usually represented in the object-oriented view by its variables and procedures in software. When so represented, the software object is merely a simulation of the real-world entity. For the machine vision prototyper, the hardware components are not simulated. Instead, these components represent themselves, not by some software data-structure. The corresponding procedures associated with these actual objects may be either software procedures that controls the state of these objects through the appropriate computer interface, or through manually operated controls.

Figure 3.2 shows an example of how object-oriented decomposition may be applied to the design of a machine vision system. In this figure, the object **Image Analysis Subsystem** belongs to the subclass of the class that **Imaging Subsystem** belongs to. The arc that links **Imaging Subsystem** and **Image Analysis Subsystem** indicates that **Image Analysis Subsystem** is an object belonging to a subclass of the **Imaging Subsystem** object. The subclass that **Scene Analysis Subsystem** belongs to is a class that inherits properties from the classes that **Image Analysis Subsystem** belong to.

Figures 3.3, 3.4, and 3.5 show a possible decomposition of the each of the objects shown in figure 3.2. Figure 3.3 in particular shows an example of a multiple

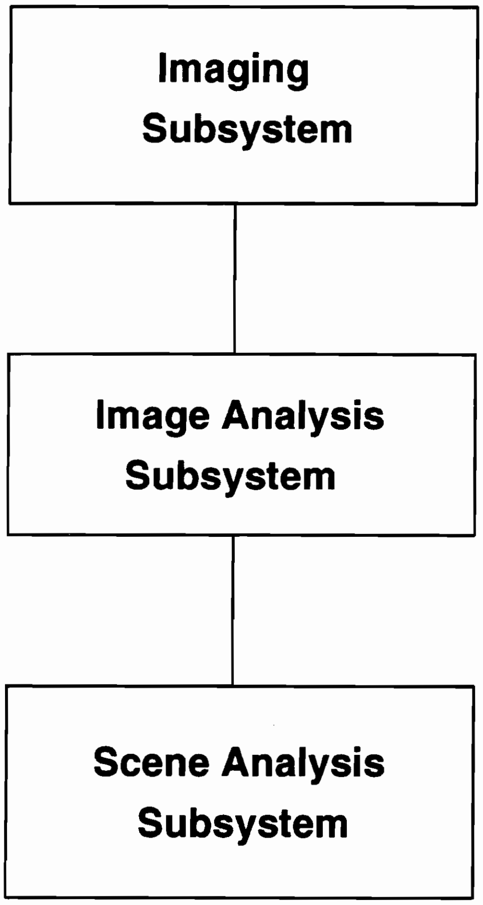

Figure 3.2. Example of the decomposition of a machine vision system.

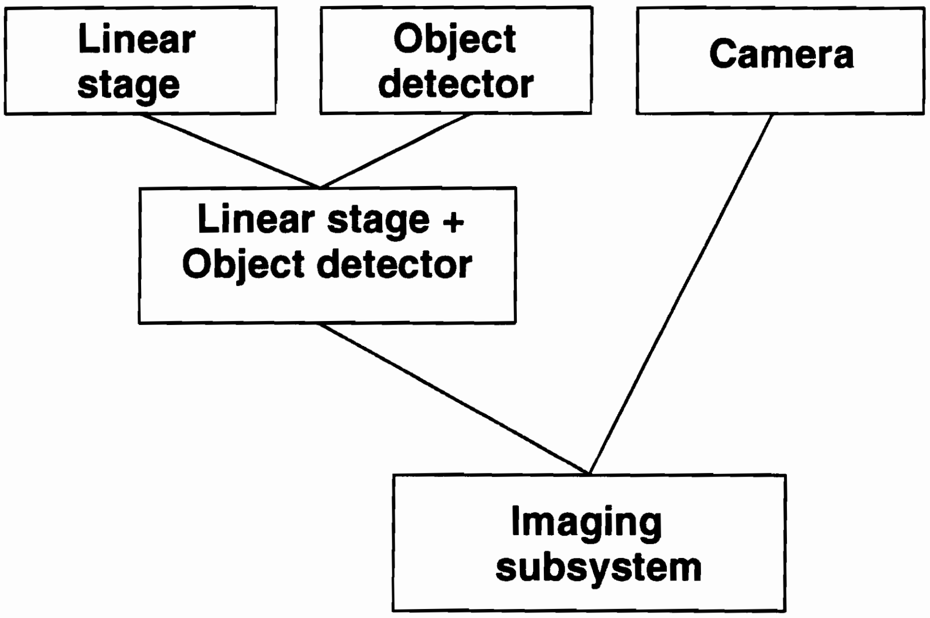

Figure 3.3. Decomposition of the imaging subsystem of Figure 3.2.

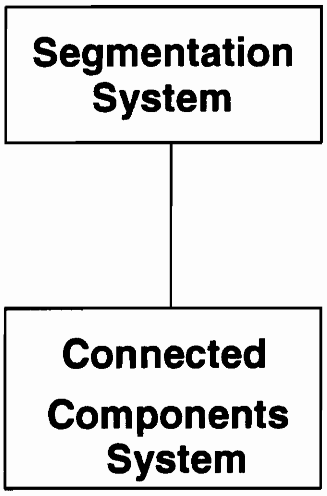

Figure 3.4. A possible decomposition of the image analysis subsystem in Figure 3.2.

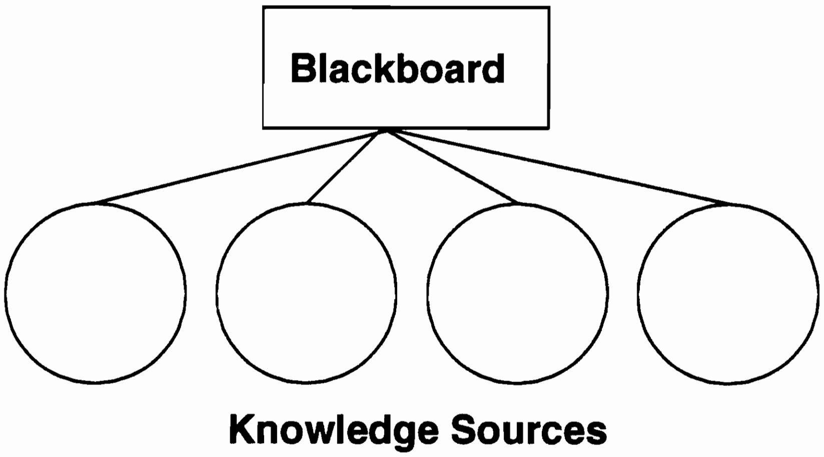

Figure 3.5. A possible decomposition of the scene analysis subsystem in Figure 3.2.

inheritance in which the **Imaging Subsystem** object inherits from (and can be considered to be composed of) the **Linear stage**, **Object detector**, and **Camera** objects.

Finally, in order for the components of a machine vision system system to be reusable, it must be functionally equivalent to components in the decomposition of other systems. To increase the match between components arising form the decomposition of different systems, the decomposition should conform, as much as is possible, to standard functions one would normally find in a machine vision system. Chapter 2 provided a background into the typical characteristics of vision systems.

3.6 Comparison With Other Work

We have seen in Chapter 2 that most machine vision systems designed for industrial inspection are highly specialized and inflexible systems. An important aspect of the approach taken in this work is the environment for designing systems that can be easily implement on "general-purpose" computer-based systems. Related to this aspect of the work are the large-number of general-purpose development systems for image processing and image analysis.

A large number of these developement systems for image processing can be categorized into three categories (Flickner (1990)). Included in the first category are the command- and menu-based systems such as KBVision (Williams (1990)), Image Analyst (Rosenthal (1990)), GYPSY (Garland (1986)), ELAS (Beverley (1989)), and RIPS (Perceptics (1986)), which are easy to use but are limited by poor communication between software modules. For example, some menu-based systems such as KBVision, Image Analyst, and RIPS, organize data and actions at varying levels of abstraction to facilitate the communication between commands. This desire to isolate the user from the underlying data and data-manipulation actions require extensive software developement that may result in increased computational complexity.

Flickner et al. (1990) lists systems built around subroutine libraries, as another category of developement systems for machine vision and image processing. Such systems while not having the same inter-module communication problems, do require extensive programming. Furthermore, extensive subroutine libraries can lead to programming errors due perhaps incorrect understanding of implementation details.

The final category listed in (Flickner (1990)) includes systems built around specialized programming languages designed for the manipulation of images. An example is the Prolog-based system described by Chan et al. (1990a) that has been used in the inspection of food products (Chan (1990b)). In this system a computer vision problem is decomposed into a Prolog based description. These systems tend to suffer from portability as the languages used are not generally available. They have to be implemented before these systems can be used. This restricts the choice of computers in which they may be implemented.

All of these systems also suffer from the need to balance portability and ease of use against efficiency (Flickner (1990)). To alleviate this problem, Flickner et al. (1990) proposes the use of OLIVE, an object-oriented language "schema" composed of a set of objects and operations to be added to existing languages.

This is the basic approach taken in this dissertation, in the design of the software environment.

Most of these general purpose systems are aimed at providing a set of general purpose image analysis operators. They do not address the special needs of specific classes of problems. The problem is that, given such a set of general purpose "methods" and "tools", a solution based on them is often forced and thus may not be optimal. This "cart before the horse" philosophy as Lawson (1990) calls it, often hinders the proper development of computer-based systems. From the software standpoint, when such a design methodology is used, Lawson states that

> "...from the beginning, a number of degrees of freedom are removed, restricting the solution domain. In this environment, the problem solution is often forced into the framework of computer-related artifacts that conflict with the natural problem philosophy. Eventually, this results in a variety of long-lasting and costly practical problems ..."

The approach recommended by Lawson (1990) is to first establish a problem-relevant philosophy, and then to define the "computer-related artifacts" as he calls it as a natural consequence of, and consistent with, this philosophy. The word "philosophy" as used by Lawson (1990) is taken to mean:

> "... a unifying common view of how a problem or class of problems shall 'in principle' be treated."

Note that this design philosophy is necessary for promoting the reusability of "computer-related artifacts."

Furthermore, the systems that have been described basically provide an environment that concentrates on the computer vision aspects of a machine vision system design. Minimal consideration is given to hardware prototyping. This dissertation takes an integrated hardware and software view, a characteristic that is considered to be important for machine vision systems designed for use in industry (Lavin (1988)). At the time of this writing, the author was not aware of any system that unifies total system design for machine vision systems in this manner.

The concept of a total design framework wherein a system design must consider both the software and hardware aspects as a single concern rather than as separate issues is further elaborated by Roman et al. (1984):

> "System inadequacies may be the fault of designers who didn't look at the whole picture. To be successful, a design must merge hardware and software concerns into a single, unified perspective."

The authors (Roman (1984)) present the total design framework in stages beginning with the problem definition stage. This is followed by the system design stage where both hardware and software requirements are considered. Only after this are the hardware and software components designed separately.

In general these general purpose image processing systems typically

(1) accept data from only one imaging modality - usually area array cameras,

(2) do not provide a mechanism for control of a materials handling system for transporting planar webs,

(3) do not support experimentation with hardware.

3.7 Summary

An overview of the machine vision prototyper has been provided in this chapter. This overview included a discussion of how the prototyper can be utilized to rapidly prototype a machine vision system. The hardware components of the machine vision prototyper was next described. To support rapid-prototyping activity, a platform for mounting hardware components, a materials transport system, and an object detection system is provided. To facilitate rapid-prototyping a database of components is made available. This limited database currently consists of components suitable for the type of inspection that analyzes planar webs for surface defects that manifest themselves as variations in color, tone, and/or texture from that of a normal web surface.

Following the description of the hardware components, the requirements of the software components of the machine vision prototyper was described. To support rapid-prototyping activity, an object-oriented approach to system design is taken. To facilitate rapid-prototyping a database of of software modules is made available. Finally, a comparison with other vision system development systems was given. It was concluded that these systems were of limited utility because they did not provide a framework for combined hardware and software design.

Chapter 4

COMPONENTS OF THE MACHINE VISION PROTOTYPER

It has been shown that machine vision system design is a complex and major undertaking requiring much experimentation. This experimentation is needed to allow consideration of different solutions to a particular inspection problem. A machine vision prototyper was thus proposed as an experimental vehicle for designing machine vision systems. In Chapter 3, the machine vision prototyper was defined to be a set of hardware and software components that can be combined in a variety of ways to allow the consideration of different solutions to a particular inspection problem. Chapter 3 also provided an overview of the machine vision prototyper. The object of this chapter is thus to provide a detailed description of the major hardware and software components of the machine vision prototyper.

4.1 Hardware Components of the Machine Vision Prototyper

To support machine vision rapid-prototyping, Chapter 3 listed the following as necessary hardware components of the prototyper:

1. A basic platform for mounting hardware components, for transporting planar webs, and for detecting the presence of a web.

2. A database of reusable modules.

3. A color monitor for displaying images, a laser printer for obtaining hardcopies of images, and a plotter for drawing lines such as outlines of webs and outlines of defects in these webs. The laser printer is also able to draw these lines.

4. A computer.

For hardware rapid-prototyping, the prototyper provides a basic platform for experimentation consisting of 1) a mounting system, 2) a materials transport system, and 3) an object detection system. These three items provide the ability to experiment with different sensors and illumination sources, experiment with different imaging geometries, and to move an object to within viewing range of a sensor.

4.1.1 The Mounting System

The most important aspects of hardware rapid-prototyping is in the ability to easily change imaging geometry and in the ability to experiment with different imaging components. The need to easily change imaging geometry and experiment with different imaging components make it very desirable to minimize the number of special purpose machined parts required for mounting imaging components. The basic objective of a mounting system must therefore be to minimize the number of special purpose machine parts. The mounting system must also be able to point sensors with some degree of accuracy. It must also have the ability to position and point the illumination sources to provide the best possible illumination over an object to be imaged.

The above requirements can be met by using an optical table with a variety of parts available including positioning and pointing hardware. The optical table that has been provided for the prototyper allows components to be mounted at different points on its surface spaced an inch apart and thus provides two degrees of freedom for positioning components. Accessories that are available with the optical table include micropositioning devices such as translators and rotators, a variety of rod slides, optical rod mounts, rod holders, carriers and optical table rails. Micropositioning devices permit sensors to be mounted and accurately pointed. Rod slides permit micropositioning devices and illumination sources to be mounted on optical rod mounts. These rod slides enable components to be mounted at any point vertically along the lengths of optical rod mounts to provide a third degree of freedom for positioning components. Rod holders are provided for mounting optical rod mounts on the optical table. Carriers are for mounting rod mounts onto optical table rails. Optical table rails permit mounted sensors to be precisely positioned at any desired distance from an object. Finally, optical mounts are also available for holding lens filters.

All of these optical fixtures provide the flexibility to quickly change imaging geometry and to experiment with different imaging hardware. These fixtures will thus ensure the reusability of imaging components. All that remains are the special connecting mounts for bolting components onto these optical fixtures. The need for milling special purpose parts are thus minimized.

It would also be very useful if all the components used the same basic types of bolts for mounting onto the optical table. One would then only need to keep a database of a number of lengths of this basic bolt instead of having to keep in store bolts of every possible size.

A cloth enclosure is used to house imaging systems mounted on the optical bench in order to provide a measure of control over the illumination. An exhaust system is also provided to lower the amount of heat generated by the illumination, down to a minimum.

In the hardware prototyping procedure, changes in imaging geometry can be made and tested by viewing images scanned on a sensor. Changes in imaging geometry are made till images of sufficient quality are obtained. Furthermore, the optical table allows the mounting of different types of sensors such as color cameras, X-ray scanners, and any other type of sensor. It will also permit a variety of imaging methods such as front-lighting, back-lighting, and structured-lighting, to be used.

4.1.2 The Materials Handling System

To capture an image of a web, the web must be transported within view of the imaging sensor. Therefore, a materials handling system that can accurately control the velocity and position of a web is needed. For this purpose, a 6 foot long Velmex linear stage using stepper motors is used. This linear stage cannot handle very long webs nor will it move webs through at high industrial speeds. It does, however, let us experiment with the imaging sytem.

The reason for the choice of a stepping motor is that simple and precise control of velocity and position through a computer can be achieved, an advantage over the DC servomotors as discussed in Chapter 2. The speed of the motor-driven slide can be specified by the number of steps per second. For the slide assembly currently being used, each step moves the slide an increment of 0.0005 inch. Therefore, the velocity (V_s) of the slide in inches per second is given by:

$$V_s = 0.0005V_n$$

where V_n is the actual velocity of the slide in steps per second.

Precise positioning can be achieved by the ability to specify, independently of the speed of the slide, the exact number of steps to move. The ability to precisely position an object permits a computer to automatically bring the object into the proper field of view of a sensor.

To demonstrate the utility of precise control over the velocity of the slide, let us consider the scanning of images by a line scan camera. For a planar web, the web to be imaged is scanned line by line as it passes through the line of view of the camera. Therefore, one dimension of the resolution of the scanned image is determined by the time between each scan performed by the camera and the velocity of the object mounted on the transport system. We shall from here on refer to this dimension as the **downweb resolution.** The other dimension is dependent on the resolution of the line scan camera and is independent of downweb resolution.

Downweb resolution can be specified in terms of the number of scan lines per inch. Let R_d represent the downweb resolution and t_s the time between each scan performed by the camera. Therefore,

$$R_d = \frac{1}{V_s t_s} = \frac{1}{0.0005V_n t_s} = \frac{2000}{n t_s}$$

We can now determine an arbitrary downweb resolution as can be seen in the following example.

Suppose we wish to have a downweb resolution of 32 points per inch. We are further given that the time between each scan performed by the camera is 0.400641 second. Therefore we have to drive the slide with a velocity in steps per second, V_n, given by,

$$V_n = \frac{2000}{R_d t_s} = \frac{2000}{32.0 \times 0.400641} = 156$$

The stepper motor that drives the materials carrying slide is controlled by a Velmex 8312 Controller/Driver. Communication with the Velmex 8312 is handled through a full-duplex RS-232C port. The host computer sends commands through one of its own serial ports to control a wide range of functions which include starting, acceleration, stopping, velocity and distance to move.

4.1.3 The Object Detection System

The lengths of planar webs vary a great deal and can be up to several feet in length. Furthermore, these webs are typically imaged at high spatial resolutions. As a result, a large amount of of data may sometimes be expected from imaging a web. To keep this amount of data at a minimum, only image data that is absolutely necessary should be scanned. This is facilitated by the use of an object detection system that can detect the beginning and end of a web. The system provided for object detection on the prototper is the Optoscan system manufactured by Scientific Technologies, Inc.. This system is made up of an array of infrared lasers on $\frac{3}{8}$ inch centers along a 24 inch length and an array of detectors also on $\frac{3}{8}$ inch centers along a 24 inch length. A web breaking the path being monitored by the Optoscan system can be instantly detected.

4.1.4 The Database of Reusable Components

Hardware rapid-prototyping requires the ability to experiment with different sensors and illumination sources. To facilitate hardware rapid-prototyping, a database of these hardware components must be available. For the inspection of planar webs, these components include the imaging components for the three kinds of inspection problems as stated in Chapter 1. The machine vision prototyper currently has a limited set of such components. The number of these components are limited due mainly to "financial constraints." However, the number of these components is expected to increase with the application of the prototyper to new and different inspection problems.

The currently available imaging components is meant for use in the first type of inspection problem as stated in Chapter 1. This type of inspection involves the analysis of the surfaces of the planar webs to look for surface defects that manifest themselves as variations in color, tone, and/or texture from that of the normal web surface. The currently available imaging components include two color line scan cameras, fiber-optic light sources with tungsten halogen bulbs, the appropriate filters, and the camera interfaces to the computer. The imaging components for this type of inspection is further described in Chapter 6 as they are part of the machine vision system for the inspection hardwood lumber described in chapter 6.

Work is currently underway to develop a structured-light based imaging system. The imaging components and computer interfaces needed for such a system are described by Hou (1993).

4.1.5 Display and Plotting

To support rapid-prototyping, the prototyper must provide the facilities to analyze and compare changes to imaging geometry, sensors, and illumination components, and to view the results of processing during software development. The facilities provided by the prototyper consist of image display capability, the ability to make hardcopies of images, and plotting capability.

Three display devices are currently available for displaying images. These display devices are utilized on the three different computer system on which the machine vision prototyper is implemented. On the VAX 11/785 system, a Perceptics display is available with ability to display images of sizes up to 512 rows and 512 columns. On the IBM PS/2 Model 80, a VGA display is available. The VGA monitor however, is limited in its ability to display color images in all the possible colors. The VGA monitor can only display a maximum of 256 colors at a time. Furthermore, the resolution of the monitor in the 256 color display mode is only 200 rows by 320 columns. The prototyper is also implemented on an IBM RS6000 model 520 workstation. The workstation has a 24 bit color graphics processor and is capable of displaying images of sizes up to 1024 rows by 1280 columns. For making hardcopies of images, an IBM laser printer and an Apple laser printer are available.

In addition to image display capability, line drawing capability is provided by a HP Draftmaster II plotter and the laser printers. This capability is needed to plot the outlines of defects, among other things. The HP plotter provides the additional capability to make line drawings to scale, i.e., the actual sizes of objects can be drawn to scale. Such a capability is necessary to demonstrate the accuracy of the machine vision system in locating a defect.

To obtain line drawings of objects drawn to scale we need to calculate the vertical and horizontal parameters. Vertical coordinates of points on the HP plotter can be calculated given R_h = the number of points per inch resolution of the plotter, and R_d = the downweb resolution of the images. Let i be the row number of a point in an image. Let y be the row number of the corresponding point on the hp plotter. Therefore y and i are related by:

$$y = R_h(\frac{i}{R_d})$$

Horizontal coordinates are calculated in a similar fashion.

4.1.6 The Computer

The requirements for a computer for the machine vision prototyper include an appropriate bus with sufficient expansion slots to provide for sensor interfaces, sufficient main memory to allow image data and computer vision software to be completely memory resident, secondary storage capability for storing images to be scanned, and a variety of parallel and serial ports for interfacing with peripheral devices such as the object detector the linear stage and a printer.

The original implementation of the software components of the prototyper was on a VAX 11/785 computer. This was the only computer available when the research first began. The first complete implementation of the prototyper was provided on an IBM PS/2 Model 80 computer with a Microchannel bus. The current configuration of the computer has 10 megabytes of main memory, a 1.44 megabyte diskette drive, 300 megabytes of hard disk, 5 serial ports, 2 parallel ports, and VGA color display. The IBM PS/2 Model 80 was chosen due the requirement by the funding agency that the computer be an IBM. The prototyper system has since been implemented on an IBM RISC 6000 model 520 workstation.

4.2 Software Components of the Machine Vision Prototyper

The rapid-prototyping approach to machine vision system development entails the ability to design partial implementations with each implementation evolving (after some experimentation) to the next higher level implementation that is closer to meeting design requirements. The key to rapid-prototyping is in the ability to reuse components of a previous partial implementation for the construction of the next implementation, and the ability to experiment with each implementation. Therefore, to support the rapid-prototyping approach to machine vision design, the software components of the machine vision prototyper must provide 1) an environment for designing components of a partial implementation, and 2) experimental tools for testing partial implementations. This environment must also allow each implementation to evolve from a previous implementation to the next with a minimum of effort. This effort can be minimized by improving the reusability of existing components.

Chapter 3 lists

1. an object-oriented design environment,

2. a database of reusable software components,

3. a kernel of support for low-level system functions and for facilitating object-oriented system design,

as software components of the environment provided by the machine vision prototyper for supporting rapid-prototyping activity. The object-oriented design environment provides an environment for designing components of machine vision systems. Experimental tools for testing prototype designs are also provided within the environment. To facilitate the evolution of each partial implementation to the next and to keep the number of these implementations to a minimum, an extensive database of reusable software components is needed. It is quite clear that the more reusable components there are that can be used in any design, the faster we can approach a final prototype.

The objective of this section is thus to provide a description of the object-oriented design environment for designing machine vision system components, and a description of the experimental tools provided for testing prototype implementations. Also provided is a description of the low-level utility functions that provide low-level system services and object-oriented system design support. The database of reusable software components is described in detail in Chapter 6 since they were developed while demonstrating the utility of the machine vision prototyper in designing the machine vision system for the inspection of hardwood lumber.

4.2.1 The Object-Oriented Design Environment

Rapid-prototyping is facilitated not only by the availability of components that can be used in implementing prototypes, but also by the ease with which these components can be interfaced with each other. An object-oriented design environment was chosen because object-oriented systems are inherently modular with each module or object being relatively independent of the next. Such an environment is well suited to rapid-prototyping due to the fact that the evolution of one implementation to the next is made easier as changes within each independent module (object) can be made without affecting the rest of the system. As was discussed in Chapter 3, only the response of the system to given sets of circumstances are altered.

Objects can be designed to be relatively independent of one another due to information hiding through encapsulation. Through information hiding, a user need not know the implementation details of objects. Such details are accessible only on a "need to know" basis. As such, ease of use and ease of learning is promoted through a minimizing of information needed by a user. Hence, an object-oriented design environment was a good choice for the prototyper.

On the prototyper, each reusable software component is designed around the concept of an object. An object is an encapsulation of data and operators which manipulate the data. This applies to both hardware and software objects. Hardware components can also be considered to be objects consisting of data and operators. As far as the software is concerned operators belonging to hardware objects are actually manipulating "data." In this dissertation, operators belonging to objects will be called "object-manipulators" to avoid multiple use of the word "operator" (such as in "image analysis operator").

To support object-oriented design, and to provide the necessary low-level system software support, a set of utility functions is provided. The structure of the software environment provided by the prototyper can thus be seen to be made up of

1. objects consisting of data and their associated object-manipulators, and
2. low-level system software.

This is a simple two-level system stru.ture that is easy to learn and use.

Object-manipulators include

(1) hardware object-manipulators that perform functions such as control of the materials handling system, control of the object detection system, control of sensors and sensor interfaces, and control of the image display and hardcopy devices, and

(2) software object-manipulators consisting of image and scene analysis software, image processing software, and general purpose utilities.

These object-manipulators can be further divided into components that can be used in a machine vision system, and components provided as experimental tools for testing prototype designs. The experimental tools will be described in the next section.

The low-level system software consists of an

1. object management system,
2. a parameter management system,
3. a specification management system,
4. a set of system functions known as the system primitives.

The low-level system software is described in subsection 4.2.3.

For portability across different computers, the software was designed to work within the UNIX environment. No other operating system is available on the wide range of computers from mainframe to personal computers, as UNIX is. The software is mostly portable across the different versions of UNIX with the exception of certain unavoidable machine dependent softwa.e. For example, software calls that communicate with devices through a serial interface on the IBM PS/2 Model 80 may not be available on a VAX 11/785 computer.

4.2.2 The Experimental Tools

Experimentation with a prototype design consists of experimenting with imaging geometry, analyzing and displaying results of applying different algorithms and experimenting with different imaging geometry, and experimenting with different software parameters.

The object of experimenting with imaging geometry is to obtain the best quality images from a given set of imaging components consisting of a sensor and illumination sources. For the linescan camera imaging system, the objects requiring software control consists of the OPTOSCAN for detecting the presence of an incoming object, a linear stage for transporting a web to be imaged and the camera for capturing an image of the web. To test the quality of images for a given configuration of the imaging system, the ability to position an object within view of the camera and the ability to acquire images are needed. To address these needs, the object-manipulators **position** and **gbw** are provided. The object-manipulator

position provides the ability to move objects mounted on the linear stage to any arbitrary position. The object-manipulator **gbw** allows one to scan a single line of view of an object over and over again for a preprogrammed number of times. This object-manipulator also allows us to scan the black and white surfaces necessary for shading correction, an image processing step that was found to be necessary during the prototyping of the machine vision system described in Chapter 6.

To display the results of scanning a web and the processing of web images, image display object-manipulators are needed. There are two different image display object-manipulators that have been provided for this purpose. These object-manipulators will display images stored in files on disk. Images can be raw, unprocessed images of an object or images resulting from a processing stage in a machine vision system. To display these images, **idsp** has been provided. Depending on the type of display device, idsp will display a full color image, or a black and white image. The VGA display device on the IBM PS/2 Model 80 for example is capable of only displaying 256 different colors at a time. Therefore, **idsp** will display only black and white images on this device. A second image display object-manipulator, **midsp**, accepts an Ascii file containing a set of red, green and blue values for each of the 256 possible gray level values for each pixel in an image. This object-manipulator was provided to permit the display of a range of gray level values in an image in any color. This is useful for thresholding an image, for example. This object-manipulator can also be used to display a color image on a VGA display device. This is done with the help of the object-manipulator **gidsp**. **Gidsp** takes a portion of an image or a whole image, and produces a single band image whose gray level values now represent an index into a set of palette values. This object-manipulator will perform a mapping from the $256 \times 256 \times 256$ possible different colors within an image down to only 256 colors.

An important aspect of experimenting with a machine vision system is the ability to change certain parameters without having to change and recompile the software. For this purpose, a specification management system is provided. The specification management system allows a user to enter specifications into a database. These specifications could be the definition of a particular defect, for example. The database of specifications is considered to be an object, and the interface between the database and a user is provided by the object-manipulator **specs**. This is actually a small database management system that manipulates a database object file. Allowable operations are delete, replace, insert and append. All records in the database are classified under different object names which we shall refer to as categories. Each category contains a set of records containing specifications that define a set of variables associated with an object. The specification management system is further described in the next section.

In any extensive software environment, there is usually a help system that allows a user to learn about the software. On the system described in this work there is a help system which is entered through the command **help**. Through the help command one can obtain information about the different object-manipulators and utility functions. Another useful utility is the **history** command. This command acts as a simple "shell" that remembers commands that have been entered up to 999 commands. This is similar to the "history" system in the C-shell of a UNIX system allowing previous command to be re-executed without having to type in the entire command. The main difference is that the **history** command allows the execution of a block of commands. This can be useful when a series of commands needs to be executed over and over again. This can happen when scanning and analyzing several planar webs in an effort to test the accuracy of a machine vision implementation.

Finally, there is a set of miscellaneous object-manipulators that perform such functions as extracting single and two dimensional histograms, displaying actual values of a small portion of images, allocation of images, image header manipulation, and more. These functins are usually found in typical image processing workstations such as ELAS and provide some of the many utilities useful for managing and analyzing images.

4.2.3 The Utility Functions

In any large software system, there is usually a set of low-level functions that form the foundation of the system. These low-level functions typically provide an interface to the resident operating system and other types of support depending on the needs of the software. For the machine vision prototyper, there is a need for functions that perform a variety of object management tasks and certain "primitive" tasks. The management tasks include the management of objects, the management of simple object parameter variables, the communication of these parameters between object, and the management of specifications of object variables.

The primitive tasks include low-level functions to control the HP plotter, the Velmex linear stage, and the OPTOSCAN, image file I/O operations, error handling, and message passing. The utility functions can be divided into four groups which will be referred to as the object management system, the parameter management system, the specification management system, and the system primitives. Each group will be briefly described in the following subsections. Detailed description requires an in-depth understanding of the "C" language and is thus not provided in this dissertation. A complete list of the source code has been made available in the Spatial Data Analysis Laboratory at Virginia Polytechnic Institue and State University.

4.2.3.1 The Object Management System

To facilitate object-oriented design, a mechanism must be provided to support the encapsulation of data and operators, and to provide for inheritance between object classes.. The object management system provides the necessary association between objects and their object-manipulators to support encapsulation. This association is enabled through the assignment of "usage" names to object data. These usage names refer to object data stored in files on disk, data structures in main memory, or to actual hardware objects. On a UNIX system, hardware objects are also accessed through a file name and can be treated as files. These usage names can be thought of as the **class** names of objects. When an object-manipulator wishes to access data in a file for example, it does not directly access the file through the file name. Instead, it accesses the file by a fixed usage name, in a request to the object management system. The object management system then looks in an object-definition file for a file name associated with the usage name. File names can be also accessed through a function supplied by the object management system. Accessing these file names are not necessary, however, as they can be referred to by their usage names. The object management system can allocate, open, close, delete, read from and write to files.

Object-manipulators that are written will permanently refer to the chosen usage names associated with an object. This enforces the encapsulation of object data and object-manipulators. Object-manipulators associated with a particular object will refer to the same usage name. However, object-manipulators may refer to more than one usage name. This is how inheritance is supported by the machine vision prototyper. Such an object-manipulator will attempt to access the object through the first usage name. Should this be unsuccessful, it will then attempt to access the second usage name that would normally be the subclass of the class given by the first usage name.

4.2.3.2 The Specification Management System

There is sometimes a need to allow a user to manipulate a set of specifications associated with a given object. The specification management system is provided for this purpose. The specification system stores a set of object specifications in a database. Specification data can range from the simple integer parameters to complex symbol definitions. For example, one could associate a character string describing a defect property such as "diameter" with the range of values "0.1 inches or greater." Or we could associate symbols with command strings that are used to control hardware objects such as the linear stage of our machine vision system prototyper. In other words, there is no limit to how specifications can be defined in the system described in this work.

Specifications are entered by a user through the specification object manipulator described in subsection 4.2.2. To access these specifications in software, only two functions are needed in the specification management system. The first of these is *Spec_init()* which reads in specification data into a linked-list. The second function is *Find_Entry()* which looks for a particular specification record in the linked-list. Specification records are retrieved keyed on a category and the first few characters of the record.

4.2.3.3 The Parameter Management System

Most software usually require some form of parameter entry. These parameters could be a particular band of a multi-channel image to be analyzed. Or they could be coordinates corresponding to a sub-portion of an image to be further analyzed. Often this is done by prompting the user interactively. This method can get quite tedious, however, especially if the parameters are the same for multiple executions of the same software. Defaults for these parameters can be given but they remain a permanent feature of the software. A better alternative is defaults to be redefinable so that they can be used over and over again till a new set of parameters are required.

The difference between the parameter management and specification management systems is that the parameter management system allows object variables to be changed within the object-manipulators, while the specification management system allows object variables to be changed in a database file containing object variable definitions. Object specifications represent variables that are seldom changed while object parameters represent variables that are most often changed.

The parameter management system in this work allows a user to set defaults which can be changed at any time. The system consists of a set of functions that allow a user to set, reset and retrieve parameters at will. The parameter management system keeps a set of parameters in internal buffers in main memory and also a set in a file on disk called the parameter subfile. By storing these parameters in the parameter subfile, an object-manipulator can access the same parameters everytime it is executed. Furthermore, different object-manipulators may also access the same subfile if they use the same parameters.

For the parameter management system, only integer and floating point parameters are allowed. For more complex requirements such as when symbolic constants are involved, or when a very large set of specifications need to be defined, the specification management system is used.

4.2.3.4 The System Primitives

The system primitives consists of a set of functions for plotting on a HP plotter, controlling the Velmex linear stage, communicating with the OPTOSCAN, image file I/O operations, terminal I/O, error management, and message passing. These functions provide most of the low-level utility functions that are needed by a programmer to write image processing or computer vision software, relieving the programmer from having to know implementation details such as the format of an image file.

For controlling the linear stage, OPTOSCAN, and HP plotter, functions are provided that open, close and communicate with these devices. These functions control the devices by sending command strings to these devices.

In order to access image object data files, functions have been provided to open and allocate, to read from, and to write to these files. Other image object data file access functions include those for reading and writing file headers.

An important component of most software systems is the user interface where the user is prompted for parameters. The function *rio()* has been provided for this reason. This function is not meant to replace the standard terminal I/O functions provided in C or Fortran or any other language that may be used but to to supplement them. The function *rio()*, provides a means of counting the number of integer or floating point parameters that have been entered at the terminal. Furthermore it allows the user to enter a string of commands that are being prompted for at different invocations of *rio()*.

Rounding out the system primitives are the error management and message passing utility functions. The error management functions provide a means of improved error reporting. The message passing utility functions allow an object-manipulator to execute another object-manipulator. Message passing is the mechanism by which objects may communicate with each other. Note that this system provides the ability for a series of object-manipulators to be invoked one after another without user interaction.

4.3 Summary

The components of the machine vision system prototyper have been described in this chapter. The hardware components described were the mounting system, materials handling system, object detection system, display and plotting devices, database of sensors, lenses, filters and illumination sources, and computer. The software components described basically consists of a set of functions that perform a variety of management and other low-level tasks including the support of object-oriented mechanisms, a set of object-manipulators for image acquisition and display, and other miscellaneous object-manipulators that enhance the software environment.

Chapter 5

THE INSPECTION OF HARDWOOD LUMBER

Having developed the machine vision prototyper, the next objective is to demonstrate its utility in designing a machine vision system. The problem that has been chosen is the inspection of hardwood lumber. The strategy to be followed in utilizing the machine vision prototyper for rapid-prototyping a machine vision system is depicted in Figure 3.1. As can be seen in Figure 3.1, the first step in rapid-prototyping a machine vision system is to define the problem. To accomplish this, a study of the inspection problem is necessary to determine the requirments. For this reason, there are two objectives to be accomplished in this chapter. The first objective is to provide a background into the hardwood lumber inspection problem to determine the characteristics of the problem. The second objective is to examine some of the previous work that has been done in lumber inspection in an effort to determine the shortcomings of these studies. In doing so we can begin to do what is necessary to overcome these shortcomings.

Motivation for applying the design of a planar web inspection system to the inspection of hardwood lumber can be found in the fact that the manufacture of secondary wood products can readily benefit from automation. The industry has up till now been relatively slow in adapting to changes in the economics of manufacturing secondary wood products. According to Conners et al. (1985), the remanufacture of hardwood lumber into specific size parts for furniture construction has not changed appreciably in the last 50 years. The remanufacture of hardwood lumber into specific size rough parts takes place in the rough mill. Therefore, a goal of this research is to show the feasibility of creating a machine vision system which will be an essential part of the automation of the operation of a rough mill. To understand the requirements of such a machine vision system, a brief discussion of the related aspects of a roughmill is necessary.

5.1 The Rough Mill

All the lumber entering a secondary re-manufacturing plant is cut up into rough parts used throughout the rest of the manufacturing process in a place called the rough mill. The primary concern of the rough mill is that of getting as many long, wide and clear rough parts as possible. Higher grades of lumber are used to achieve this goal. The percentage of a board suface being defect free determines its grade. The higher the grade the lesser the number of defects. Therefore one needs higher grades of lumber to obtain long, wide and clear parts. However, costs can double in going from one grade of lumber to the next. The objective must then be to obtain the longest, widest and clearest parts from the lumber being processed.

In the roughmill an operator will visually examine both sides of a board for defects and attempt to cut the board into parts to eliminate unacceptable defects while maximizing the yield of usable parts. To accomplish this, the operator must (Conners (1985)):

(1) be able to see and recognize the defects;

(2) have the mental aptitude to properly locate the cuts;

(3) possess the physical strength to position the board manually;

(4) resist boredom and maintain an alert mental attitude.

Rough mills however are typically dusty, noisy and often have a hazardous work environment which can distract the operator. These factors tends to place stress on the operator and affects the four necessary attributes of the operator listed above. As such the quality of the operator's performance tends to deterioriate during the course of the day, which will affect the sawing decisions he has to make. The basic sawing decisions that are being made in the rough mill are important as raw material costs make up as much as 50 percent of the total manufacturing cost of a hardwood piece of furniture. Therefore, deterioration of an operator's performance can result in increased manufacturing cost.

It has, however, proven very difficult for management to assess the performance of rough mill employees. In fact there is little in the way of quantitative data to assess the quality of this performance (Conners (1985)). While most will agree that this performance is suboptimal, it is unclear how well they perform. A computer based system that automates an operator's functions, on the other hand will perform consistently. While there is no guarantee that such a system could always outperform a human operator, it's consistency will facilitate assessment of it's performance. Furthermore, such a system could be constantly subjected to improvements and the results of these improvements can be directly measured. Automation of the operator's functions in the rough mill would thus be desirable. To automate an operator's functions, one must first understand what is expected of the operator.

In addition to locating and identifying defects, an operator must develop a cutting strategy based on the day's cutting bill for cutting the board into the needed parts that are free of undesirable defects, and saw up the board based on this cutting strategy. A cutting bill is a list of part dimensions and the number of parts of each dimension that are needed. The day's cutting bill depends on what is currently being manufactured in the plant. The grade of lumber to be processed is determined by the number of long parts specified in the cutting bill. The longer the parts required, the higher is the grade of lumber needed. As the manufacturing requirements change, the cutting bill usually changes too. This occurs fairly frequently. Therefore, not only does the worker have to adapt his sawing strategy to frequent changes in cutting bills, the worker has to aim at getting as many long parts as possible.

For each cutting bill, the worker must know which surface features to remove and which to leave in the parts to be sawn. Not all features that appear on a board's surface represent undesirable defects. Undesirable features are removable defects that are not to appear in any rough parts. A feature is considered undesirable depending on the product being manufactured and the relative quality of the product within the industry. In order to make a decision on whether to remove a surface feature, the operator must first identify the type of surface feature. Once the type of surface feature has been determined, the worker must then determine the characteristics of that feature type. For example, a knot would be considered an undesirable defect if its diameter exceeds a certain size, if it contains decay, or if it is loose.

From the above discussion, it can be seen that manufacturing requirements may change from day to day. These changes in manufacturing requirements result in changes in cutting bills and the definitions of removable defects. Therefore any system for automating an operator's functions must be flexible enough to accomodate changes in inspection requirements.

Finally, most furniture and fixture plants handle multiple hardwood species. Some of these include red oak, white oak, cherry, hard maple, soft maple, walnut, pecan, ash, poplar, mahogany and hickory. Most plants have to deal with at least two to three of these species. Plant workers must then have to deal with a number of different species, species which often differ in visual appearance as can be seen in Figures 5.1, 5.2 and 5.3. Therefore the system for automating an operator's functions must be able to perform in a species independent manner.

The function of the operator that is most difficult to replace is his ability to see and recognize defects. It would be of considerable benefit to the wood products industry if a machine vision system could be designed to replace an operator's ability to locate and identify unacceptable surface defects. Therefore, one of the goals of this research is to determine the feasibility of designing a machine vision system for the inspection of hardwood lumber. From the discussion presented so far, such a machine vision system must: 1) be flexible enough to accomodate changes the definitions of removable defects since manufactuing requirements and hence definitions of removable defects may change from day to day, and 2) be able to perform its inspection in a species independent manner since most plants handle multiple hardwood species.

In order to design a machine vision system to identify surface defects, we must first understand the nature of the color variation in wood and the characteristics of the different defects. We will next discuss the factors that affect wood color.

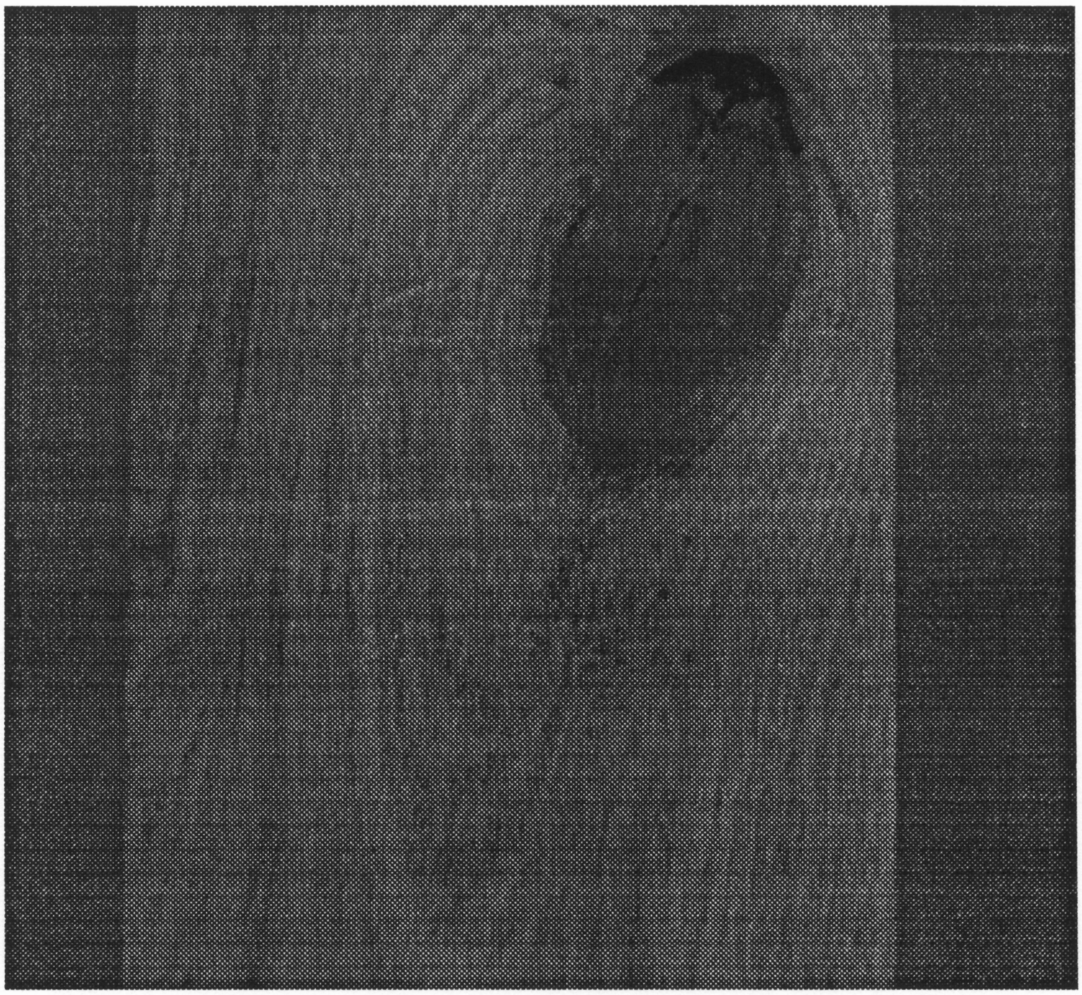

Figure 5.1. An oak board with a knot.

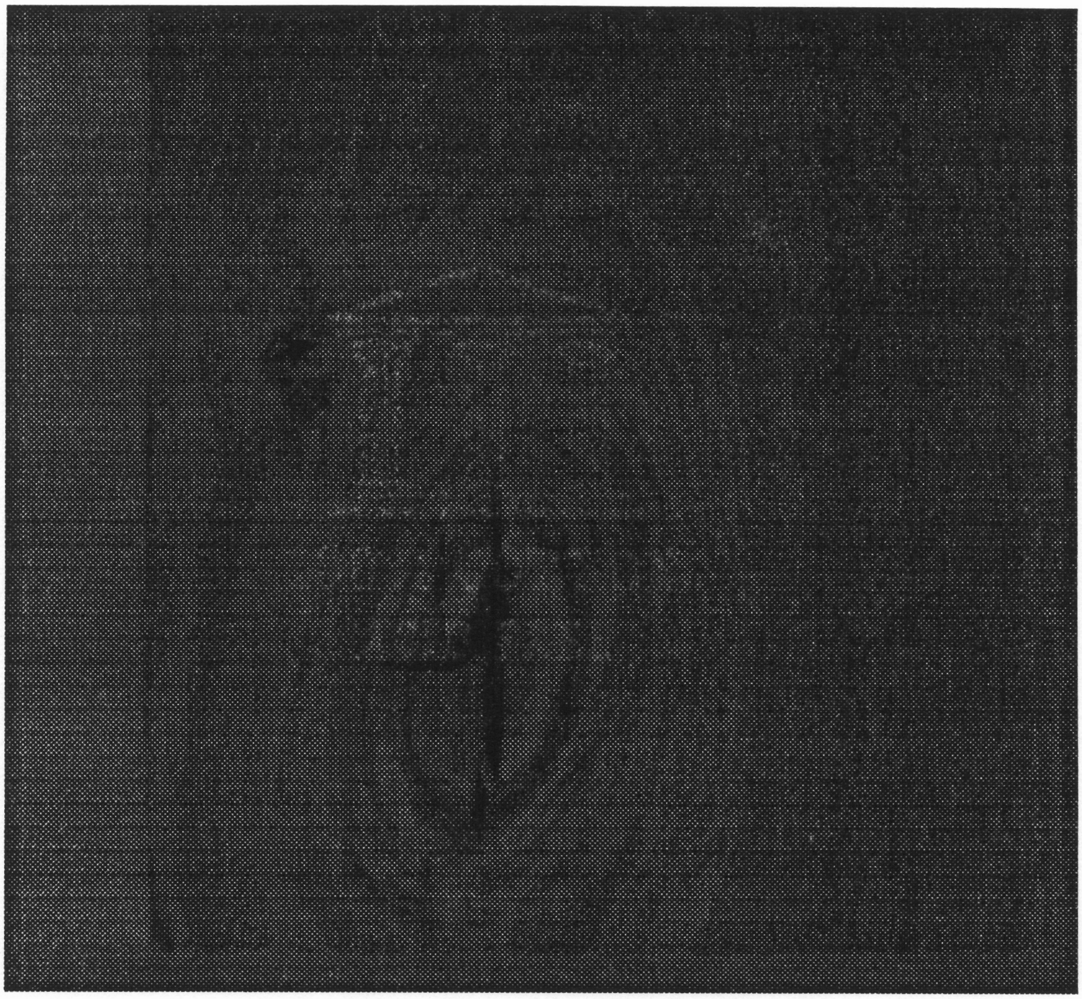

Figure 5.2. A walnut board with a knot and a hole.

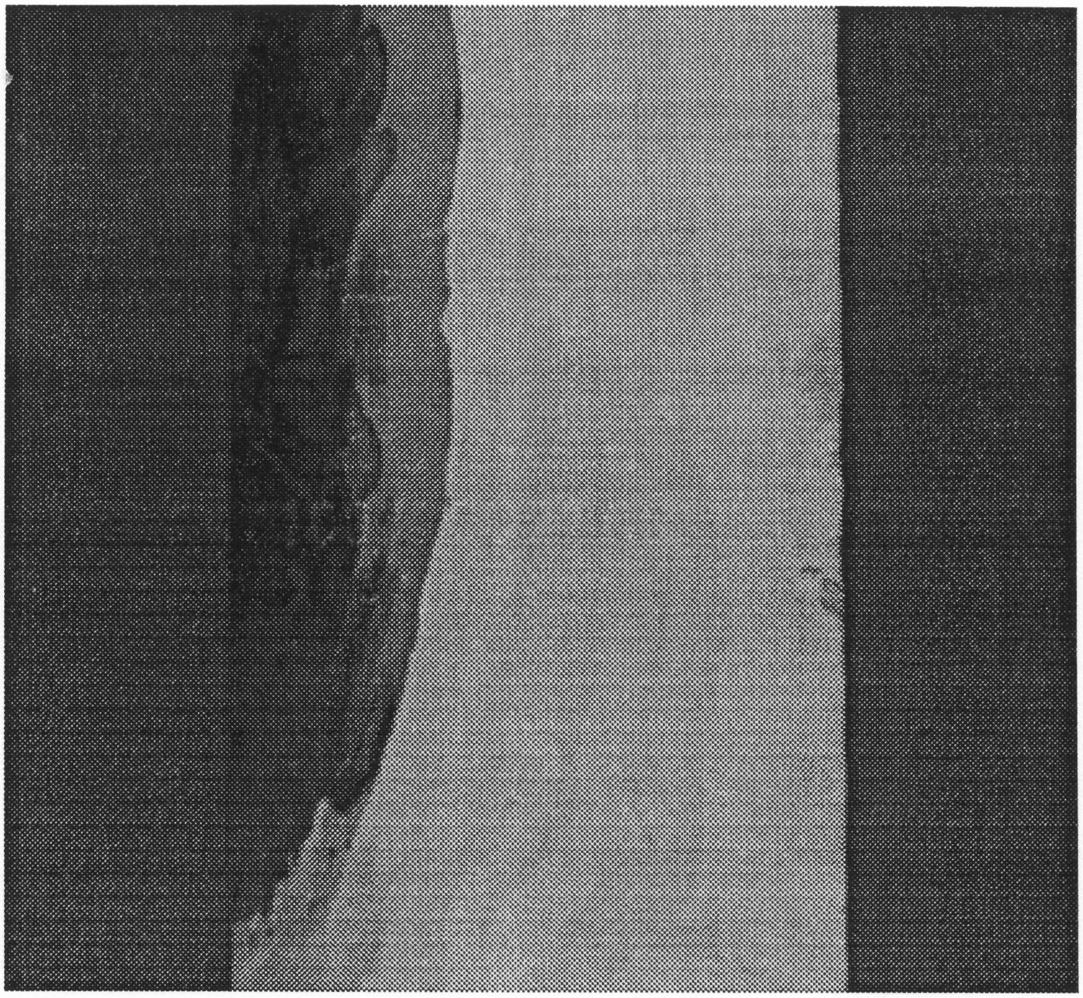

Figure 5.3. A maple board with wane.

5.2 Factors Affecting Wood Color

In visually examining boards for removable defects, the human operator relies on color information to help him identify the features. The more the color of the feature deviates from the color of clearwood, the easier it is for the operator to notice it and identify it. Hence, if humans use color information for feature location and identification, it would seem natural to explore the utility of color in designing the algorithms for the machine vision system for the location and identification of these features. In the 1950s', Lakatosh (Lakatosh) made an extensive study of the characteristics of surface defects in wood and concluded that:

> "Each type of wood and defect has its own color characteristics. Moreover, the basic indices characterizing the color of wood (luminance, coefficient of chromaticity) have values with accuracy index from 0.1 to 1 percent and variation coefficient for 0.07 to 3 percent which show that the indices are highly stable."

This means that color information can be used to separate clear wood from various features and also to uniquely identify a feature. If this claim is valid, then data collected from different defects and clearwood should cluster differently in the color space thus allowing the possibility of identifying the different features and clearwood relative to their position in the color space. In order to use this information we need to first understand the factors that affect the variability in wood color. These factors can be divided into factors which can be controlled and factors which cannot be controlled.

Conners et al. (1985) lists moisture content, drying temperature, surface roughness, exposure to ultraviolet light and exposure to air as factors affecting wood color that can be controlled. Lumber entering the rough mill is kiln dried to reduce moisture content to about seven percent. This factor is determined by the drying process before rough mill processing. All the other factors mentioned affect only the surface color of the wood. Therefore, color variations caused by these other factors can be eliminated by planing the rough lumber prior to any inspection by a machine vision system.

There are a number of factors affecting wood color which cannot be controlled. Conners et al. (1985), lists the following uncountrollable parameters:

1. Wood color can vary markedly from species to species.

2. Within a species there can be a marked difference in the color of the heartwood and sapwood.

3. Within a species there is some difference between the color of boards cut either tangentially or radially.

4. Within a species the color of wood can vary with the geographical location in which the tree is cut.

5. Within a species the color properties of clearwood vary with the heigh of the tree.

These factors must be taken into account when designing the machine vision system. They show that while Lakatosh's results indicate that defects may be differented from clearwood based on color information, a supervised classification procedure will not be applicable on a global basis due to the variability in color from species to species and even within a single species of hardwood lumber. These factors, in fact, suggest that an adaptive or unsupervised classification scheme is required for identifying clearwood from potential defects.

Having shown a potential scheme that is suitable for locating and identifying surface defects, what is left is to determine the types of features that are to be identified by the machine vision system.

5.3 Surface Defects in Hardwood Lumber

Surface features in hardwood lumber can be attributed to three different causes Lamb (1990). These are the growth related features, biological features and mechanical/processing features. Growth related features are caused by the manner in which the the tree grows. The following is a list of growth related features: 1) Knots, 2) Holes, 3) Pith, 4) Bark Pocket, 5) Wane, 6) Heart/Sap wood, 7) Stain, 8) Grain Deviations, 9) Shake, and 10) Warp. Biological features are features caused by organizms such as insects and fungi. The following is a list of biological features and their characteristics: 1) Insect damage (eg. holes), 2) Rot (decay), and 3) Stain. Mechanical/Processing features are caused by the stresses and strains on the wood caused by the handling and other forms of physical processing of the wood. These features are listed as follows: 1) Warp, 2) Splits, 3) Checks, 4) Thick and Thin, and 5) Holes. The surface features that occur most frequently include knots, wane, holes, decay, stain, checks, and splits. Table 5.1 gives an indication of the relative probability of occurrence of some of these feature types in red oak for the grades of lumber typically used by the furniture and fixture industry.

Table 5.1

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

FEATURE	No. 1 Common	No. 2-Common	No. 3-Common
	% Area	% Area	% 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	8.0	13.0

For the purposes of establishing feasibility, only a subset of the most frequently occurring defects were considered. From Table 5.1, knots, splits/checks and wane were seen to be the most frequent in terms of percentage of total surface area in southern red oak. Hence knots, splits/checks and wane were chosen for this research. Holes were also chosen as they also occur very frequently in the samples provided for this research.

The characterization of these defects are given below:

Knots are characterized by whether they are sound or unsound, by diameter, and by whether they are loose or tight. Sound knots are knots that are as hard as the surrounding wood and shows no sign of decay. Tight knots are sound knots. Unsound knots in the strictest sense contain decay or rot but loosely defined, they may include splits or holes. Unsound knots are generally sawn off. Loose knots are knots that are loosely bound to the board and capable of dropping out, thus resulting in a hole.

Holes are cut out based on size only. Generally they are cut out if greater than half an inch. If they are less than half an inch it would depend on manufacturing requirements.

Wane is defined as bark or an absence of wood. Wane can be characterized by its length and width.

A **Check** is a lengthwise separation of wood that usually extends across the rings of annual growth and commonly results from stresses set up in wood during processing. Width and length can be used to characterize checks.

A **Split** is a lengthwise separation of wood due to the tearing apart of wood cells. Splits can be characterized in the same way as checks. The main difference between a split and a check is that a split starts at the end of a board while a check is always completely inside the outline of the board.

Due to the relatively small widths that are possible with splits and checks, a final requirement of the machine vision system is that it must be able to process high spatial resolution image data. A spatial resolution of up to 64 points per inch might be needed to detect small splits and checks.

5.4 Vision System Requirements

Based on the discussion provided by the previous sections, the following requirements were determined for the machine vision system:

(1) The machine vision system must be flexible enough to accomodate changes in inspection requirements. This requirement is necessary because manufacturing requirements may change from day to day. Changes in these manufacturing requirements may result in changes in the definitions of removable defects. Hence inspection requirements could change.

(2) The machine vision system must perform its inspection in a species independent manner. This is necessary due to the fact that most secondary remanufacturing plants handle multiple species of hardwood lumber.

(3) The machine vision system must be able to adapt to the wide variability in wood color.

(4) The machine vision system must be able to locate and identify knots, holes, splits/checks, and wane. These are among the most frequently occuring defects and were chosen for the purposes of establishing feasibility.

(5) The machine vision system must be able to process high spatial resolution image data. This is necessary in order to accomodate locate and identify splits and checks.

5.5 Previous Work in Wood Defect Detection

An early attempt to locate and identify surface defects in lumber is reported in (Conners (1983)). Black and white scans of wood surfaces were made and analysed. The images were $512 \times 512 \times 8$ in size with a spatial resolution of about 64 pixels per inch. The defects considered in this study were knots, mineral streak, decay, stain, wane, splits and checks, grub holes and holes, dark bark and light bark. The wood used in this study was southern red oak. Both tonal and pattern related measures were used in two different supervised classification procedures in an effort to locate defects in wood surfaces. An extension of this work used three different color bands (Conners (1985)). These were the red, green and blue bands. A fourth band (black and white) was added and was actually computed from the average of the three bands. Only tonal measures were used in this study.

The studies done in (Conners (1983)) and (Conners (1985)) used supervised classification procedures that could not be extended across species of lumber because of the wide variability of color and tonal qualities encountered. A further deficiency with the work described in (Conners (1983)) and (Conners (1985)) is the use of large region sizes of 64 and 32×64 respectively. The use of such regions do not allow an accurate characterization of small defects such as holes where the diameter of the hole is one of the parameters considered in classifying the hole as a removable defect. Last but not least, the computational complexity of the algorithms used limited the speed of classification.

Other work done in surface defect detection in wood is described in (Paul (1988)) and (Koivo (1989)). The vision system described in (Paul (1988)) is basically a simple system that uses fixed and adaptive thresholding and detection of high variation of gray level (detects roughness and chatter marks) within the image. Adaptive thresholding is accomplished by a low-pass digital circuit and is used to detect dark regions. Very bright and very dark regions are detected by a combination of fixed and adaptive thresholding. Very bright regions indicated open defects such as holes as the camera sees light from underneath the board. Very dark regions are considered to be knots, bark pockets, and shadows while dark regions indicate discoloration due to sound knots, decay, or stain. The system described did not address the issue of species independence. Furthermore, the criteria that dark regions be marked as defect would not work in species such as yellow poplar which have wide variation in brightness of clearwood, and in walnut where sapwood and heartwood could have a great difference in brightness.

Koivo and Kim (1989) uses causal autoregressive (CAR) models to determine the characteristic features of images. The CAR models represent the statistical dependency of the gray levels of each pixel, on the gray levels of its neighboring pixels. The estimated parameters of the model equations are used as feature vectors and input to a hierarchical classifier. The authors have suggested that the CAR model seems to be more reliable than the cooccurrence matrices used in (Conners (1983)). Unless both methods are tested on the same data or are theoretically compared we cannot come to such a conclusion. It is possible that the samples chosen in (Koivo (1989)) did not have as wide a variability within each class as the samples chosen in (Conners (1983)). Furthermore, the same samples were used in training and testing which could also further explain the high accuracy reported. In actual fact both methods as described in (Conners (1983)) and (Koivo (1989)) suffer from many of the same deficiencies described above. Also, the use of 64X64 regions do not allow the description of small defects such as holes where the diameter of the hole is important to its being classified as a removable defect.

In summary, the machine vision systems described in this section did not satisfy all the requirements listed in Section 5.4. They typically used supervised classification procedures that are not suitable for the wide variability in wood color. Furthermore, they do not address the issue of species independence. Finally, most use large region sizes of up to 64X64 for the analysis and are unsuitable for small defects such as holes. These systems were also based on computationally complex algorithms that limited the speed of classification.

5.6 Summary

As a specific example of a web inspection problem the location and identification of surface defects in hardwood lumber has been chosen to be the domain of application for the machine vision system to be designed by the machine vision prototyper. The rapid-prototyping of a machine vision system on the prototyper begins with problem definition. As such a study of the inspection problem was made to determined the requirements for the design of the machine vision system.

The description of the rough mill operation indicates the following requirements for the computer vision system: the system must 1) provide a simple mechanism for management to change inspection requirements, and 2) be able to handle a number of hardwood species. The nature of the color of wood indicated the need for a third requirement that the system must 3) be able to adapt to a wide variability in wood color thus implying the need for adaptive algorithms. The selection of a subset of defects imposed the following requirements: the system must 4) be able to locate and identify knots, holes, splits/checks, and wane; the system must 5) be able to process high spatial resolution image data of up to 64 points per inch.

Finally a brief discussion of previous work done in hardwood lumber inspection was provided and the shortcomings of each work were discussed.

Chapter 6

A MACHINE VISION SYSTEM FOR INSPECTING HARDWOOD LUMBER

The objective of this chapter is to describe the result of using the machine vision prototyper in designing a machine vision system for attacking the problem of locating and identifying surface features in surfaced hardwood lumber in a species independent manner. As stated in Chapter 1, there are two goals associated with this objective. The first goal is to demonstrate the utility of the prototyper in developing a machine vision system for a fairly complex planar web inspection problem. The second goal is to create at least a partial solution to the surfaced lumber inspection problem.

In accordance to the rapid-prototyping procedure depicted in Figure 3.1, Chapter 5 defined the hardwood lumber inspection problem to be attacked and provided a set of requirements for the machine vision system to be designed. The next step in the rapid-prototyping procedure is to configure an initial set of hardware and software components based on the given requirements.

6.1 Initial Hardware and Software Design

Chapter 5 generated the following general set of requirements for the machine vision system:

(1) The machine vision system must be flexible enough to accomodate changes in inspection requirements.

(2) The machine vision system must perform its inspection in a species independent manner.

(3) The machine vision system must be able to adapt to the wide variability in wood color.

(4) The machine vision system must be able to locate and identify knots, holes, splits/checks, and wane. A fifth feature class is also added to this list, that of an "unknown" feature. Unknown features are features that do not fall into one of the above categories and which the machine vision system is unable to identify.

(5) The machine vision system must be able to process high spatial resolution image data of up to 64 points per inch.

Given these requirements, initial hardware and software design entails the selection of components from an existing database of hardware and software components. Ideally, the components selected will be configured into a machine vision system capable of at least partially satisfying these requirements. The components include components that make up an **Imaging Subsystem**, an **Image Analysis Subsystem**, and a **Scene Analysis Subsystem**. The imaging subsystem is responsible for imaging a lumber board. The image analysis subsystem segments an image of the board into regions of uniform color and intensity. The scene analysis subsystem locates and identifies regions that are features.

The initial configuration of the machine vision system will be the first partial implementation of the rapid-prototyping procedure. The degree to which this first implementation satisfies all the requirements, i.e., how close this implementation is to the final version of a satisfactory machine vision system, depends on the number and reusablity of available components. The number and reusability of available components, in turn, depend on the number of times that the machine vision prototyper has been applied to planar web inspection problems. Since the hardwood lumber inspection problem represents the first application of the machine vision prototyper, most of the needed components were not available and had to be designed.

On the early version of the machine vision prototyper, existing hardware components included a standard black and white RS-170 solid state camera having a resolution of 512x480 with illumination provided by four tungsten halogen light lamps. To satisfy Requirements 2 and 3, the ability to obtain color images was needed. A **color image**, $\overline{f}(\overline{x})$, is defined to be a vector valued function,

$$\overline{f}(\overline{x}) = [r(\overline{x}), g(\overline{x}), b(\overline{x})],$$

whose independent variables $r(\overline{x})$, $g(\overline{x})$, and $b(\overline{x})$ are the red, green, and blue components of a pixel $\overline{x}$. Each of the red, green, and blue components can have a gray level value ranging from 0 to 255 with 0 indicating the complete absence of that component and 255 indicating a maximum intensity in that component. Color images were obtained by using red, green, and blue color filters with the black and white solid state camera.

To satisfy Requirement 5, the resulting color images had to have a spatial resolution of approximately 64 points per inch. This initial design of an imaging subsystem thus restricted the research by allowing only images of approximately 8 inch by 8 inch square areas of board surfaces to be considered. However, it facilitated the software prototyping of an image analysis subsystem and a scene analysis subsystem before an imaging subsystem capable of acquiring images of full size material could be designed. This is the nature of rapid-prototyping. In the rapid-prototyping approach to design, partial implementations are incrementally developed till a full prototype design results.

6.2 Software Prototyping

The next step taken in the rapid-prototyping procedure depicted in Figure 3.1 was the software prototyping of the image and scene analysis subsystem.

6.2.1 The Image Analysis Subsystem

To satisfy Requirement 5 on the list given in Section 6.1, the imaging system must be able to process high resolution imagery. Since some surface features to be detected may be very fine, a point-by-point segmentation scheme would be more suitable than the use of large region sizes. If we recall from Chapter 5, the use of large region sizes was one of the weaknesses of some existing lumber inspection systems.

To satisfy Requirements 2 and 3 listed in Section 6.1, the image analysis subsystem must be able to adapt to the wide variation in color found in different species and even within each individual specie of hardwood lumber. It is hypothesized that clear wood and the different features in hardwood lumber may be distinguished based on color. Given that the hypothesis is true, the goal of the image analysis subsystem would be to segment an image into regions that are uniform with respect to color. To determine if this hypothesis is true, one must first study the color characteristics of hardwood lumber as manifested in color images of the lumber. A way of studying the color characteristics of hardwood lumber is to study a **full-color histogram** computed from the color image of a piece of lumber.

To compute a full-color histogram, one must first understand the concept of a **color space**. The color space used in this work is a three-dimensional Cartesian coordinate system whose axes are the red, green and blue components of a color with each component having a range of from 0 to 255. This is illustrated in the shape of a cube as seen in Figure 6.1. The color of a pixel $\overline{x}$ can be represented by a point in this color space. Based on this concept of a color space, a **full-color histogram** can be defined to be a function $H(r, g, b)$ that gives the number of times a color with coordinates [r,g,b] in the color space, occurred in an image. If we are accurate in our belief that clear wood and other features in hardwood lumber can be distinguished based on color, then, the different features in wood should form distinct clusters in their full-color histograms. If these clusters have fixed positions for all species of wood, then a point-by-point supervised classification procedure can be derived to locate and identify the different features in lumber. However, due to the wide variability of wood color, we know that such a supervised classification procedure is not possible.

Figure 6.2 shows a full-color histogram of clear wood samples taken from the oak, maple, and walnut images shown in Figures 5.1, 5.2, and 5.3. From this

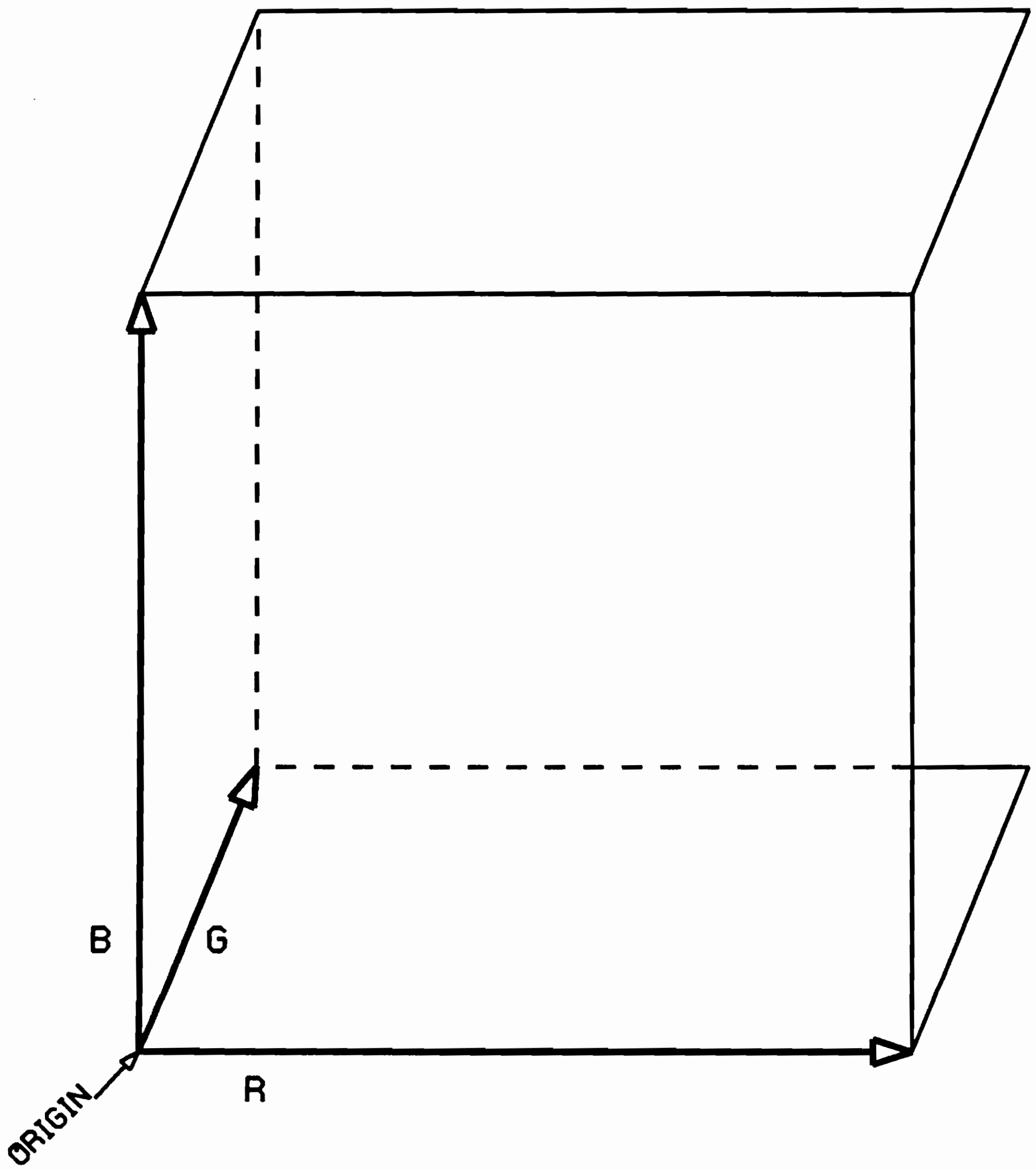

Figure 6.1. The three-dimensional color coordinate system used in this study. The red, green, and blue axes each has a maximum value of 255, thus making the color space a cube.

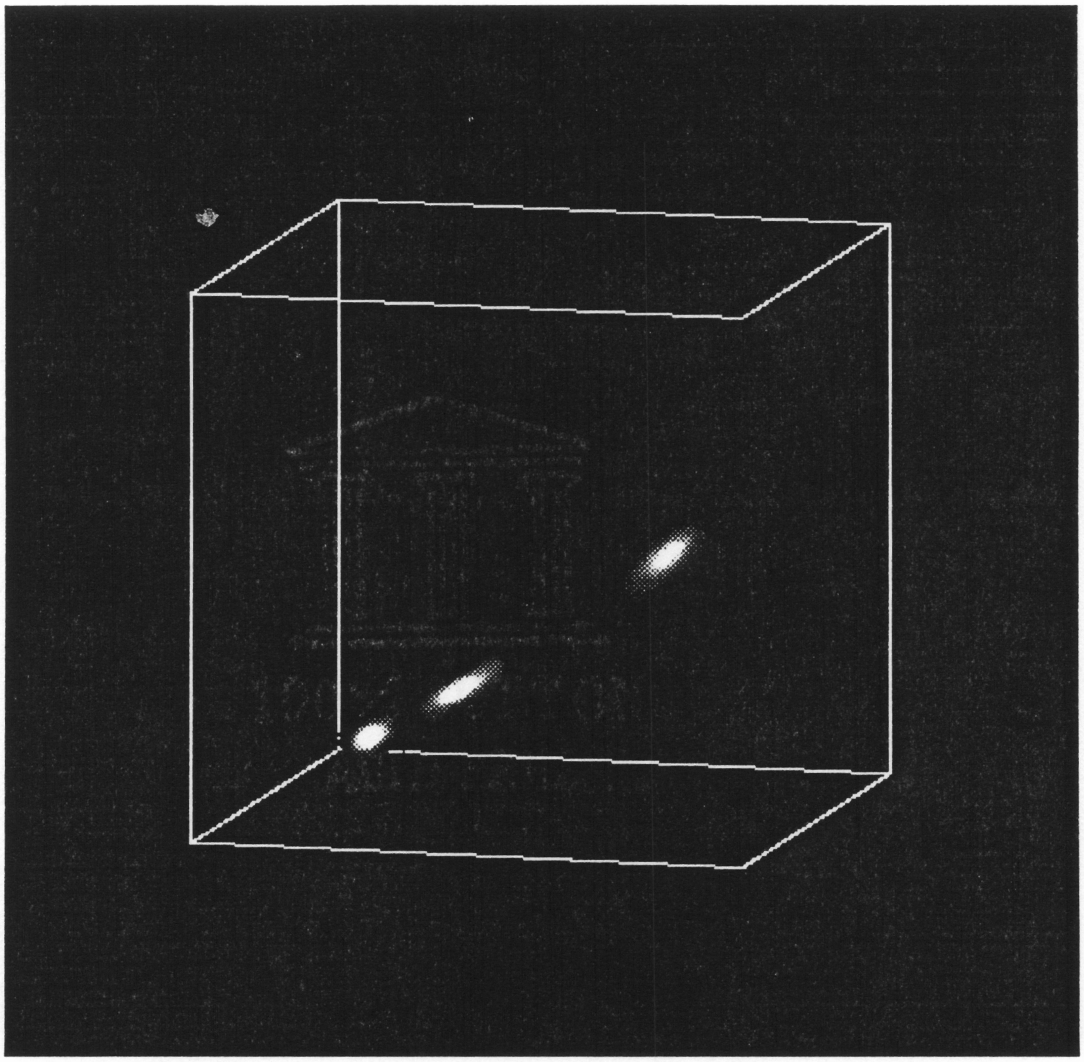

Figure 6.2. Full color histogram of clearwood samples of walnut, oak, and maple.

figure it is clear that even clear wood samples from these three different species cluster differently in a full-color histogram. The need for image analysis methods that can adapt to such a wide variation in color across different species, can thus be seen. Given that adaptive methods are needed, the most obvious way would be to compute full-color histograms for every board and to perform unsupervised cluster analysis on these histograms. However, due to the huge volume of memory required to store even a single full-color histogram (16 megabytes to be exact), such a method may be computationally expensive.

All the information in a full color histogram, however, may not be necessary. The analysis of images (Conners (1985)) indicate that partial-color histograms of the red and blue color bands may provide sufficient information for distinguishing features and clear wood based on these two color bands. A partial-color histogram is defined in the same way as the full-color histogram except that it has a reduced number of color coordinates. For example, the partial-color histogram $H_p(r,b)$ gives the number of times a color, whose red component is r and blue component is b, occurred in an image. In this dissertation, a partial-histogram is also called a scattergram. Figures 6.3, 6.4, and 6.5 show the scattergrams of the red and blue color bands for the oak, walnut, and maple images of Figures 5.1, 5.2, and 5.3 respectively. On these scattergrams, the upper left hand corner corresponds to the origin, the side extending from the upper left hand corner down to the lower left hand corner corresponds to the red-axis, and the side extending horizontally from the origin to the upper right hand corner corresponds to the blue-axis.

From these scattergrams we can see that background, clear wood, and other surface features of a hardwood lumber board tend to form separate clusters. Clusters can be seen as "hills" in the scattergrams. The cluster corresponding to

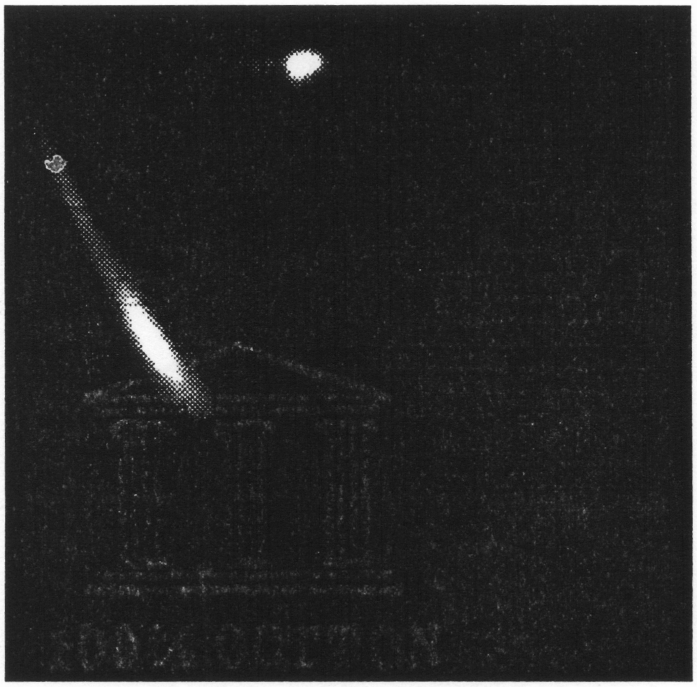

Figure 6.3. Two-dimensional histogram of Figure 5.1. The two axes are red and blue.

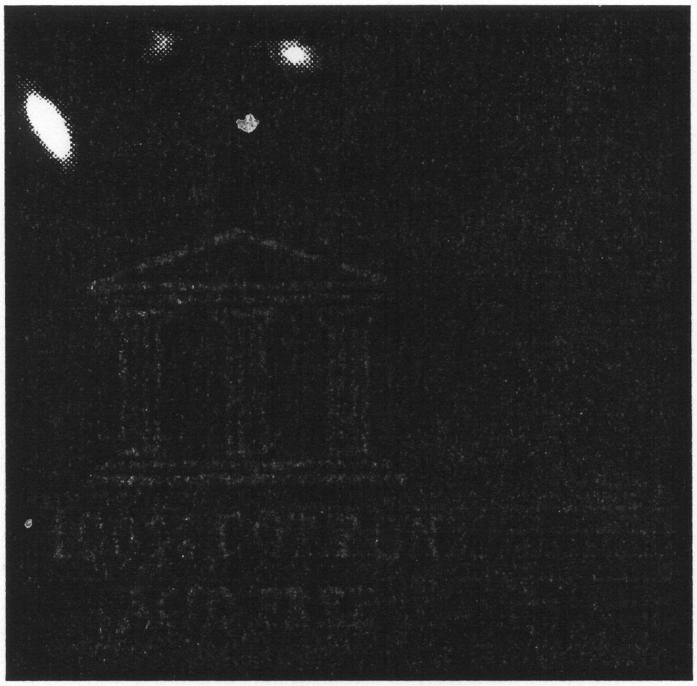

Figure 6.4. Two-dimensional histogram of Figure 5.2. The two axes are red and blue.

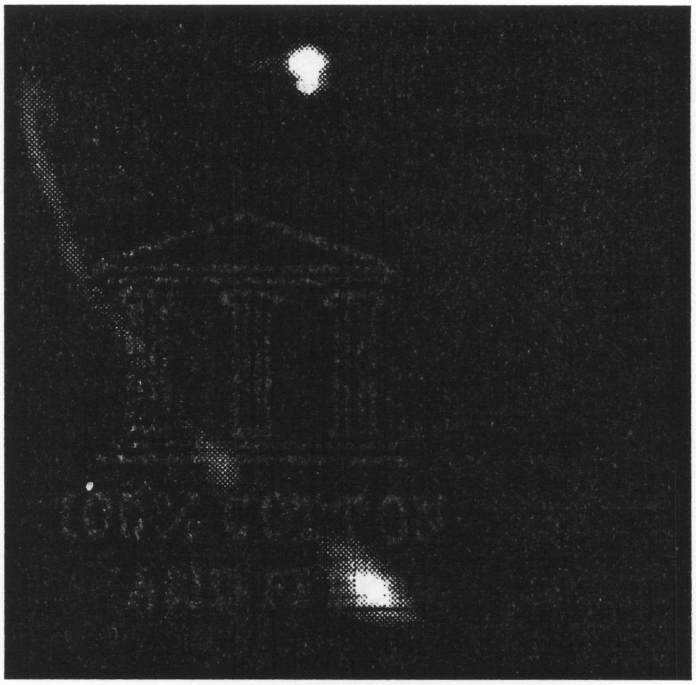

Figure 6.5. Two-dimensional histogram of Figure 5.3. The two axes are red and blue.

background can be found closest to the upper right hand corner in each of these scattergrams. This indicates that the background is blue in color. The choice of background color is addressed in Subsection 6.2.1.1. Clearwood can be seen as the largest most distinct cluster while the other features correspond to the darker areas attached to the clearwood cluster. The areas of the scattergrams corresponding to these features are not as bright as the clearwood areas because surface features of boards typically account for less than twenty percent of the total surface area of each board while clearwood accounts for the rest. Therefore, the largest cluster will always be clearwood. The background cluster could, conceivably, be larger than the clearwood cluster depending on the size of the board being inspected, but we can always identify the background cluster since the color of the background can be fixed.

From an examination of these scattergrams, it would seem that a method of separating features from clear wood is to detect the clusters in the scattergrams. Such a method would be adaptive in that no *a priori* information is needed on the absolute position of the clusters in scattergrams. In fact, no such information exists due to the wide variability of color in wood, even within any single species of hardwood lumber. However, these scattergrams seem to indicate that, on a local basis, i.e., on a single board, pixels belonging to clear wood and other features of hardwood lumber can be grouped into separate clusters. This points to a way of adapting to the wide variability of wood color. It in fact suggests that, to segment an image of a board, one would just need to locate and separate clusters in a scattergram computed from the image of the board. Pixel points in the image can then be classified as belonging to a particular cluster, with each cluster representing a particular surface feature or clear wood. Furthermore, since we know that the largest cluster will always be clearwood, we also have a simple way of identifying the clearwood cluster. Scattergrams could thus be used as look-up tables in an adaptive point-by-point classification scheme, once clusters within these scattergrams have been located and separated. This look-up table classification technique was originally reported by Goldberg and Shlien (1978) for the analysis of Landsat satellite imagery.

The technique just described is a simple, yet effective technique for segmenting images. It has several advantages over the techniques used in most of the previous work described in Chapter 5. Its main advantage is its speed. Instead of having to analyze the original image, the problem is reduced to the analysis of a 256x256 scattergram matrix, assuming a maximum of 256 gray levels in each color band of an image. This has a great deal of appeal as images to be analyzed could be prohibitively large (like in hardwood lumber images, for example). Most of the previous work on this problem performed the analysis on the original image. Furthermore, this method is adaptive, unlike the supervised classsification techniques used in previous work described in Chapter 5. Due to the wide variability in wood color, this method would thus be superior to these supervised techniques. Finally, in being able to perform a point by point classification, this method is better equiped to detect very small features such as holes, splits, and checks.

In this dissertation, the above methodology for segmenting images of hardwood lumber was incorporated into an image analysis subsystem composed of the following components:

(1) A background detection module.

(2) A scattergram computation module.

(3) A look-up table derivation module.

(4) An image segmentation module.

Modules 1 to 3 provide the image segmentation module (Module 4) with the information it needs to segment an image using the adaptive point-by-point classification scheme described above. There is actually another module in addition to the 4 listed above. This module provides shading correction. Since shading correction is concerned with a hardware problem, it will be discussed in the section on hardware prototyping. The following subsections will describe each component of the image analysis subsystem. Each such component corresponds to a reusable object consisting of well-defined data and object-manipulators that operate on the data.

6.2.1.1 The Background Detection Module

A common problem in designing machine vision systems for industrial inspection is the need to differentiate background from objects that are to be further analyzed. Typically, image background regions comprise a significant portion of images to be analyzed, and valuable processing time is spent identifying these background regions. Thus there is a need for a fast and simple method for identifying background regions. Once the background regions are identified, more complex image analysis algorithms may be applied to the object identified without wasting processing time on the background.

A method of background/object differentiation is the use of a background color that will trivialize this operation. The idea is to choose a color so that two planes of a color image can be used to segment the image into background and object regions. This can be done by using a 256x256 look-up table denoted by

B(i,j) where i and j are the gray level values of a pixel, in each of the two image planes, used to index into the look-up table. In a "look up" operation, the entry in the table whose position is given by i and j will correspond to either a background or not-background symbol. The region in the look-up table corresponding to the background can easily be determined by examing a scattergram computed from an image witb fully illuminated and shadowed background areas. To derive the look-up table from the scattergram, a closed region is first drawn around the cluster representing the background. The scattergram can then be converted into a look-up table by replacing points within the the closed region with a pre-determined background symbol.

The color of the background should be chosen so that background regions of an image will be represented by a cluster that is located as far away as possible from other clusters in the scattergram computed from an image. This is to eliminate, if possible, any overlap between background clusters and clusters corresponding to an object. In this research, a blue color background was chosen. The color blue was chosen because the color of wood typically contains a significant red component and almost no blue component (Conners (1989)).

Based on the above discussion, a background detection module was designed. This module, labeled **BACKGROUND**, is composed of image data inherited from the imaging subsystem, a look-up table defining the background color, an object manipulator *F_mask()* that will create a binary image divided into a background region and an object region. An image of a look-up table suitable for locating background regions of the images in Figures 5.1, 5.2, and 5.3, is shown in Figure 6.6.

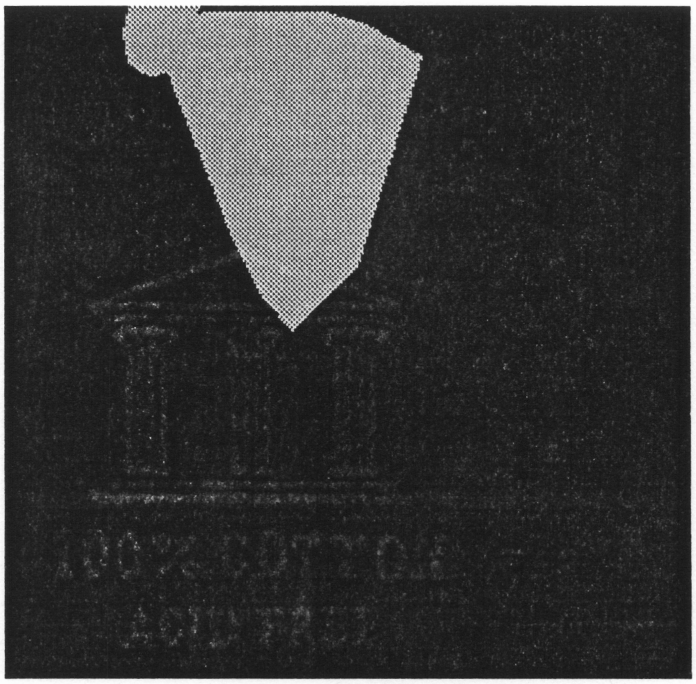

Figure 6.6. Look-up table for locating background regions of Figures 5.1, 5.2, and 5.3.

The differentiation of background pixels from object pixels is performed on a line by line basis. For each line, we first determine if each point on the line belongs to the background or not. Then we find the points on the line where object edges occur. Pointers are used to define the positions of these edges. Currently, these pointers are used to create a binary image with regions corresponding to the background and regions corresponding to an object marked by two different symbols. This binary image can be used as a "mask" for inhibiting or permitting further processing of a pixel point in an image to be analyzed, depending on whether each point is a background pixel or not. Figure 6.7 shows the "mask image" for the image in Figure 5.1.

6.2.1.2 Scattergram Computation Module

To segment an image into uniform regions based on how these regions form clusters in a scattergram of the red and blue image planes, the ability to compute these scattergrams is needed. For this reason, a module, labeled **SCATTERGRAM**, was developed. This module consists of the image inherited from the imaging subsystem, the mask image inherited from the **BACKGROUND** module, and the object-manipulator *F_sct()* that will compute a scattergram from the red and blue planes of a hardwood board image.

By using the mask image to inhibit processing of background regions, the scattergram is computed only from pixels corresponding to the board. An element in row i and column j of the scattergram is computed by counting the frequency of occurrence of gray level i in the red image plane and gray level j in the blue image plane. This is done for all possible combinations of i and j, resulting in a

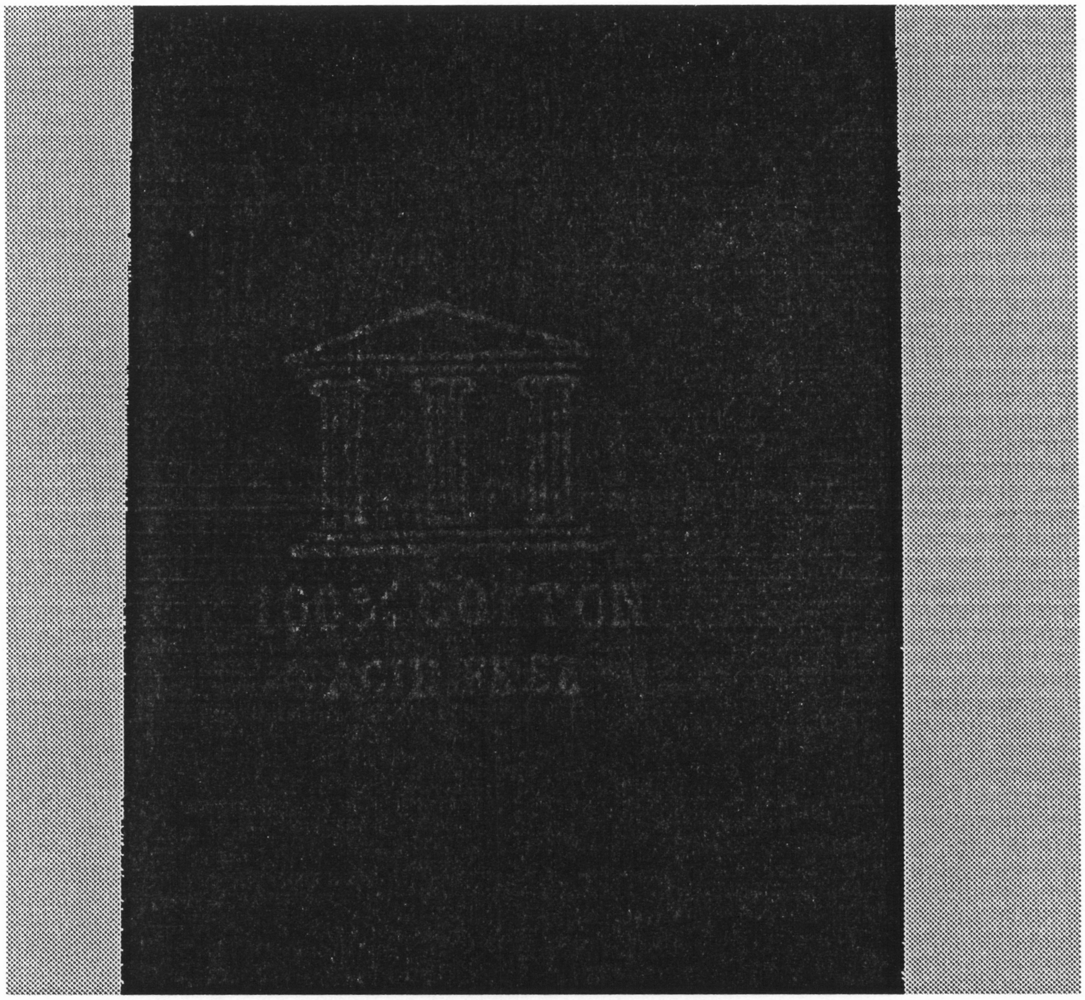

Figure 6.7. "Mask image" for the image in Figure 5.1.

two dimensional 256x256 array. For the images shown in figures 5.1, 5.2, and 5.3, the corresponding scattergrams are shown Figures 6.8, 6.9, 6.10.

6.2.1.3 The Look-up Table Derivation Module

Once a scattergram of an image is available, the next step is be to locate and separate clusters in the scattergram. As was previously discussed, clusters appear as "hills" in a scattergram. A simple way of locating these "hills" is by thresholding the scattergram. Elements of the scattergram with values above the threshold correspond to elements that are part of these "hills." This method of locating clusters was first proposed by Goldberg and Shlien (1978). To determine an appropriate threshold, Goldberg computed the average value per element of the scattergram. This average value can be found by computing the sum of non-empty elements of the scattergram and dividing this sum by the number of these non-empty elements. This average value is used as the threshold value.

For this dissertation research, better results were obtained by examining the frequency of occurrence of each value of the non-empty elements of the scattergram. It was found that elements of a scattergram array with relatively high values are closer to prominent peaks in the scattergram and tend to have a lower frequency of occurrence. Elements with lower values corresponding to the lower and "flatter" areas of the scattergram tend to have a higher frequency of occurrence. If we plot the frequency of occurrence of each possible value of the elements of the scattergram, against the value itself, we obtain plots such as those seen in Figure 6.11. Figures 6.11(a), 6.11(b), and 6.11(c), were computed from the scattergrams of Figures 6.8, 6.9, and 6.10, respectively. As can be seen, all three plots look

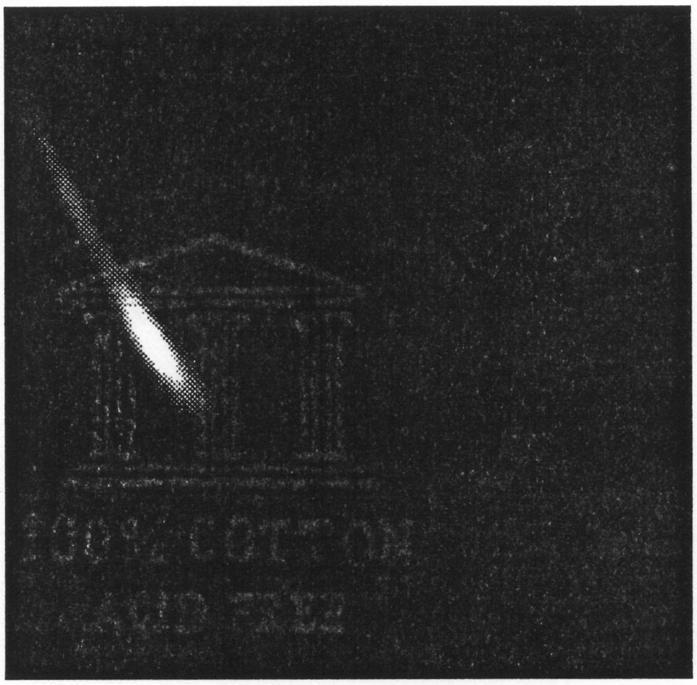

Figure 6.8. Two-dimensional histogram of Figure 5.1 without the background. The two axes are red and blue.

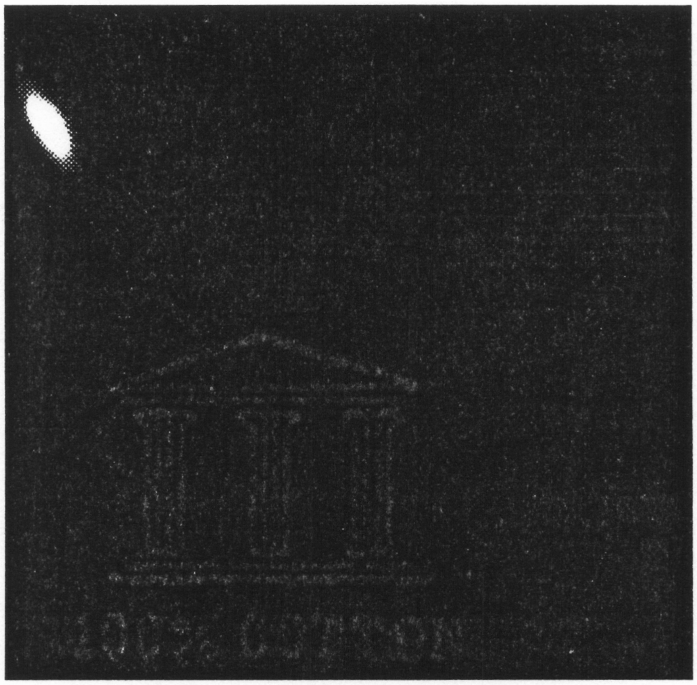

Figure 6.9. Two-dimensional histogram of Figure 5.2 without the background.

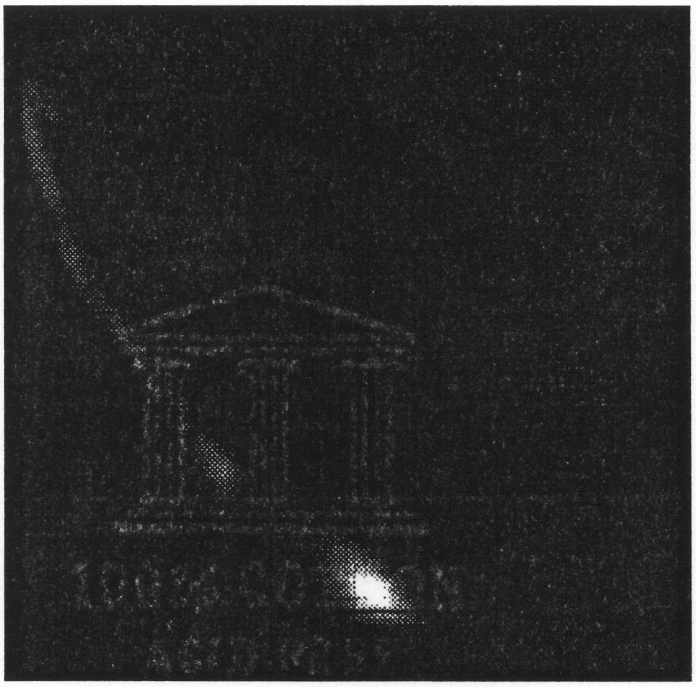

Figure 6.10. Two-dimensional histogram of Figure 5.3 without the background.

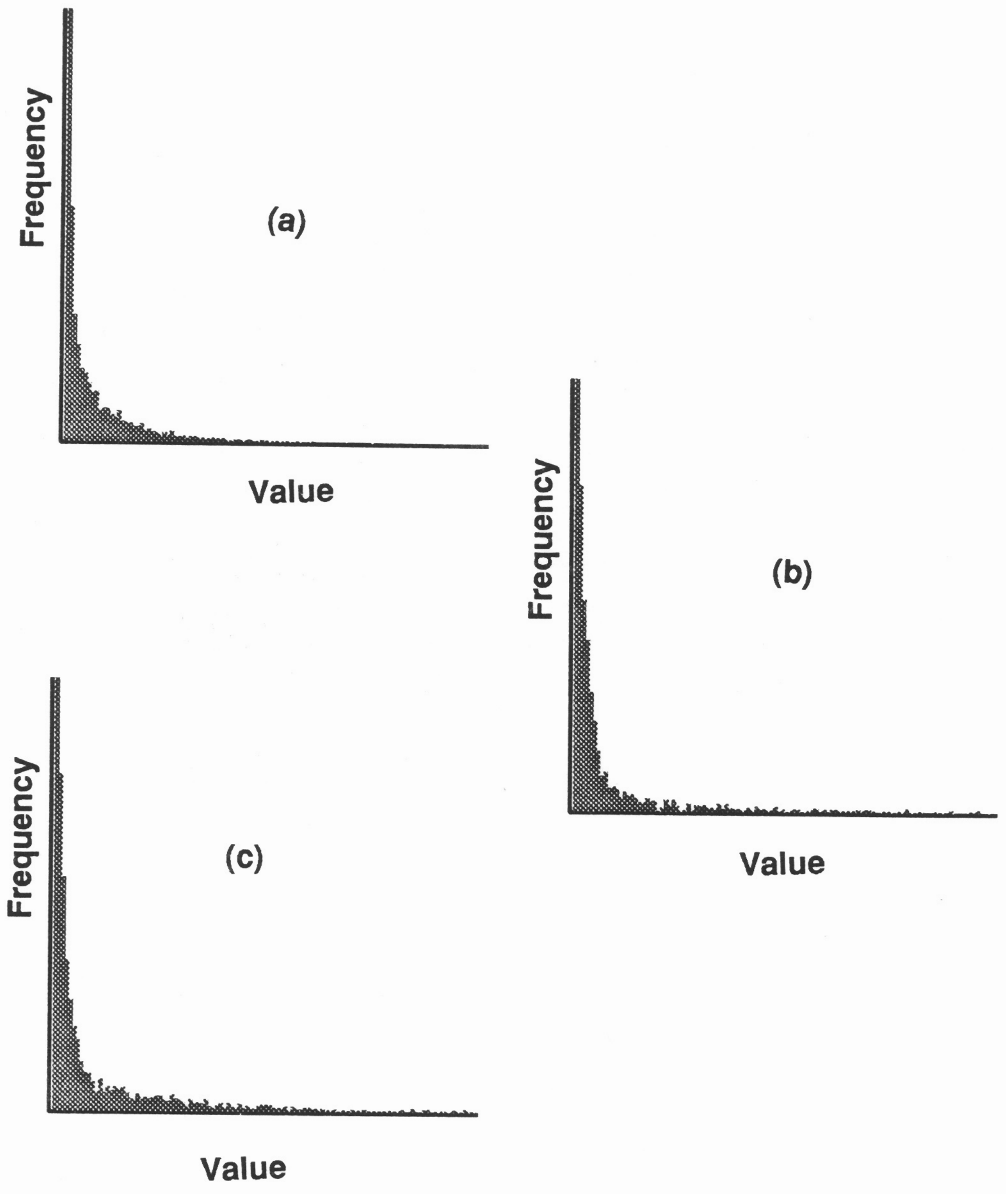

Figure 6.11. Plot of frequency of occurrence of the values of the the elements of the scattergrams in Figures 6.8, 6.9, and 6.10.

similar. In fact, all such plots computed by the author from scattergrams of other wood images are characterized by similarly shaped curves. These plots are typical of scattergrams that have "hills" representing clusters, surrounded by larger flat regions. It was found that good threshold values can be selected at a particular gradient value of these curves. This gradient value was selected by trial and error.

Having selected a suitable threshold value, elements of the scattergram with values below this threshold are eliminated. A connected components algorithm (Haralick (1992)) is then applied to the remaining regions in the scattergram. Each connected region found is given a unique label. The points that were eliminated during the thresholding operation are given the label "unknown."

Simple thresholding of the scattergram, however, is often not sufficient for separating all clusters. This usually occurs only with the largest cluster found. Unfortunately, the largest cluster always contains the clear wood cluster as clear wood usually makes up at least 80 percent of board surfaces. Since the largest cluster always contains the clear wood cluster, it is important that non-clear wood clusters be completely separated out to eliminate the possibility of classifying non-clear wood features as clear wood.

Fortunately, for the defects being considered in this research, it was found that non-clear wood clusters that were not separated from this large cluster could be more easily detectable in one-dimensional histograms computed by summing together elements of this large cluster in the scattergram along lines perpendicular to the main diagonal (line from the origin of the scattergram to the opposite corner of the scattergram). A multi-thresholding algorithm (Haralick (1992)) based on the detection of valleys in these histograms provided the methodology for separating the merged clusters. The lowest points in each valley are selected as threshold values. Perpendicular lines passing through these points in the scattergram provide the separating lines for splitting up the cluster under further analysis.

The steps for deriving look-up tables from scattergrams are given below:

1. Compute scattergram.

2. Threshold scattergram.

3. Perform connected components labeling on remaining regions in the thresholded scattergram.

4. Compute a one-dimensional histogram by summing elements of the largest cluster in the scattergram in a direction perpendicular to the main diagonal.

5. Set thresholds at the lowest points of valleys in the histogram.

6. Split up the cluster under further analysis by projecting boundary lines from the selected thresholds in a direction perpendicular to the main diagonal.

To derive look-up tables from scattergrams, the module **LOOKUP_TABLE** consisting of a scattergram inherited from the scattergram module, and an object-manipulator *F_seg()* for converting the scattergram into a look-up table, was designed. The algorithms for deriving look-up tables are incorporated into *F_seg()*. The results of applying the above algorithms to the scattergrams in Figures 6.8, 6.9, and 6.10 are shown in Figures 6.12, 6.13, and 6.14, respectively.

In Figures 6.12 and 6.13, small clusters can be seen that are separated from the clusters corresponding to wood. These clusters correspond to background regions

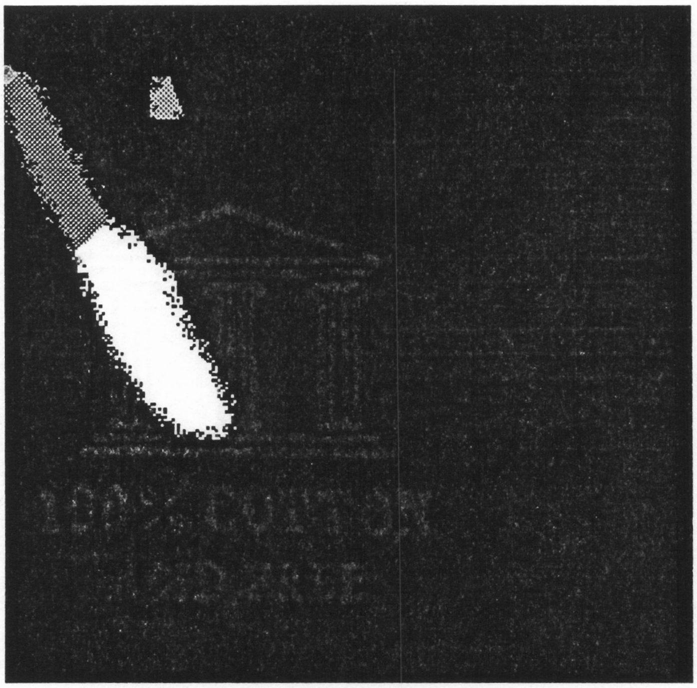

Figure 6.12. Segmented histogram of Figure 6.8. This can be used as a look-up table for classifying the image in Figure 5.1.

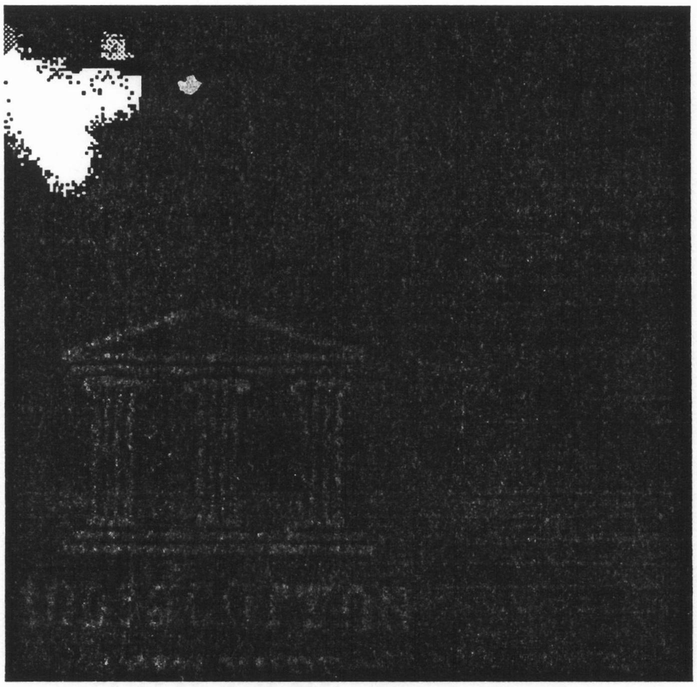

Figure 6.13. Segmented histogram of Figure 6.9. This can be used as a look-up table for classifying the image in Figure 5.2.

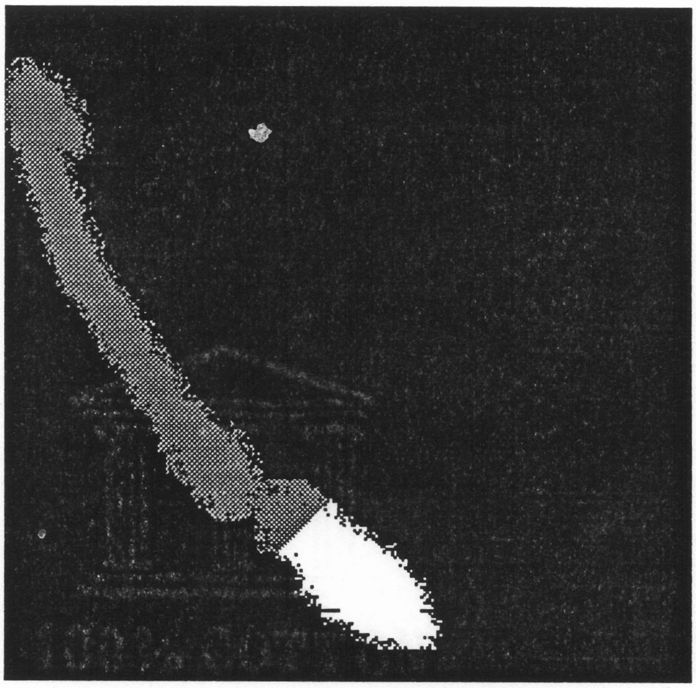

Figure 6.14. Segmented histogram of Figure 6.10. This can be used as a look-up table for classifying the image in Figure 5.3.

very close to the edge of the boards that were not detected by the **BACKGROUND** module. This is because the "region of definition" of the background, in the look-up table for the background (see Figure 6.6), had to be conservatively picked to avoid an overlap with clusters corresponding to wood. The reason for this can be seen in Figure 6.13 where the small "leftover" background cluster is very close to the larger wood cluster.

6.2.1.4 The Image Segmentation Module

The image segmentation module, labeled **SEGMENTED_IMAGE**, inherits the image of a lumber board from the imaging subsystem, the look-up table from **LOOKUP_TABLE**, and the mask image from **BACKGROUND**. Image segmentation is performed in this module in the following steps:

(1) Classify each pixel point in an image using the look-up table.

(2) Perform noise reduction.

(3) Perform connected components labeling on classified image.

(4) Mark connected regions that are close to board edges.

The object-manipulators implementing each of these steps are described below.

Pixel classification using look-up table

Using the look-up table, the object-manipulator *F_cls()* classifies each pixel of the board image to produce a segmented image. *F_cls()* uses the mask image to determine if a pixel point belongs to the background or to the lumber board before it tries to classify the pixel. To classify a pixel, the red and blue values of the pixel are used as row and column coordinates of an element in the look-up table array that identifies the class to which the pixel belongs.

Noise reduction

The initial segmentation, however, usually has many misclassified noise points and some form of preprocessing is useful to reduce this noise. The object-manipulator *F_thin()* was designed to reduce this noise. The processing is performed in a 3x3 window. The noise reduction steps are given below:

1. If the pixel under consideration is surrounded by at least 6 neighbours of the same gray level, the pixel is changed to that gray level.

2. If the test fails in (1) and if the pixel is labeled "unknown" and if none of the neighbours are "unknown", it is then relabeled with the label of the highest frequency among its neighbours.

Points labeled "unknown" are points that could not be placed in any cluster in the look-up table during the classification procedure.

The results of applying *F_cls()* and *F_thin()* to the images in Figures 5.1, 5.2, and 5.3 are shown in Figures 6.15, 6.16, and 6.17. The results of the processing before *F_thin()* was applied is not shown because the noise points are very small (mostly single pixels) and are not obvious in pictures printed by the laser printer available (the laser printer does not have sufficient resolution for the sizes of the images seen in the figures).

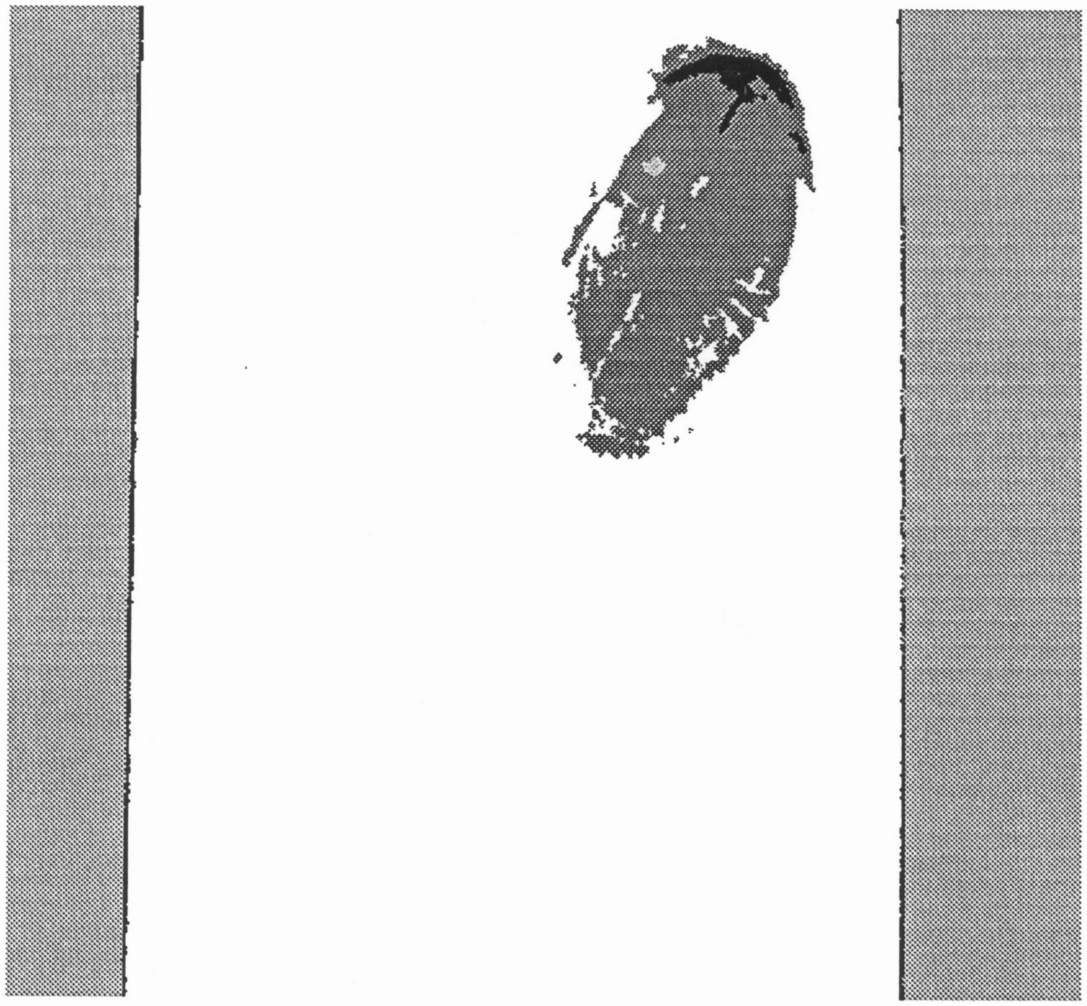

Figure 6.15. Results of the classification of Figure 5.1.

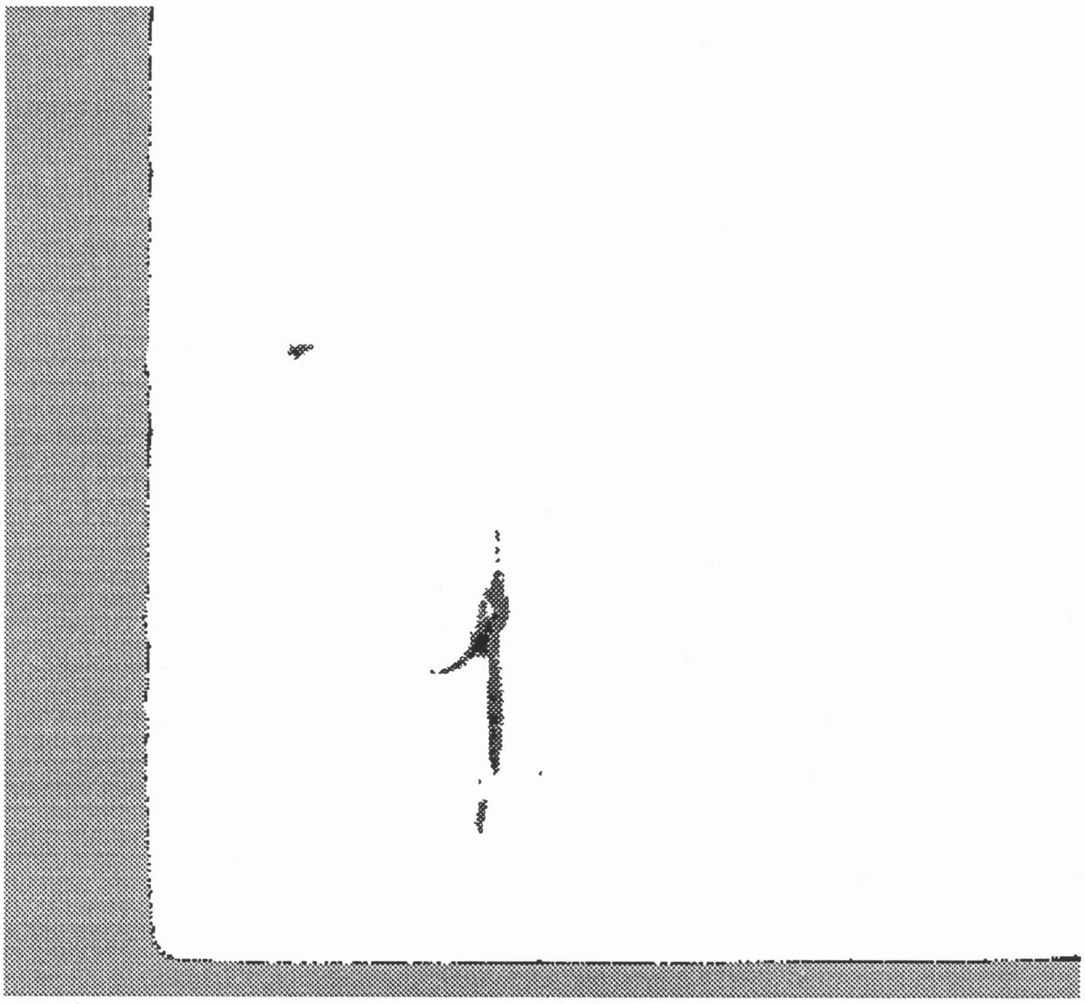

Figure 6.16. Results of the classification of Figure 5.2.

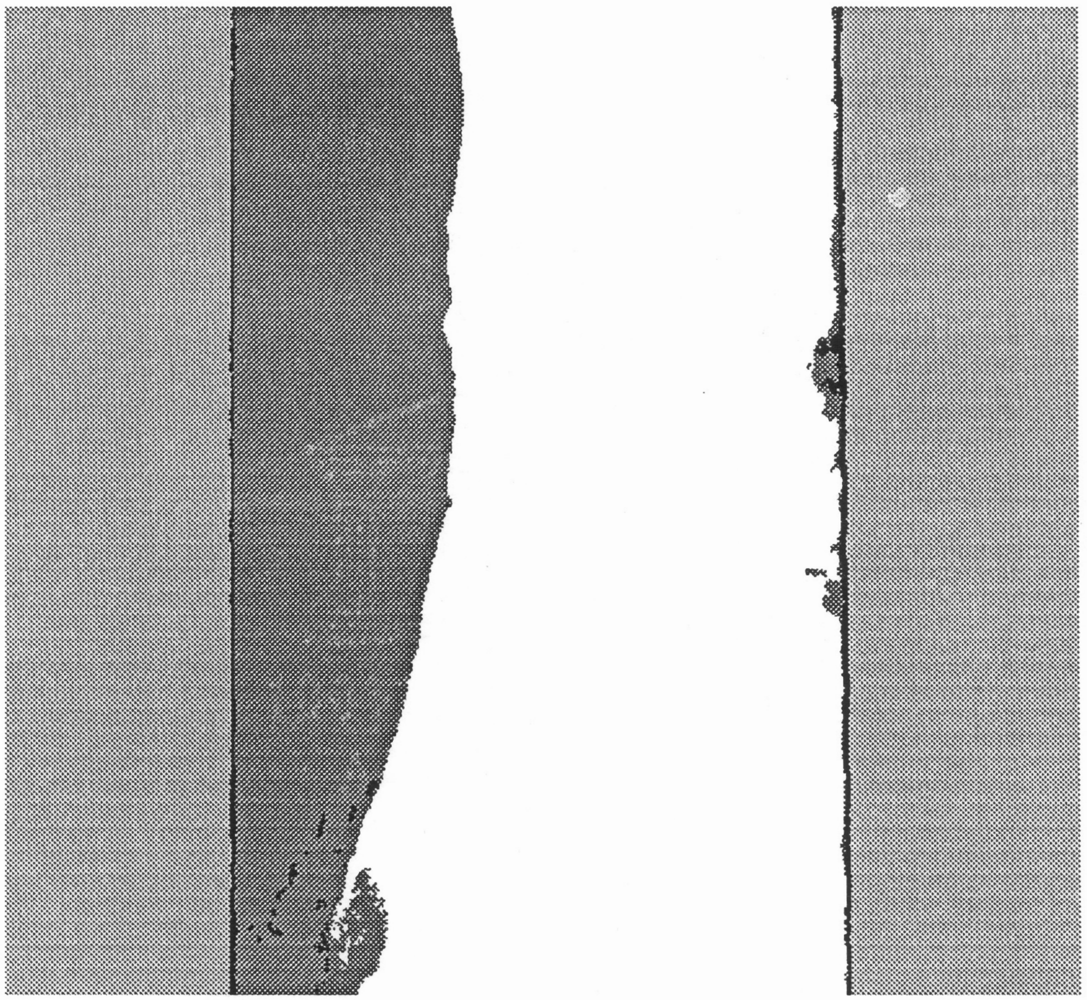

Figure 6.17. Results of the classification of Figure 5.3.

Connected components labeling

After the initial segmentation is performed, connected regions are formed by grouping pixels belonging to the same cluster. Connected components labeling is performed by the object-manipulator *F_dfct()*. Each connected region found is given a unique label to differentiate it from other connected regions.

The connected components labeling algorithm used is an efficient run-length implementation (Haralick (1992)) of a two-pass connected components labeling algorithm that uses a local equivalence table (Lumia (1983)). In this method, a run-length encoding of every line in the image is extracted and the connected components labeling algorithm applied to the more compact run-length encoded image. Processing results at the end of this stage are shown in Figures 6.18, 6.19, and 6.20.

Mark connected regions close to board edges

After connected components labeling has been performed, the object manipulator *F_bord()*, marks connected regions that are close to board edges. This information is needed by the scene analysis component of the machine vision system. The scene analysis component is described in the next section.

6.2.1.5 The Scene Analysis Subsystem

At present, only a simple scene analysis subsystem has been implemented. This is because most of the research effort was aimed at the segmentation of board images. The image segmentation problem is a very difficult problem and much work remains to be done particularly in the segmentation of board images with

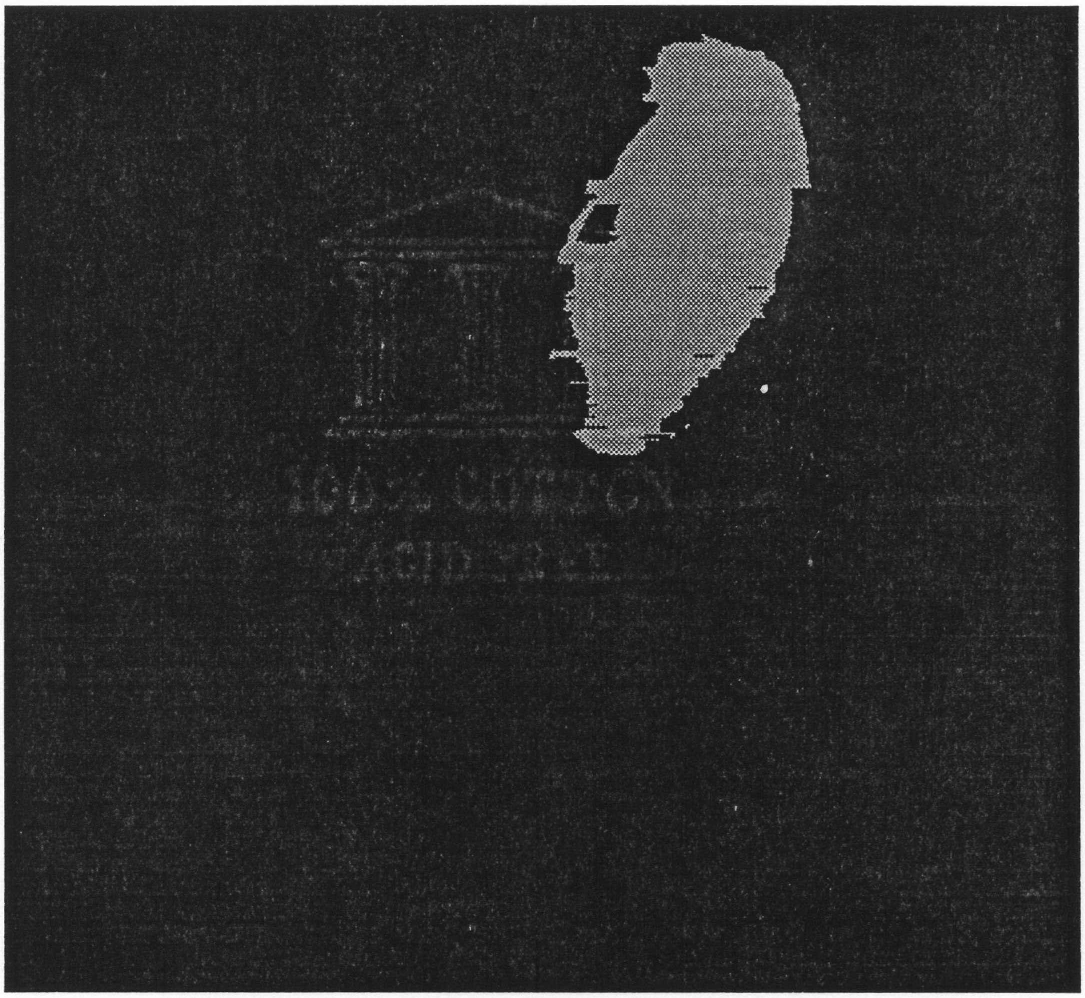

Figure 6.18. Connected components analysis of Figure 6.15.

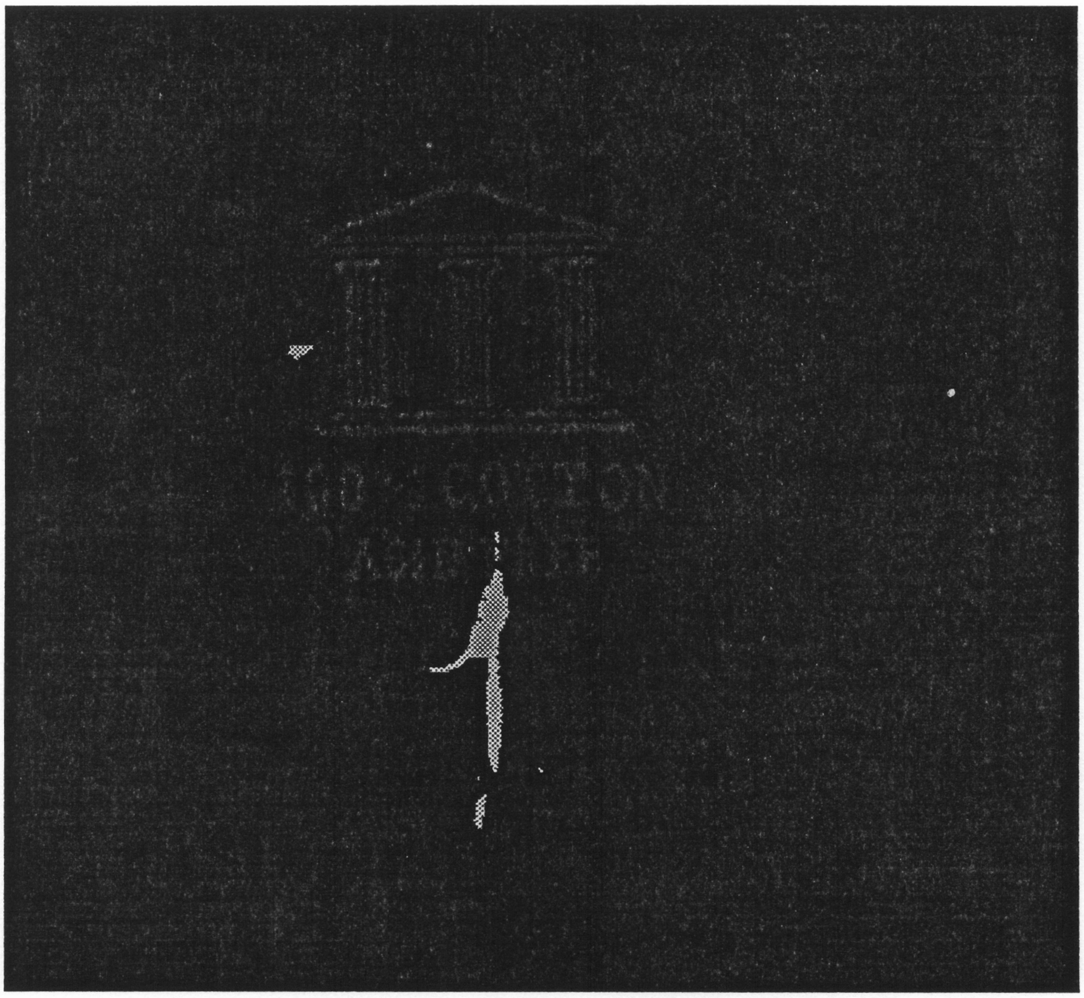

Figure 6.19. Connected components analysis of Figure 6.16.

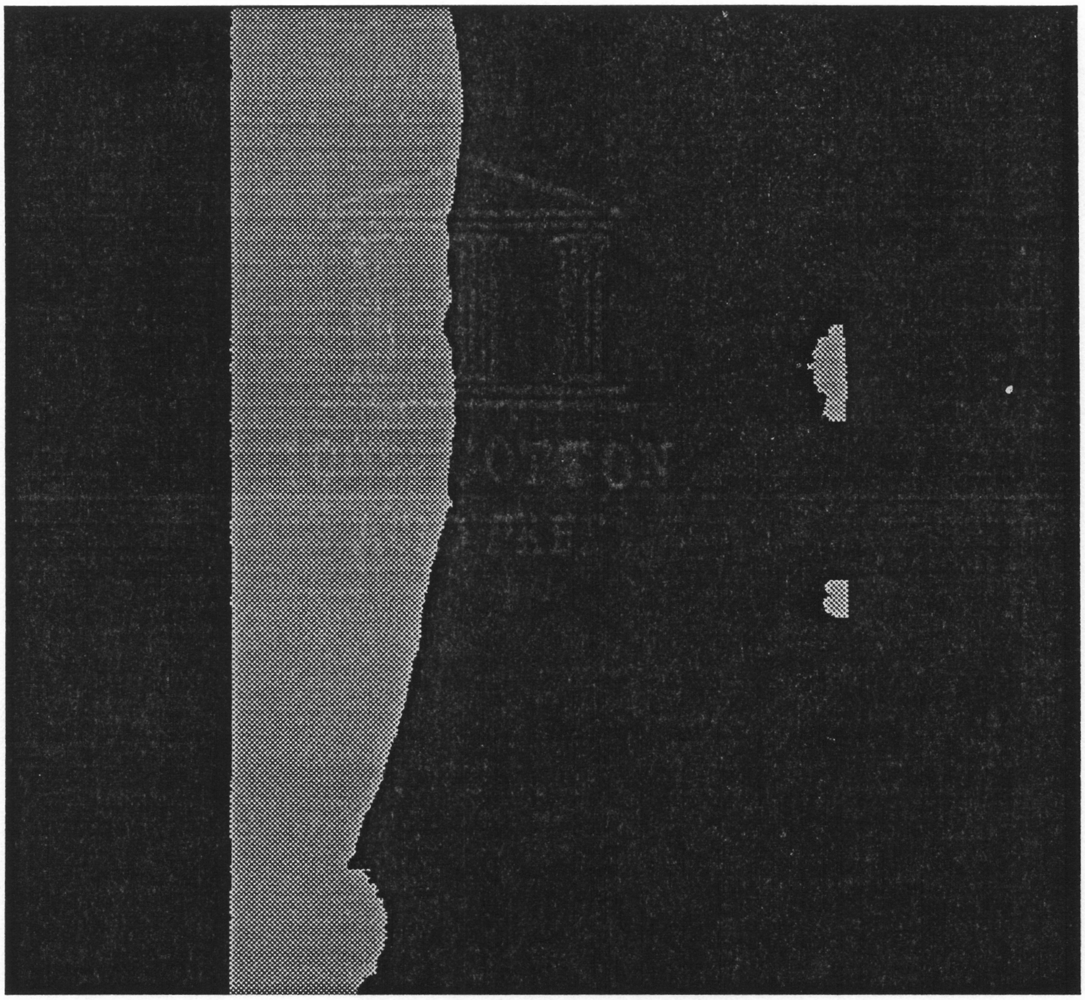

Figure 6.20. Connected components analysis of Figure 6.17.

features such as decay and blue stain. As such, more effort was concentrated on this problem.

The requirements of the machine vision system were given in Chapter 5. To satisfy Requirement 1, i.e., that the machine vision system must be flexible enough to accomodate changes in inspection requirements, the scene analysis provides the capability to redefine defects. This capability is provided through the specification system described in Chapter 4. The scene analysis reads the definitions of removable defects from a specification file that is easily modified by a user through the specification system. The current values of these definitions are given below. To satisfy Requirement 4 of the requirements for the machine vision system, the scene analysis subsystem is designed to identify knots, wane, splits/checks, and holes.

The scene analysis subsystem is made up of a set of object-manipulators that function as independent knowledge sources. These object-manipulators operate on the segmented image inherited from the image analysis subsystem and on the scattergram of the image inherited from the scattergram computation module. The first object-manipulators to be activated, *X_cut()*, computes enclosing rectangles around connected regions. It also removes very small regions. The main function of *X_cut()* is to coordinate the execution of the other object-manipulators. From this point on, each object-manipulator performs its analysis based only on the coordinates of the enclosing rectangles.

To reduce the amount of processing in the scene analysis subsystem, some preprocessing is first performed on small regions in the segmented image. Small regions close to each other are merged by object-manipulator *Check_small()* while small regions that are enclosed by larger regions are deleted from further consideration by the object-manipulator *Check_large()*.

Given the coordinates of the enclosing rectangles, a series of object manipulators are then activated in an attempt to identify each of the connected region in a segmented image. The object-manipulators attempt to identify each region representing a surface feature of a board one at a time in the following manner:

1. First, *is_clear()* checks to see if the feature could actually be Clearwood that was misclassified. It does this by checking to see if the average gray value is close to Clearwood.

2. Next, *F_group()* looks at the area occupied by the feature. Table 6.1 gives the probable feature type based on the size of the feature. *F_group()* identifies the group that the feature has a high probability of being apart of. This object-manipulator tries to narrow down the possibilities to help out the next object-manipulator.

3. Then, *label()* looks at the ratio of height/width of the enclosing rectangle and tries to identify each hardwood feature. The rules used are as follows:

Let **ratio** = **height/width** of an enclosing rectangle. From the previous processing, we now have a list of possible board features. The order of this list is important as will be seen.

Let the initial defect TYPE default to the first possibility on the list.

If $2 \leq$ **ratio** < 3 and **height** > 300 *then* TYPE = WANE.

If **ratio** ≥ 3 *then* change TYPE default to SPLIT.

If $3 \leq$ **ratio** < 4 and **height** < 15 *then* TYPE = HOLE.

Table 6.1

DEFECT GROUPS BASED ON AREA.

Area (no.of pixels)	Defect type
area < 20	probably split or unknown
$20 \leq$ area < 100	probably hole or split
$100 \leq$ area < 200	probably hole, knot or split
$200 \leq$ area < 1000	probably knot or split
area ≥ 1000	probably knot, wane or split

If **ratio** ≥ 3 and **width** > 50 *then* TYPE = WANE.

The next step depends on the result of the previous rules applied.

If AREA < 1000 pixels and defect is on the boundary of the wood *then* TYPE = UNKNOWN.

The last rule recognizes the fact that a number of small regions at the edge of the board are very dark shadows (near black) and are sometimes misclassified as holes. Furthermore, small defects at the edge of the board may be affected by these dark shadows as they form a significant percentage of the total defect (the defects being small).

The results of the recognition system are shown in figures 6.21, 6.22, 6.23.

6.3 Hardware Prototyping

As the previous partial implementation of the machine vision system neared completion, 4 feet long lumber boards became available for testing. It quickly became quite clear that this partial implementation would not be able to process these boards easily. The main problem lay in the imaging subsystem. This imaging subsystem was based on an area array camera that could only acquire images of an approximately 8 inch square area. Lumber boards, on the other hand, could be as long as 16 feet and as wide as 13.5 inches. The introduction of a new requirement such as this is typical in rapid-prototyping. The next step was therefore to design a more suitable imaging subsystem. The machine design effort at this point enters the hardware prototyping stage in the rapid-prototyping procedure depicted in

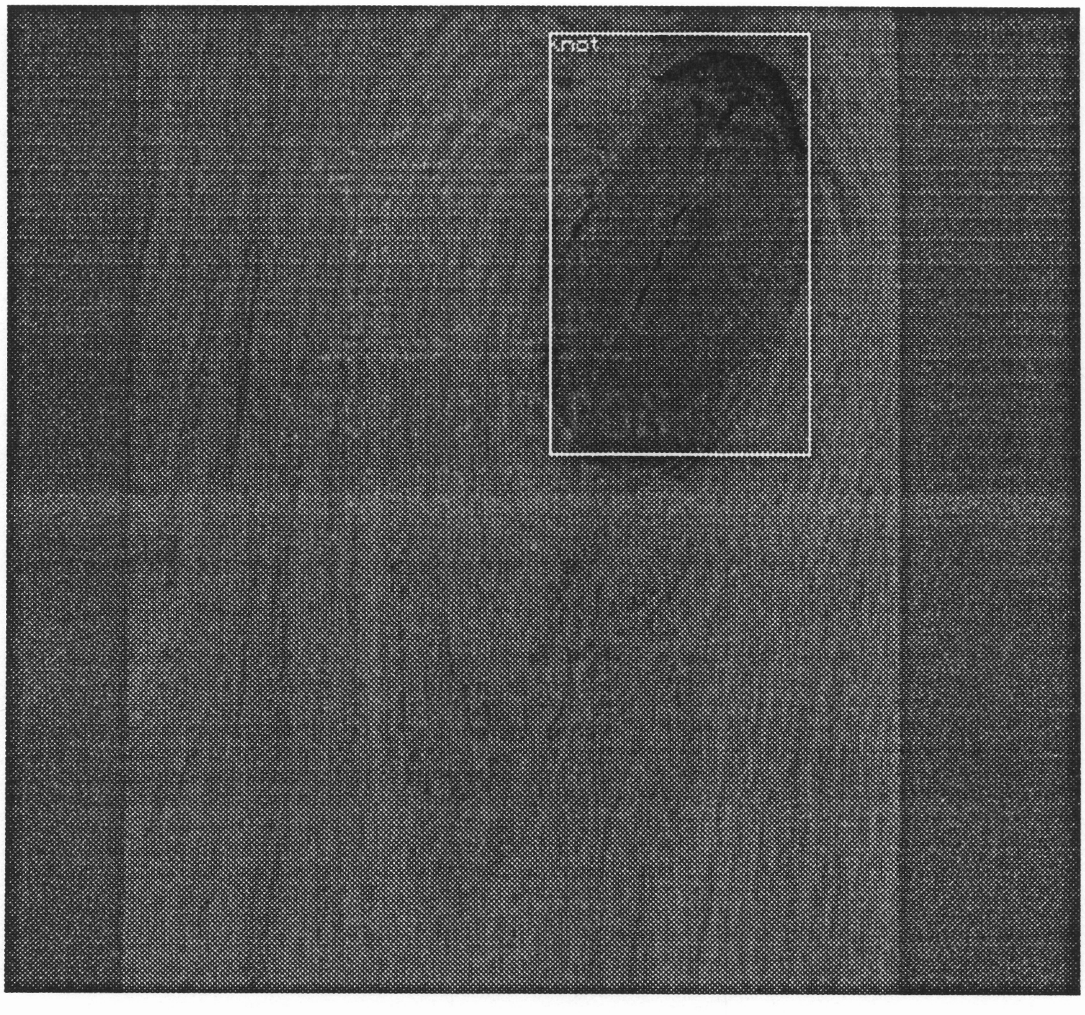

Figure 6.21 Results of the recognition system for figure 5.1.

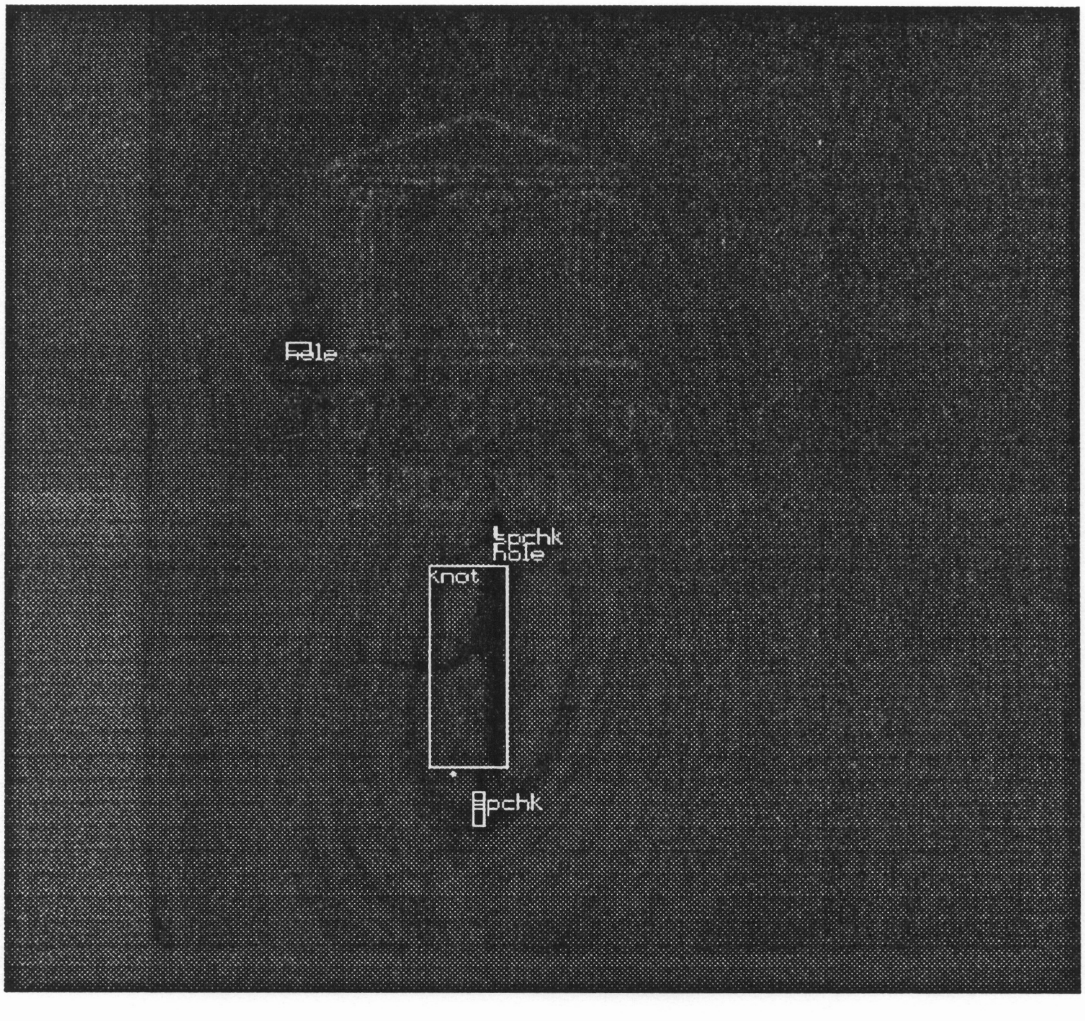

Figure 6.22. Results of the recognition system for figure 5.2.

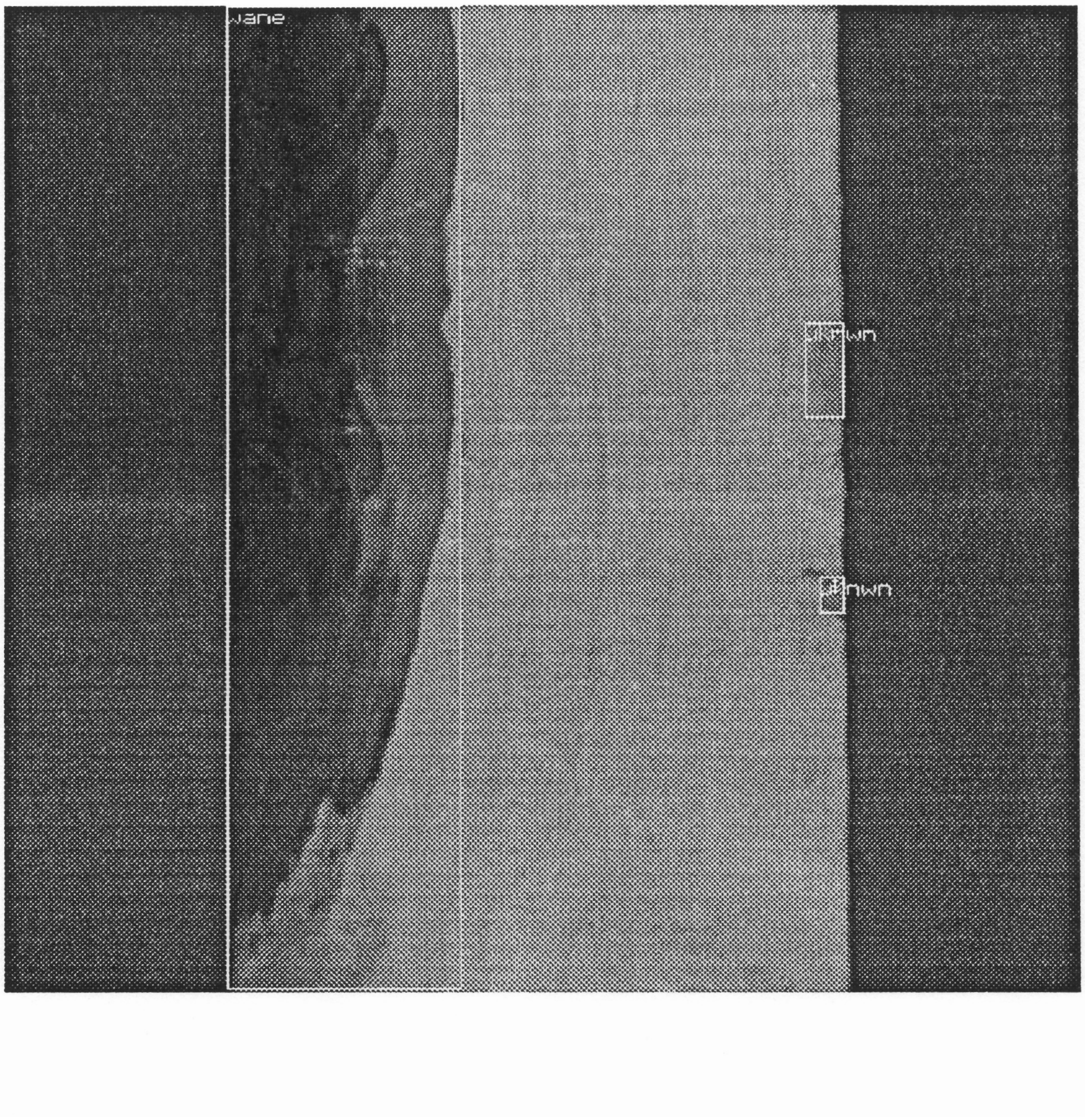

Figure 6.23. Results of the recognition system for figure 5.3.

Figure 3.1. The first step in the hardware prototyping was the selection of a suitable camera and is described in the next subsection.

6.3.1 The Color Camera

Planar webs such as lumber boards are usually moved along in a continuous motion for most of the way and it would not be convenient for them to be momentarily stopped to imaged. Therefore, the initial implementation of the imaging subsystem based on an area array camera would not be suitable. A line scan camera is a better choice as the web can be imaged while in continuous motion. Furthermore, the length of these webs may not be known in advance thus making the linescan camera a more sensible choice.

Another characteristic of planar webs is that the resolution of images across the width of the web and along the length of the web, need not be the same. This is because many features that need to be detected on these webs are long and thin with the length of the feature typically oriented along the length of the web. It would be an advantage if image resolution across the web and along the length of the web could be independently controlled. The **crossweb** direction is considered to be in the direction perpendicular to the longest side of the web. A line scan camera only permits crossweb resolution to be determined. Downweb resolution is a function of the speed of the materials transport system. This gives us independent control of downweb and crossweb resolution and a greater degree of freedom to choose these resolutions depending on the nature of the web. This is another advantage of a linescan camera.

At the time when cameras were being evaluated (October 1988) for selection, there was only one color line scan camera on the market. This was the Pulnix model TL-2600RGB. At the time of this writing, it is still the only color line scan camera on the market. It has 864 pixels per line of scan with each pixel having a dynamic range of 256 gray levels for each of the color bands red, green and blue. At 64 points per inch of spatial resolution, this camera will allow a 13.5 inch field of view, a field of view wide enough to handle the vast majority of hardwood lumber. The scanning speed of the camera is adjustable up to 2.5 megahertz. At the speed of 2.5 megahertz, the camera can generate images with a crossweb resolution of 64 points per inch and a downweb resolution of 32 points per inch, for lumber boards moving at two linear feet per second. Tests indicate that this is more than enough spatial resolution for the needs of the lumber inspection problem. These tests consisted of visually comparing images with 64 points per inch crossboard and 64 points per inch downboard resolution, against images with the same crossboard resolution but with 32 points per inch downboard resolution. The second set of images were derived from the first set of images by replacing alternate lines in the image with previous lines. As might have been expected, it was visually difficult to distinguish between the two sets of images. Currently, there are two of these lines scan cameras available. These cameras are each equiped with a 35 millimeter lens having a focal length of 50mm. This lens has a minimum f-stop of 1.8.

A problem with the color line scan camera is that the camera is a charged-coupled device (CCD). Such a device has a high response to the red and infrared components of the electromagnetic spectrum. To compensate for this problem, an infrared rejection filter was included as part of the lens system.

6.3.2 Light Source Selection

Given the appropriate selection of a camera system, the next step is to select the light sources. The main considerations involved in the selection of appropriate light sources are the need for uniform lighting across the area to be scanned and the ability to incrementally increase or decrease the light intensity on this area. Keeping these two considerations in mind, fiber optic light sources made by the Fostec Company were chosen. Each of these fiber optic light sources consist of a bundle of optic fibers, a tungsten halogen light bulb at one end and an enclosure at the other end permanently holding each individual fiber optic line stacked on top of one another and with their ends forming a straight line. This enables the light source to illuminate a linear region. One advantage of such a light source concerns the environment in which an industrial inspection system often operates in. The environment in a manufacturing plant is typically very dusty, making it necessary for the imaging system to be placed in an enclosure. The use of these fiber optic light sources enable these enclosures to be more or less permanently sealed as the light bulbs that provide illumination to the fiber optic light lines can be placed outside the enclosure. This allows a defective light bulb to be easily replaced. Currently 11 inch fiber optic light lines are being used. A configuration employing six such light sources is being used with four being used to illuminate the web face and two to illuminate the background.

The light bulb used is a 150 watt tungsten halogen bulb, a type of incandescent light source. In comparison with other incandesent light sources, tungsten halogen provide the most uniform illumination and spectral properties across the life of each bulb. Other types of incandescent light sources experience a reduction by as much as twenty percent in the amount of illumination across the life of the bulb. This figure is about two percent for a tungsten halogen light bulb.

Unfortunately, tungsten halogen bulbs do not have a flat spectral response across the visible spectral range of 400 to 700 nanometers. This response is marked by a high red to infrared component and a sharp drop off as the spectrum approaches the blue component. This response can be seen if we were to capture an image of a uniform white surface illuminated by tungsten halogen light lamps. The resulting image would be reddish in color. This problem was first noticed in the early images capture by the imaging subsystem. These images had a high red color content and were not suitable for analysis. The red/blue scattergram of one such image is shown in Figure 6.24(a). Notice that the points on the scattergram are very close to the red-axis of the scattergram.

To compensate for this problem, two blue filters that have characteristic response curves that are low towards the 400 nanometer end of the spectrum and high towards the 700 nanometer end of the spectrum, are employed. When these filters were in place, the images showed considerable improvement. Figure 6.24(b) shows the red/blue scattergram of the same scene imaged that resulted in the scattergram in Figure 6.24(a). As can be seen, the color balance has improved.

There is one light source that has a desirable flat spectral response across the 400 to 700 nanometer range. These are Xenon light sources. However, they require a very high voltage to start up, they contain gas under high pressure and thus have to be handled with great care, and they are also very expensive. On the other hand, they have a very good expected life time of about 1000 hours, which is at least ten times the life expectancy of the halogen bulbs being used. The use

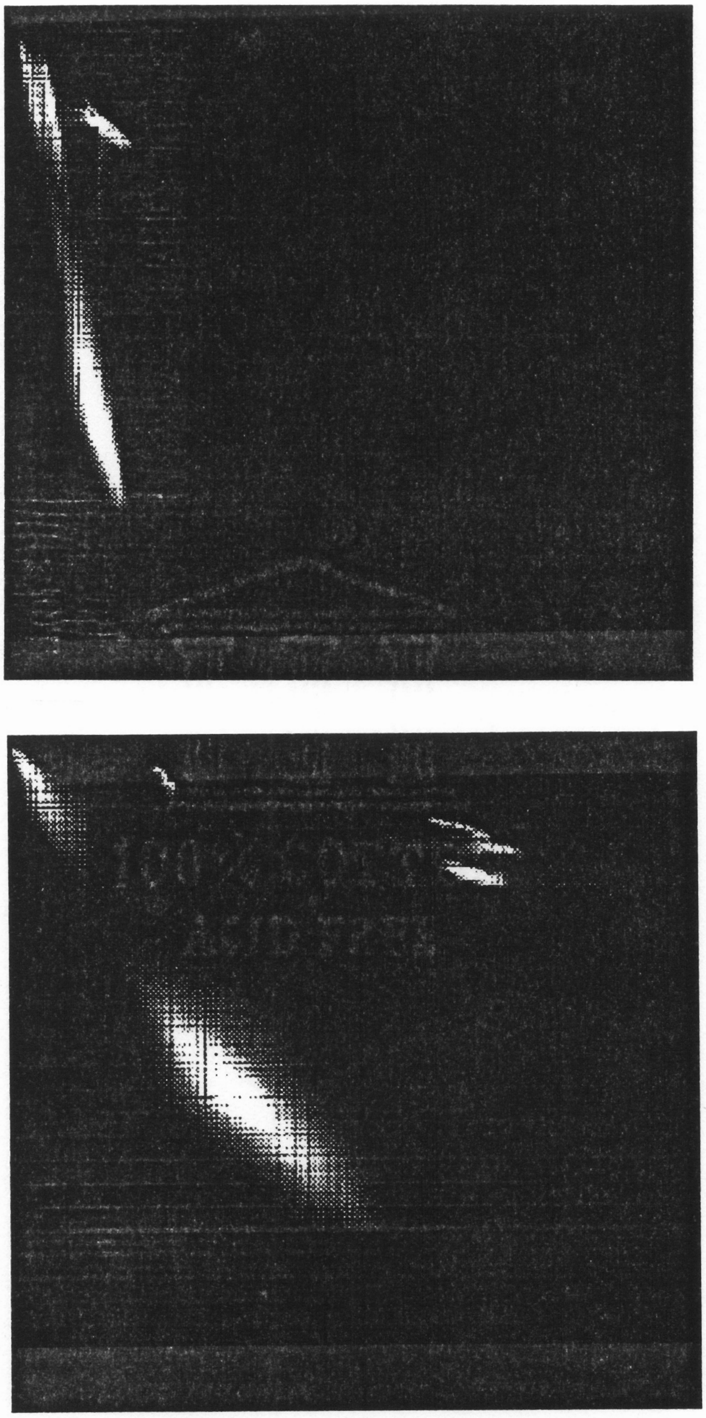

Figure 6.24. The red/blue scattergram of an image scanned without filters is shown in (a). After the addition of blue filters, the red/blue scattergram in (b) is the result.

of Xenon technology is being considered as a possible light source to use in the future.

6.3.3 Interfacing to the IBM PS/2 Model 80

To access the data from a camera, a computer interface is needed. Initially, a specialized parallel port interface was designed by Drayer (1990) for the Model 80 as an off the shelf interface for the camera did not exist. Data collection through the parallel port interface, however, is slow. The design of a high speed interface, on the other hand, is more complex and more time was required for its design. Therefore, to facilitate the prototyping of other machine vision components, the parallel port interface was first designed.

Together with the parallel port interface, comes the need for a software device driver. Most operating systems do not allow software written by non-priviledge users to communicate directly with external devices. The mechanism by which this is done is through software device drivers. For example, when a request for a read from a port connected to some external device is made by a user's process, the operating system takes control of the computer and selects the appropriate device driver to perform the desired function. Control is then passed back to the user process.

Under a **Unix** system, which, in our case is AIX, a software device driver has several functions it has to perform. These include device initializations, code to perform reads and writes and to send control information. It also has to provide some sort of handshaking or interrupt mechanism to allow the data to be transferred correctly and completely.

The device driver written for the parallel port interface has four functions. The first function is an initialization routine that is performed when the computer is first started (booted) up. In executing this initialization routine, the operating system establishes the link between a particular device and the driver routines that perform the different functions required by user processes communicating with the device.

The other three routines are directly related to the three types of Input/Output (I/O) request by user processes for the parallel port interface. One is the request to open the device. On such a call the operating system executes the open routine provided by the device driver. This open routine basically just enables the bi-directional mode of the parallel port. Another request is to close the device. For the camera interface and device driver, no action is necessary beyond returning an error flag that indicates that no errors were encountered.

Finally, the request by a user process to read causes the operating system to execute the read routine in the device driver. This routine will perform the necessary handshaking to enable the the correct transfer of data. Every request for a read results in the transfer of one scan line of data.

For higher scanning speeds, a high speed shared memory interface has been designed (Drayer (1991)) and is currently available. For this high speed interface, a device driver has also been written. The device driver merely provides the capability to access port addresses on the interface adapter. The strategy for communicating with the interface, which is normally implemented in the device driver, was thus left to the user/programmer.

6.3.4 Configuring the Imaging Subsystem

With the selection of a camera, lens, and light sources, and the design of camera interfaces, the imaging subsystem was completed by combining these components with the materials handling subsystem and object detection subsystem. Figure 6.25 shows a picture of the some of the components of the imaging subsystem mounted on the hardware prototyper.

The imaging subsystem which we label as **LINESCAN_IMAGER**, can be viewed as an "object" in the object-oriented sense. It is composed of the **OPTOSCAN**, the **LINEAR_STAGE**, and the **LINESCAN_CAMERA**, objects. Given the design of these objects, a brief digression into software prototyping was necessary to design the object-manipulators for directing an image capture sequence of operations. The object-manipulator *imgscan()* coordinates these three objects in an image scanning sequence to capture an image of a planar web. First, *imgscan()* sends a command to the **LINEAR_STAGE** object to begin transporting a web. When the web is detected by the **OPTOSCAN** object, it is stopped. This is to allow *imgscan()* to bring the web into position for scanning by the camera. Once the web is close to the field of view of the camera, it is moved again as soon as the **LINESCAN_CAMERA** object is activated by *imgscan()*. After a certain number of scans the web is returned to its normal position. As the image is collected, it is sent through the appropriate interface for further processing by the image analysis subsystem.

Once images could be captured for analysis, the imaging subsystem was ready to be tested. There were basically three major problems encountered in the imaging subsystem that had to be addressed during testing. The first of these concerns the

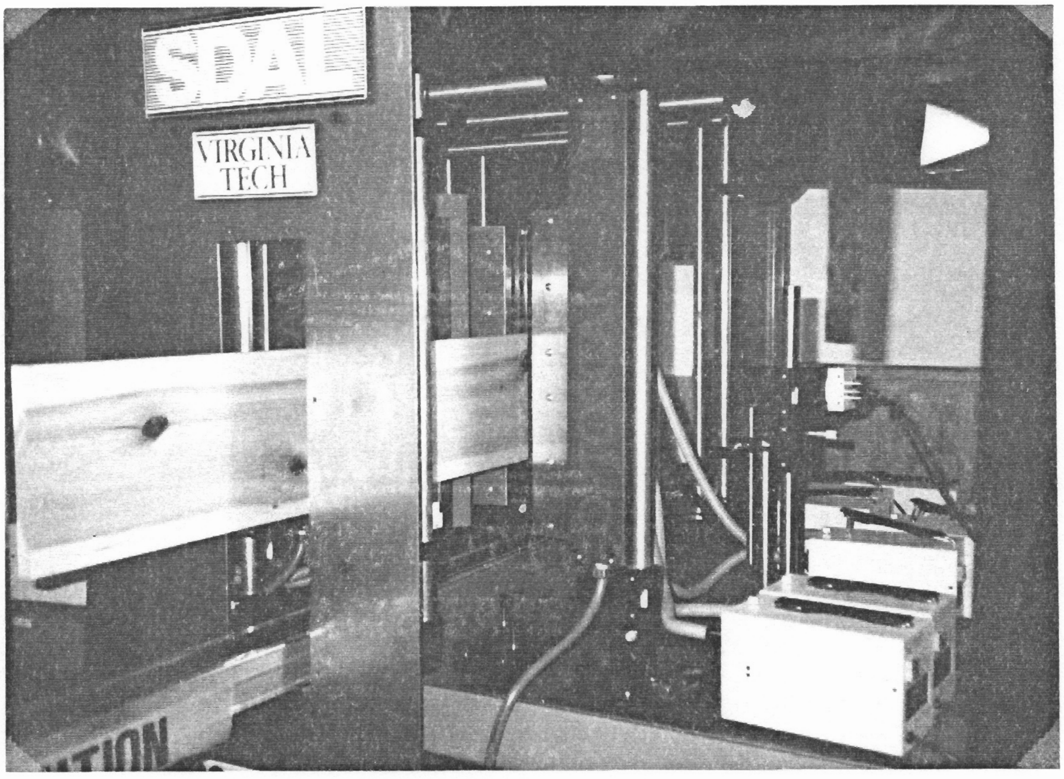

Figure 6.25. Picture of the hardware prototyper with some of the components of the imaging subsystem.

ability to relate each point on an image to its corresponding point on the web. Due to radial distortion of the camera lens, a procedure had to be found to reduce the effect. Another concern is the selection of an aperture size. Finally, shading across the field being imaged is not uniform due in part to some non-uniformity encountered in the light sources and in part to the characteristics of the lens. Therefore some form of shading correction procedure had to be found. Each of these problems are addressed in the next three subsections.

6.3.4.1 Radial distortion in scanning

An important consideration in the machine vision system is the ability to locate a defect accurately on the actual web we are analyzing. To do this we must be able to relate each point on the web surface to each point on the scanned image. There are two aspects of the problem to be considered. The first aspect concerns the downweb resolution which was discussed in Chapter 4. The second aspect of the problem concerns the resolution of each pixel on a scan line in the direction of the scan across the web. We shall denote this the **crossweb resolution.** Crossweb resolution is determined by limiting the field of view of the camera. The field of view of the camera can be determined by adjusting the distance of the camera to the object.

Due to radial distortion in the lens, however, we cannot relate points on the image directly to points on the surface of the object being scanned, with sufficient accuracy. This problem was solved by scanning an image with several points marked and performing a least squares approximation to relate the points on the image, to the points on the actual surface.

To simplify the analysis, a pinhole camera model is used. This is shown in Figure 6.26. From the figure we can see that, by similar triangles, $s = (\frac{l}{f})p$ where s is a point on the surface, p is a point on the image, f is the focal length of the camera, and l is the distance between the imaged surface and the pinhole of the camera. If we measure s in inches and p is given in terms of its pixel position (i.e., column number which is 1,2,..., or 864 for the color line scan camera under consideration) than we need to add a constant k to the above equation which would act as a unit conversion factor. The equation should now be:

$$s = (\frac{lk}{f})p$$

The goal of the procedure is to calculate $(\frac{lk}{f})$. In this way we need not be concerned with the focal length of the camera or the distance between the imaged surface and the pinhole of the camera or the unit conversion factor separately. Once we have the value of $(\frac{lk}{f})$ estimated we can directly calculate the position of any point on the surface from its corresponding pixel position in the image.

The least squares procedure used is explained below:

let $s(i)$ be the position of each marked point on the image surface, $p(i)$ be the corresponding pixel position in the camera image, n be the number of these points, f be the focal length of the camera, and l the distance between the image surface and the camera that is to be estimated. Minimizing the expression

$$\sum_{i=1}^{n}[s(i) - \frac{lk}{f}(p(i)]^2$$

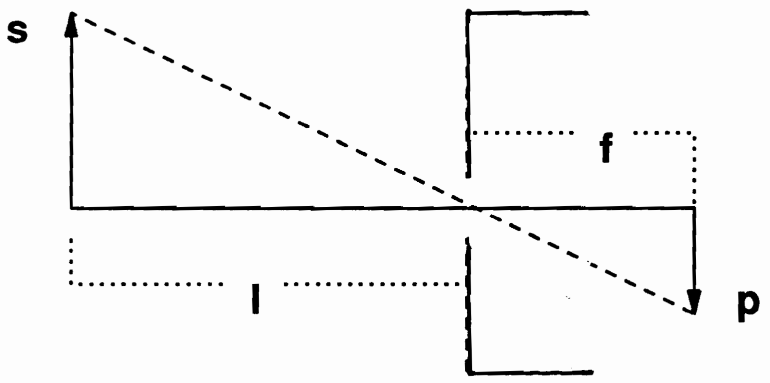

Figure 6.26. A pinhole camera model.

with respect to l results in the following steps for finding $(\frac{lk}{f})$:

1. Sum the squares of each pixel position in the image.

$$P = \sum_{i=1}^{n} p^2(i)$$

2. Sum the product of each known point on the image with its corresponding pixel position in the image.

$$SP = \sum_{i=1}^{n} p(i)s(i)$$

3. Calculate $(\frac{lk}{f})$.

$$\frac{lk}{f} = \frac{SP}{P}$$

6.3.4.2 Choosing an Aperture Size

Aperture sizes are specified on lenses in terms of what is known as the f-stop number. For example, an f-stop number of 1.5, designated as f/1.5 means that the focal length of the lens is 1.5 times the aperture size. Larger apertures allows more light to be gathered by the lens in a given time. Smaller apertures, on the other hand, have greater depth of field and also reduce the effects of certain types of lens aberrations such as spherical aberrations. Spherical aberrations occur because lens surfaces are sections of spheres and light rays passing through the edge of a lens comes to focus at a point different from where light rays passing through the center of the lens comes to focus (Price (1976)). Thus, the smaller the aperture, the closer are the paths of all the light rays to the center.

Ideally, we can mathematically model a lens to help us understand the nature of the lens aberrations and other characteristics of the lens but the necessary information about the lens is not available from the manufacturer. It is certain that the lens is a compound lens designed to reduce the effects of certain aberrations, but any configuration of lenses will not completely eliminate these effects.

Currently, the procedure for determining aperture size is by viewing the output of the line scan camera on an oscilloscope.

6.3.4.3 Shading Correction

Raw image data from a camera must often be shade corrected due to uneven lighting conditions and unequal distances from different parts of the image to the focal plane of the camera. This results in a nonuniformity in the output response of the camera to a uniform surface. The latter condition is especially noticeable as the camera is placed closer to the viewing surface. The nonuniformity in ouput response to a given input (such as a unformly white surface) can be modelled as a nonlinearity (Sawchuk (1977)). Let

$$g(i) = N\left(f(i)\right)$$

where $g(i)$ is the output response to a given input $f(i)$ and i denotes the pixel position. N denotes the nonlinearity function. The nonlinearity of the output response $g(i)$ about a fixed input value $f_0(i)$ for any value of i can be represented as a Taylor's series expansion as follows:

$$g(i) = N\Big(f_0(i)\Big) + \Big(f(i) - f_0(i)\Big)\frac{\partial N}{\partial f}\Bigg|_{f_0(i)} + \frac{\Big(f(i) - f_0(i)\Big)^2}{2!}\frac{\partial^2 N}{\partial f^2}\Bigg|_{f_0(i)} + \cdots$$

Collecting all the powers of $f(i)$ we obtain:

$$g(i) = a_0(i) + a_1(i)f(i) + a_2(i)f^2(i) + \cdots \tag{4.1}$$

where $a_0(i)$, $a_1(i)$, $a_2(i)$, $\cdots$, are constant coefficient terms. For most high quality sensors, the higher order terms of f (f^2 and above) in equation (4.1) can be neglected (Sawchuk (1977)). Therefore, the actual inputs $f(i)$ can be approximated by:

$$\hat{f}(i) = \frac{g(i) - a_0(i)}{a_1(i)}$$

which gives us the shading correction algorithm.

In order to perform shading correction we must estimate the constant coefficient terms $a_0(i)$ and $a_1(i)$. This can be done by having two known values of $f(i)$ as inputs. For this work, the values of $f(i) = 0$ and $f(i) = 255$ were chosen. These values represent the minimum and maximum gray level values in the images. Note that there are sensors that may be chosen that have different dynamic ranges. The input $f(i) = 0$ is obtained for all values of i by imaging a uniform black surface while the input $f(i) = 255$ is obtained by imaging a uniform white surface. The uniform white surface represents the brightest possible surface that can be imaged by a camera and each pixel in the image of such a surface is expected to have a value of 255 when sufficient illumination is provided (assuming also that the image is not saturated). Any nonuniformity in the output response of the camera can be seen as a deviation from the value of 255. For $f(i) = 0$ equation (4.1) gives

$$a_0(i) = g(i)$$

where g(i) is the measured output for the given uniformly black input and which we will denote as $g_b(i)$. For a value of $f(i) = 255$, and dropping the higher order terms of equation (4.1) we get:

$$a_1(i) = \frac{g(i) - a_0(i)}{255}$$

where $g(i)$ is the measured output for the given uniformly white input and which we will denote as $g_w(i)$. Therefore, $\hat{f}(i)$ is now given by:

$$\hat{f}(i) = \frac{g(i) - g_b(i)}{g_w(i) - g_b(i)} \cdot 255$$

A scanned image is represented by $g(i)$, a scanned image of a uniform black surface is represented by $g_b(i)$, a scanned image of a uniform white surface is represented by $g_w(i)$, and the shade corrected image is represented by $\hat{f}(i)$.

For color images, the uniform white surface must have a flat spectral response across the 400nm to 700 nm range (visible range) of the electromagnetic spectrum.

The reason for this is that, when shade correcting color images, each color channel needs to be corrected independently.

6.4 System Integration

At this point, all the components were now ready for integration into a full implementation of the machine vision system. This is the next to the last stage depicted in Figure 3.1. A diagram of the complete system is shown in Figure 6.27.

Each of the modules in the diagram was described in the previous sections. For this implementation, the **Imaging Subsystem** basically consists of hardware components, while the rest of the system consists of software components. However, since much work on the hardwood lumber inspection problem remains to be done, this implementation represents just one iteration of the rapid-prototyping procedure illustrated in Figure 3.1. More iterations may be needed before an acceptable solution is reached. Apart from future research needed on the lumber inspection problem, there is also the need for a real-time implementation. Further prototyping in this area would consist of implementing some of the components in hardware. For example, the **BACKGROUND** module, the **SCATTERGRAM** module, and the **LOOKUP_TABLE** modules, could easily be implemented in hardware. This is where the issue of hardware/software tradefoffs become important and is all part of the nature of rapid-prototyping.

In this stage of the design, the system was tested on 4 feet long boards (Figures 6.28 and 6.30) to determine if the algorithms still worked. The only significant problem encountered in this stage was the need to increase array sizes in the software. No change to the algorithms were needed. Figures 6.29 and 6.31 show the

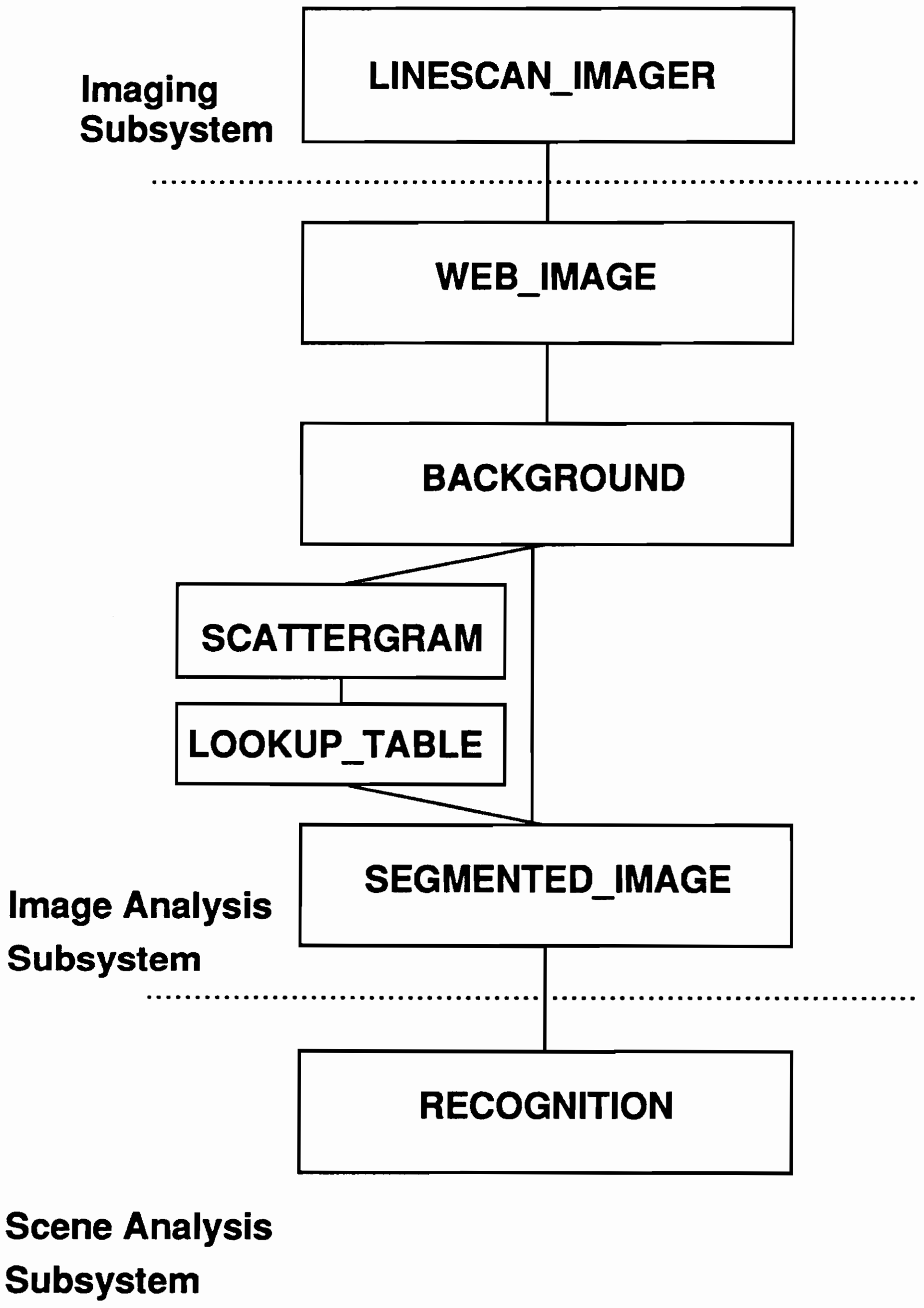

Figure 6.27. Components of the machine vision system for inspecting hardwood lumber.

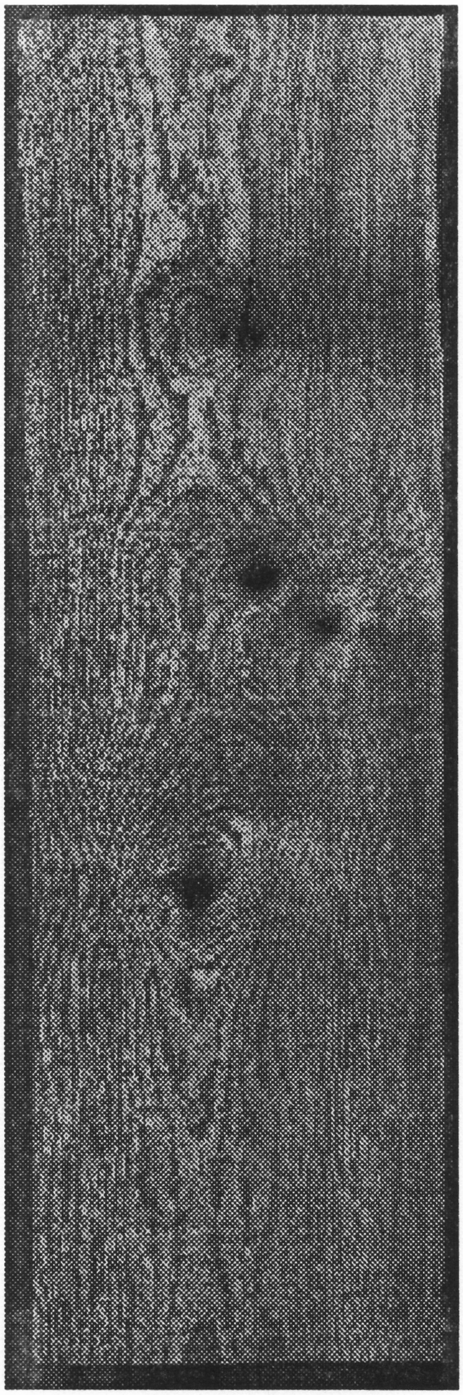

Figure 6.28. Image of a four feet long oak board.

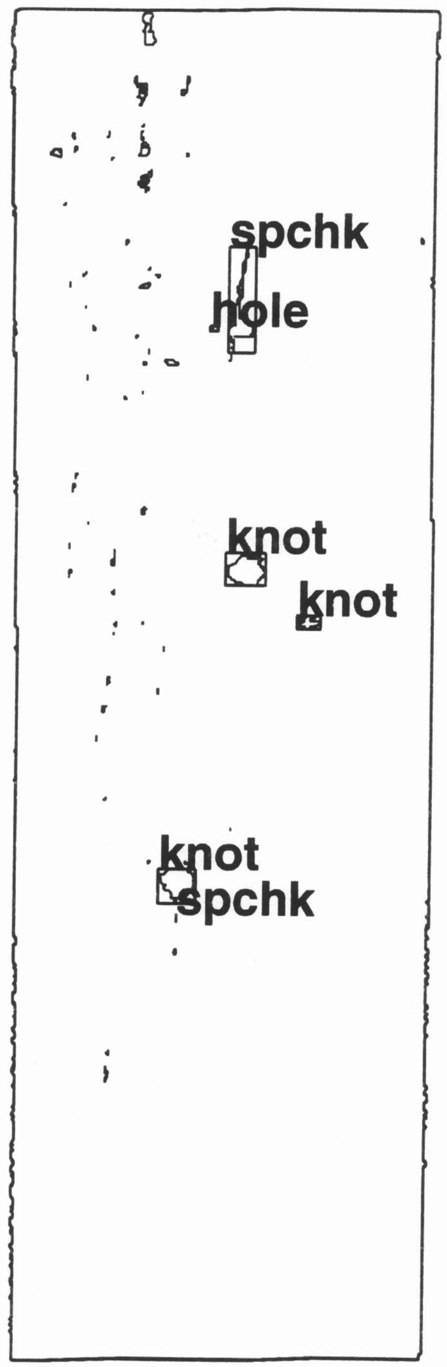

Figure 6.29. Result of applying the machine vision system to the image of Figure 6.28.

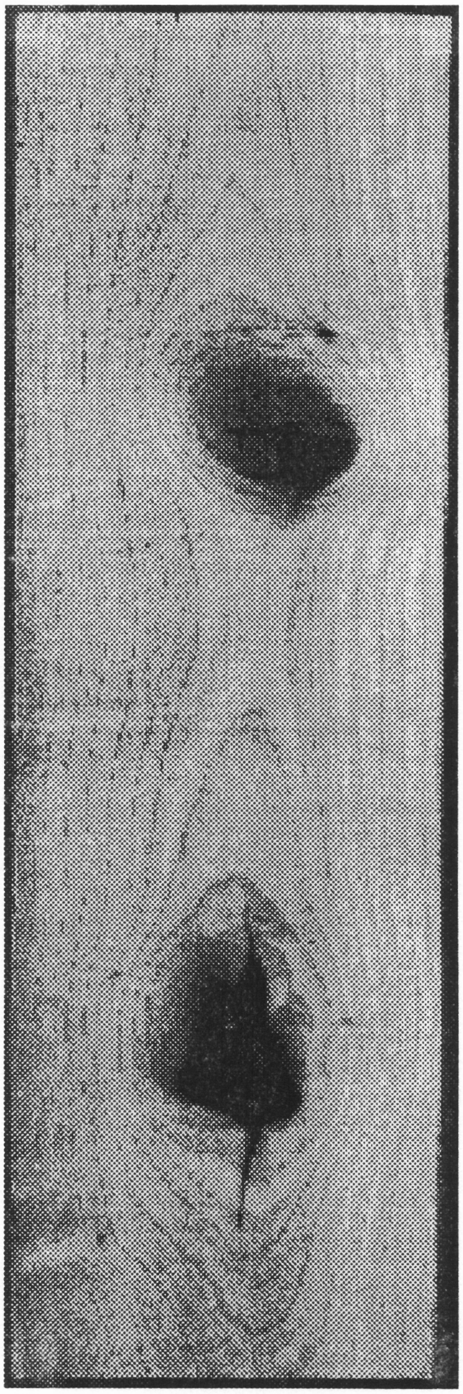

Figure 6.30. Image of a four feet long yellow poplar board.

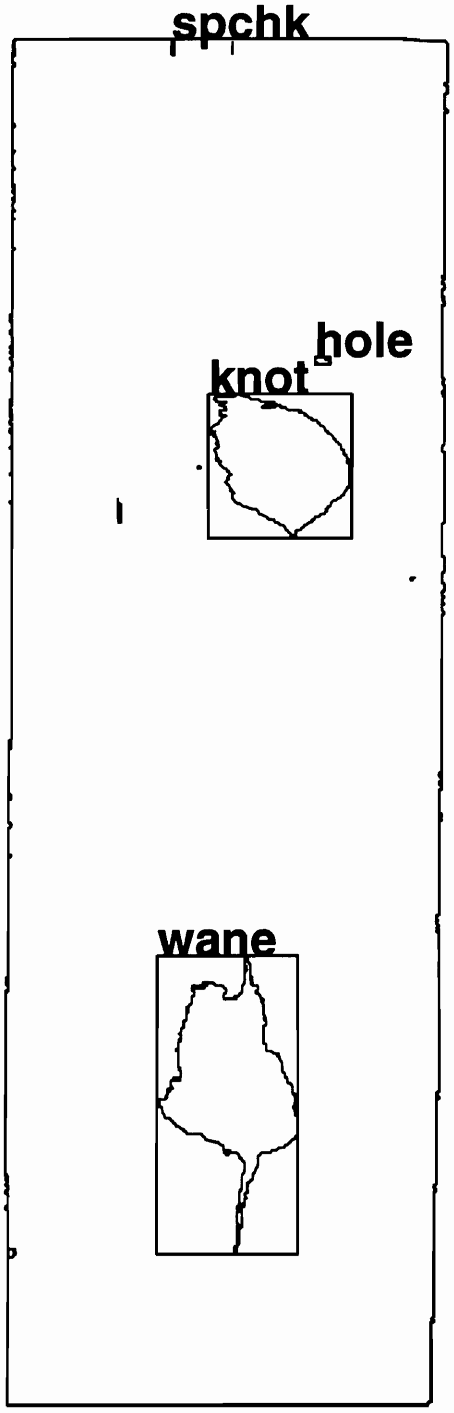

Figure 6.31. Result of applying the machine vision system to the image of Figure 6.30.

processing results on an oak board (Figure 6.28) and a yellow poplar board (Figure 6.30). Included in Figures 6.29 and 6.31 are the processing results of the **Image Analysis Subsystem**. This can be seen by the un-labeled outlines of small regions and is especially obvious in Figure 6.29. One of the functions of the recognition module is to perform an initial check to see if a feature is actually clearwood before attempting to place it in a particular defect class. The un-labeled outlines have been left to demonstrate the re-labeling of features back to their correct feature class, that of clearwood, since the recognition module will not explicitly label a feature as clearwood.

6.5 Summary

To demonstrate the utility of the machine vision prototyper, a machine vision system for the inspection of hardwood lumber was derived and has been described in this chapter. The design process followed the rapid-prototyping procedure depicted in Figure 3.1. The effort showed that partial implementations of the machine vision system could be incrementally developed to a final prototype, thus reducing the complexity of the design effort.

A result of this effort was the creation of a partial solution to the surfaced hardwood lumber inspection problem. While the feasibility of a complete solution has not been shown, much work towards establishing this feasibility has already been accomplished. As a result of this effort a number seemingly general purpose reusable modules have been developed for planar web inspection. The imaging subsystem for example, could be used to image other types of planar webs. Furthermore, components of the image analysis subsystem used methodologies that are generally applicable to images that can be segmented based on color and is thus relatively independent of the hardwood lumber inspection problem. With a minimum of modifications, these components could conceivably be used on other planar web inspection problems.

The machine vision system described in this chapter represents one possible implementation in an ongoing effort to design a hardwood lumber inspection system. This is typical of evolutionary rapid-prototyping. The next step is to examine the system to see if each module meets the requirements or to see if there are any new requirements that were not considered before. Then the system should be put through another rapid-prototyping iteration.

Chapter 7

ATTACKING ANOTHER PLANAR WEB INSPECTION PROBLEM

The final objective of this dissertation is to indicate how the machine vision prototyper can be used to attack a second planar web inspection problem. The goal is to further demonstrate the utility of the prototyper in a rapid-prototyping approach to machine vision system design. To accomplish this final objective, the conceptual design of a structured light based machine vision system will be described. This type of machine vision system is different from the one described in the previous chapter as it requires a different set of sensors and algorithms. Due to the time limitation, design of this system could not be realized by the author. However, such a system is currently being developed by Hou (1993). This chapter will be restricted to a description of how the prototyper could be used in its design.

7.1 Structured Light Based Machine Vision

The machine vision system methodology described in the previous section was basically designed to attack the problem of locating and identifying surface defects in planar webs. However, these methods are not suitable for the second type of planar web inspection problem defined in Chapter 1. In this type of problem, defects manifest themselves by causing a variation in the cross section that a web presents in a plane P as the web travels through P along a line perpendicular to the plane P. These defects alter the three-dimensional shape characteristics of the longitudinal planar web. To attack this type of problem structured light based machine vision methods are proposed.

In a structured light based machine vision system, illumination of an object is carried out with a special pattern. The three-dimensional shape of an object can then be extracted based on the resulting pattern as viewed by a camera. The types of patterns used are usually composed of parallel light stripes (Ballard (1982)) or with a grid pattern (Dunn (1989)). There are several variations in the types of patterns used. For example, Boyer and Kak (1987) uses a color coding scheme that projects a single pattern of red, green, blue, and white vertical stripes onto the scene. In another example, LeMoigne and Waxman (1984) use square shaped "dots" in a grid pattern to guide grid labeling. To extract three-dimensional information from an image, the deviation of these light patterns from the normal form of these patterns on a flat surface, are studied.

7.2 A Structured Light Based Machine Vision System

Based on the need for a structured light based machine vision system the utility of machine vision prototyper is conceptually demonstrated on the second type of planar web inspection problem. The application problem that is considered is the inspection of hardwood parts coming out of a molder. The molding operation introduces machine defects in otherwise acceptable parts. The defects caused by the molder typically involve the lack of the desired smoothness on the surface of the machined part or in a deviation from a desired three-dimensional shape. A structured light based machine vision system is proposed for attacking this problem.

To demonstrate the utility of the prototyper, the problem is considered under two possible scenarios. The scenarios represents the two possible extremes in the maturity of the prototyper. In the first case, the prototyper has not been utilized to attack this second type of planar web inspection. As such, most of the components will have to be designed from ground up. In the second case, the prototyper has already been utilized on a similar problem. In such a case, the prototyper will have in its database of reusable components containing the necessary parts to configure into a machine vision system capable of attacking the problem.

At the conclusion of the research by this author the prototyper had been applied only to the detection of surface defects in a planar web inspection problem. For the second problem being considered in this chapter, a structured light based machine vision system is currently being implemented on the machine vision prototyper. The hardware components are being designed by Hou (1993). A possible configuration for utilizing these components is shown in Figure 7.1. In this system, cross board laser light lines are generated by bouncing a laser beam of a rotating polygon to produce uniform scan lines. Mirrors are mounted on the sides of the rotating polygon to deflect the laser light. This object is labeled the "scanning mirror" in Figure 7.1. A linear array camera images the scene to produce a 128X128 image. Specialized hardware is then utilized to detect the pixel position of each point on a scan line. The positions of these points on a scan line can then be used to construct a range image. A range image is an image where the intensity of each pixel point represent points in a three dimensional space.

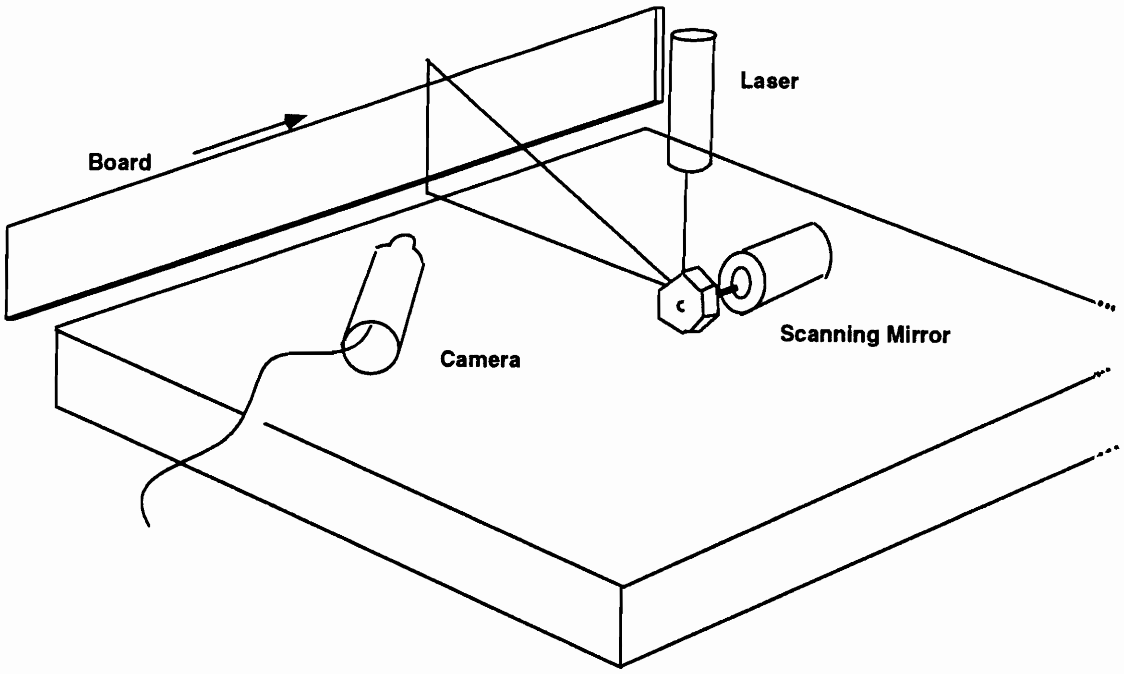

Figure 7.1. A structured-light based imaging system.

To design the machine vision system, the machine vision prototyper can be utilized in the following manner:

(1) The prototyper can be used to mount the hardware components, i.e., the laser, the scanning mirror, and the camera. The best imaging geometry can easily be determined by simply changing the positions of these components. The experimentation with imaging geometry can easily be performed on the prototyper. Once the computer interfaces to the imaging system have been designed, the imaging system can be treated as a software object and software rapid-prototyping can begin.

(2) The prototyper can be used in the rapid development of software components for image acquisition and for analysis of the resulting range images. This is facilitated by the availability of a variety of experimental tools such as the image display utility, and by the availability of system level software providing file access, parameter specification, and object-oriented development support.

The rapid-prototyping strategy to be used is basically that shown in Figure 3.1 and is the same as that used in designing the machine vision system described in the previous chapter.

A possible configuration of the machine vision system is depicted in Figure 7.2. The objects corresponding to objects of the same names in the machine vision system described in Chapter 6 indicate that these objects can be extended to include range image data from a structured-light imaging system. The object **LSTRUC_IMAGER** corresponds to the structured-light imaging system in Figure 7.1. The object **WEB_RIMAGE** would correspond to a range image, with an object-manipulator that finds the boundary of a hardwood part coming out of a molder,

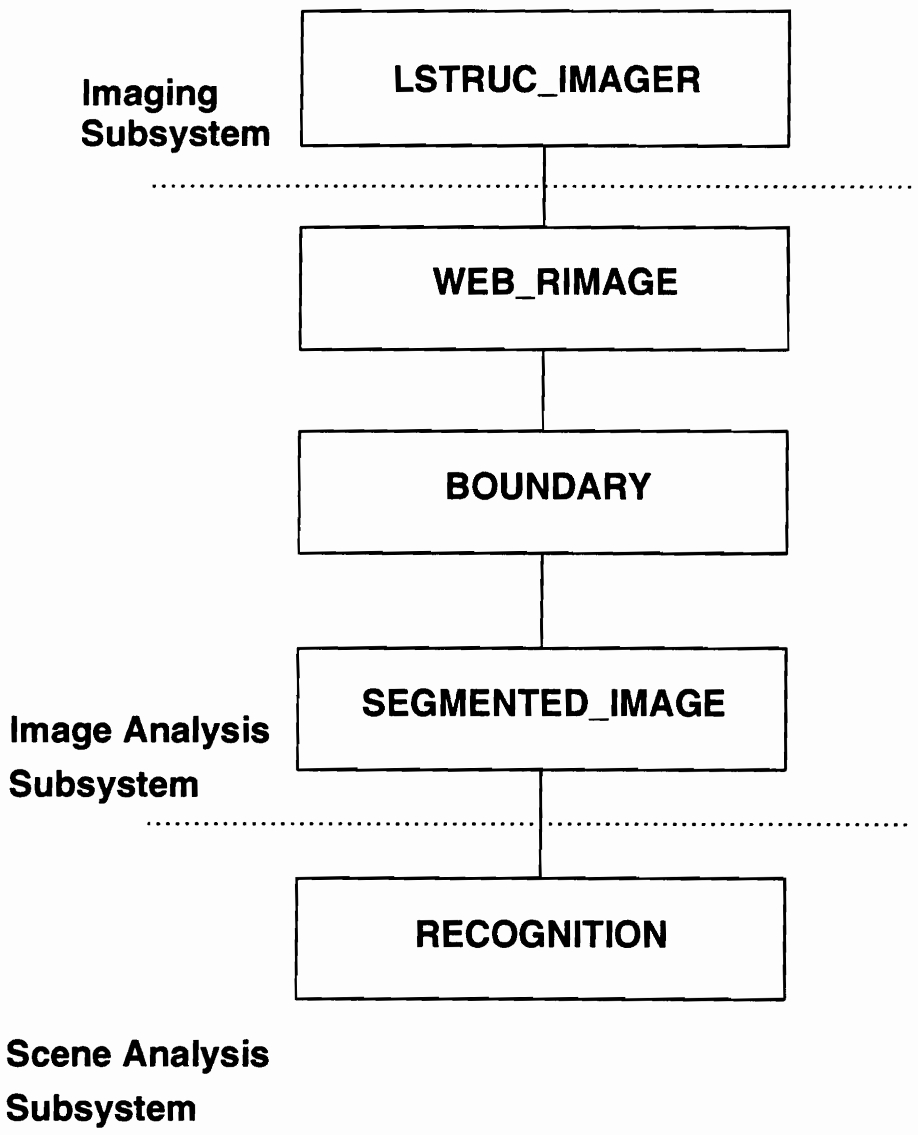

Figure 7.2. Possible components of a structured-light based machine vision system.

to create the **BOUNDARY** object. The **SEGMENTED_IMAGE** object must now include an object-manipulator to segment regions in a range image corresponding to defects.

For the second case, i.e., the case in which the prototyper already has existing components for attacking the problem of detecting defects in the three-dimensional shape characteristics of planar webs, the prototyper is utilized a trivial manner. In this case, existing components will be configured into a first partial implementation. Each partial implemented is tested until a final implementation that meets all requirements has been designed.

7.3 Summary

In this chapter, the utility of the machine vision prototyper in designing a second machine vision system was conceptually demonstrated. This machine vision system is a structured light based machine vision system capable of detecting defects in planar webs that are manifested as an alteration of the expected three-dimensional characteristics of the web. The ease of designing such a new system can be found in the ability to easily mount new components, the ability to experiment with imaging geometry, and the ability to use previously designed components.

With the design of the second machine vision system, new techniques for defect detection will be available. The second machine vision system can then be integrated with the first machine vision system to provide a system that can be utilized to attack a wider range of planar web inspection problems. The machine vision prototyer provides a platform for such a system integration.

Chapter 8

CONCLUSIONS

Most current machine vision systems for industrial inspection have been developed to accomplish specific tasks and are not easily adapted to other applications. They are also not easily adapted to changes in requirements associated with the inspection of a particular product. Often, the complete set of these requirements are not known in advance. These machine vision systems are typically implemented in special purpose hardware that perform specific operations. This lack of flexibility is mainly due to the real-time requirements of most industrial inspection tasks.

Inexpensive microcomputers with the computational capability to satisfy the necessary real-time requirements will, for the first time, allow "general purpose" machines to attack a variety of industrial inspection problems. This should spur the development of machine vision systems for industrial inspection, perhaps allowing machine vision technology to finally play an important role in manufacturing. In order to assume this increased role, new ways of designing machine vision systems for industrial inspection tasks must be found. New systems must be able to use components from previously designed systems. Ways must be found to consider broad classes of applications as opposed to designing machine vision systems for specific problems. These systems must be "flexible" in order to accommodate rapid changes in requirements for each inspection tasks. The current lack of these capabilities has limited the role of machine vision systems in industrial inspection.

The ability to experiment with different solutions and to rapidly adapt to changes in requirements for specific problems must be considered.

In this dissertation, a machine vision prototyper has been developed as an experimental tool to aid in rapidly prototyping machine vision systems for the planar web inspection problem. The ability to use "general purpose" computers will permit the rapid implementation of these machine vision systems. The objective of the prototyper is to provide the experimental tools needed for the design of such systems.

The experimental tools provided by the machine vision prototyper for rapid-prototyping machine vision systems for the planar web inspection problem has been described in Chapters 3 and 4. These tools are summarized as follows.

(1) An optical bench for mounting imaging components. This optical bench, together with a set of rods, and rod mounts enable the imaging geometry of the imaging components to be experimented with.

(2) A materials handling system for transporting planar webs.

(3) An object detection system for detecting the presence of a planar web.

(4) An object-oriented environment that enables hardware and software components to be treated in the same manner. In such an environment, different components (objects) can be selected and assembled into a machine vision system in a uniform way. Such an environment facilitates the rapid-prototyping of machine vision systems.

(5) Display objects such as an image display, a HP plotter, and a laser-printer, are available to view or plot images and the results of operations on these

images. Experimentation is faciliated by being able to test an algorithm and to display the results of the test in a form suitable for interpretation.

To facilitate rapid-prototyping a set of reusable components have been developed. These include

(6) a set of object-manipulators that control the hardware objects for imaging of planar webs,

(7) and a set of object-manipulators capable of performing computer vision related tasks.

Finally,

(8) a set of image analysis object-manipulators are provided to perform initial analyses on images. In order to define features that can be extracted from an image, one has to understand the nature of the data. For example, if one wishes to study the color differences between knots and clear wood, one could perhaps compute a two dimensional histogram to see how they relate to each other in two color bands of an image. Other image analysis object-manipulators such as edge-operators and the Fourier transform, have also been implemented for image objects.

To demonstrate the utility of the prototyper, a machine vision system for the inspection hardwood lumber has been developed. The hardwood lumber inspection problem is a complex problem that is representative of a wide variety of planar web inspection problems. In designing this system, a number of reusable hardware and software components have been designed that could be applicable to a number of other planar web inspection problems. The development of this machine vision system has also resulted in a simple but effective segmentation methodology capable of faciltating a real-time implementation. Note that most previous work in the area of lumber inspection has been based on computationally complex algorithms. Furthermore, these systems typically used supervised classification procedures that could not be easily adapted to the wide variability in wood color.

Finally, the utility of the machine vision prototyper was also demonstrated on another planar web inspection problem. The application problem that was considered is the inspection of hardwood parts coming out of a molder requiring the use of a structured light based machine vision system.

It is hoped that this work will provide the starting point on which other researchers in machine vision can build upon. Recommendations for further research include the following:

- *Apply the machine vision system prototyper to other web inspection problems.* Though the planar web inspection problem has been chosen as the problem domain of the machine vision prototyper, many of the resulting experimental tools can be applied to other web inspection problems. This would provide further insight into the flexible methodologies needed for developing "general purpose" computer-based machine vision systems.

- *Acquire other types of sensors.* This would provide the capability to test other imaging methods and to expand the variety of planar web inspection problems that can be attacked. Currently, a machine vision system based on structured light is being developed for detecting holes and wane and other such defects in hardwood lumber. The imaging hardware for the system will be mounted on the prototyper when its development is complete. Finally, for the third type of planar web inspection problem, i.e., for the detection of internal defects, the acquisition of an x-ray imager is being planned.

- *Implement the Machine Vision Prototyper on a more specialized system.* Such a specialized machine vision prototyper would be capable of being moved to and tested in, an industrial environment. Work is currently in progress to implement the system on an IBM Risc 6000 computer working in tandem with a 37 feet long materials handling system that can transport lumber boards up to 16 feet in length at speeds of up to 2 linear feet per second and greater. This system is being targeted for use in actual lumber processing plants. Figure 8.1 shows a picture of the materials handling system.

- *Refine the segmentation component.* Thresholding of the two-dimensional histogram to separate clusters representing defects and clearwood regions works well if these clusters are well defined and do not overlap. Problems occur when clusters representing small regions are close to larger regions. These clusters are either "captured" by the larger regions or they become undefined.

- *Refine the recognition component.* Most of the effort in developing the machine vision system in this dissertation has been aimed at image segmentation, due to the difficulty of the problem. As such, much work is needed to refine the recognition components. In fact, work has been done in this area and is reported by Cho (1991).

The work described in this dissertation has served as a usefull starting point for continuing research. In fact, a machine vision system for the inspection of rough lumber has been designed based on the initial work done in this dissertation (Cho (1991)). Furthermore, at the time of this writing, all the recommendations for

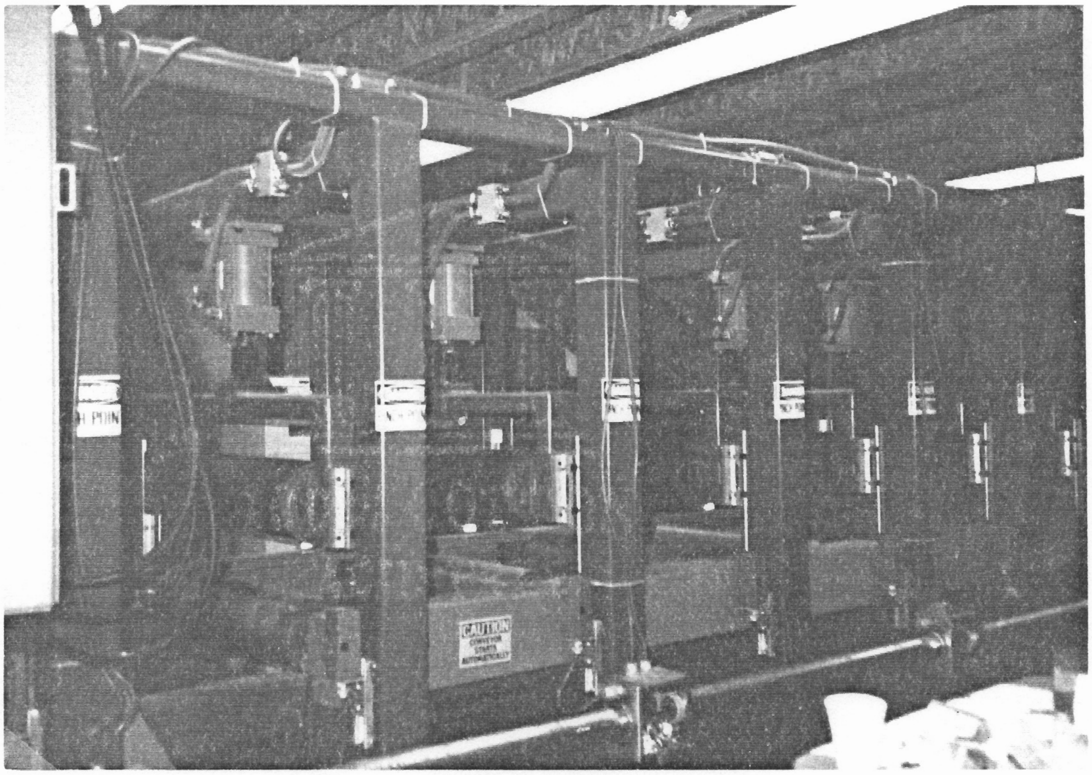

Figure 8.1. A materials handling system for handling up to 16 feet long lumber boards.

further work is either currently being performed, or is being planned. It is hoped that the prototyper will continue to serve as a much needed experimental vehicle for designing machine vision systems.

REFERENCES

Alpert, S.R., S.W. Woyak, H.J. Shrobe, and L.F. Arrowood, "Object-Oriented Programming in AI," *IEEE Expert*, pp. 6-7, December 1990.

Andriole, S.J., *Storyboard Prototyping: A New Approach to User Requirments Analysis*, QED Information Sciences, Inc., 1989.

Atwood, T.M., "The case for object-oriented databases," *IEEE Spectrum*, pp. 44-47, February 1991.

Ballard, D. H., and C. M. Brown, *Computer Vision*, Prentice-Hall, Inc., 1982.

Bartlett, S.L., P.J. Besl, C.L. Cole, R. Jain, D. Mukherjee, and K.D. Skifstad, "Automatic Solder Joint Inspection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 1, pp. 31-43, January 1988.

Benhabib, B., C.R. Charette, K.C. Smith, and A.M. Yip, "Automatic Visual Inspection of Printed Circuit Boards: An Experimental System," *International Journal of Robotics and Automation*, Vol. 5, No. 2, pp. 49-58, 1990.

Beverley, A.M., and P.G. Penton, eds., *ELAS Science and Technology Laboratory Applications Software: Volume I, Programmer Reference*, National Aeronautics and Space Administration, John C. Stennis Space Center, Science and Technology Laboratory, May 1989.

Binford, T.O., "Survey of Model-Based Image Analysis Systems," *International Journal of Robotics Research*, vol. 1, no. 3, pp.57-82, 1982.

Blair, G.S., J. Malik, J.R. Nicol and J. Walpole, "A sysnthesis of object-oriented and functional ideas in the design of a distributed software engineering environment," *Software Engineering Journal*, Vol. 5, No. 3, pp. 193-204, May 1990.

Boehm, B.W., T.E. Gray, and T. Seewaldt, "Prototyping Versus Specifying: A Multiproject Experiment," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 3, pp. 290-302, May 1984.

Booch, G., *Software Engineering With ADA*, The Benjamin/Cummings Publishing Company, Inc., 1983.

Booch, G., "Object-oriented development," *IEEE Transactions on Software Engineering*, Vol. SE-12, no. 2, pp. 211-221, February 1986.

Boyer, K.L., and A.C. Kak, "Color-encoded Structured Light for Rapid Range Sensing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, no. 1, pp. 14-28, 1987.

Brice, C., and C. Fennema, "Scene Analysis Using Regions," *Artificial Intelligence*, vol. 1, no. 3, pp. 205-226, 1970.

Brooks, R.A., "Symbolic Reasoning among 3 Dimensional Models and 2 Dimensional Images," *Artificial Intelligence Journal*, vol. 17, pp. 285-348, 1981.

Brooks, R.A., "Model Based Three Dimensional Interpretation of Two Dimensional Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, pp. 140-150, March 1983.

Campbell, W., "A C interpreter for Scheme - an exercise in object-oriented design," *Software Engineering Journal*, Vol. 6, No. 4, pp. 130-136, July 1991.

Cappellini, V., A. Del Bimbo, and P. Nesi., "Integrating Object Oriented Programming Paradigm Concepts in Designing a Vision and Pattern Recognition System Architecture," *Proceedings of IAPR*, 10th International Conference on Pattern Recognition, Vol. 2, pp. 572-575, June 1990.

Chan, J.P., and B.G. Batchelor, "Integrating vision and AI in an image processing workstation," *Proceedings of SPIE*, Machine Vision Systems Integration in Industry, Vol. 1386, pp. 163-170, November 8-9, 1990a.

Chan, J.P., B.G. Batchelor, I.P. Harris, and S.J. Perry, "Intelligent Visual Inspection Of Food Products," *Proceedings of SPIE*, Machine Vision Systems Integration in Industry, Vol. 1386, pp. 171-179, November 8-9, 1990b.

Chien, R.T., W.E. Snyder, "Hardware for Visual Image Processing," *IEEE Transactions on Circuits and Systems*, Vol. CAS-22, No. 6, pp. 541-551, June 1975.

Chin, R.T., "Algorithms and Techniques for Automated Visual Inspection," *Handbook of Pattern Recognition and Image Processing* (T.Y. Young and K.S. Fu, eds.), pp. 587-612, Academic Press, Orlando, Florida, 1986.

Chin, R. T., "SURVEY, Automated Visual Inspection: 1981 to 1987," *Computer Vision, Graphics, and Image Processing*, Vol. 41, pp. 346-381, March 1988.

Chin, R. T., and C. A. Harlow, "Automated Visual Inspection: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 6, pp. 557-573, November 1982.

Cho, T. H., "A Knowledge-Based Machine Vision System for Automated Industrial Web Inspection," Ph. D. Dissertation, Bradley Department of Electrical Engineering, Virginia Tech, 1991.

Cieslak, R., A. Fawaz, S. Sachs, P. varaiya, and J. Walrand, "The Programmable Network Prototyping System," "A Rapid Prototyping Facility for Fligh Research in Advanced Systems Concepts," *Computer*, pp. 61-66, May 1989.

Coad, P., "Object-Oriented Analysis," in *System and Software Requirements Engineering* (R. H. Thayer and M. Dorfman, eds.), pp. 272-289, IEEE Computer Society Press, Los Alamitos, California, 1990.

Conners, R. W., C. W. McMillin, K. Lin, and R. E. Vasquez-Espinosa, "Identifying and Locating Surface Defects in Wood: Part of an Automated Lumber Processing System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 6, pp. 573-583, November 1983.

Conners, R. W., C. W. McMillin, and C. T. Ng, "Utility of color information in the location and identification of defects in surfaced hardwood lumber," *First Int. Conf. Scanning Technology in Sawmilling*, pp. 18.1-18.33, 1985.

Conners, R. W., C. T. Ng, T. H. Cho, and C. W. McMillin, "Computer Vision System for Locating and Identifying Defects in Hardwood Lumber," *Proceedings of SPIE*, Applications of Artificial Intelligence VII, Vol. 1095, pp. 48-63, March 28-30, 1989a.

Conners, R. W., and C. T. Ng, "Developing a Quantitative Model of Human Preattentive Vision," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, No. 6, pp 1384-1407, November/December 1989b.

Cook, S., "Languages and object-oriented programming," *Software Engineering Journal*, Vol. 1, No. 2, pp. 73-80, March 1986.

Darwish, A.M., and A.K. Jain, "A Rule Based Approach for Visual Pattern Inspection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 1, pp. 56-68, January 1988.

Destor, C., A. Scortesse, and J. De Man, "Object-Oriented Techniques for Switching Systems," *Electrical Communication*, Vol. 64, No. 4, pp. 37-45, 1990.

Dickinson, S. J., and L. S. Davis, "An Expert Vision for Autonomous Land Vehicle Road Following," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 826-831, 1988.

Drayer, T.H., *Personal Communication*, 1990.

Drayer, T.H., "A High Performance Micro Channel Interface for Image Processing Applications," Master's Thesis, Bradley Department of Electrical Engineering, Virginia Tech, 1991.

Duff, B.L., and C.K. Carlson, "Applications of Object-Oriented Approaches to Expert Systems in the Earth Sciences," *Proceedings of IEEE/ACM*, International Conference on Developing and Managing Expert System Programs, pp. 160-165, Sept.30 - Oct. 2, 1991.

Duke, E.L., R.W. Brumbaugh, and J.D. Disbrow, "A Rapid Prototyping Facility for Flight Research in Advanced Systems Concepts," *Computer*, pp. 61-66, May 1989.

Dunn, S.M., R.L. Keizer, and J. Yu, "Measuring the Area and Volume of the Human Body with Structured Light," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, No. 6, pp 1350-1364, November/December 1989.

Flickner, M., M. Lavin, and S. Das, "An Object-oriented Language for Image and Vision Execution (OLIVE)," *Proceedings of IAPR*, 10th International Conference on Pattern Recognition, Vol. 2, pp. 561-571, June 1990.

Galbiati, Jr., L.J., *Machine Vision and Digital Image Processing Fundamentals*, NJ: Prentice-Hall, Inc., 1990.

Garland, E., and R.W. Ehrich, "A Gipsy Primer," *Technical Report*, Spatial Data Analysis Laboratory, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, October 29, 1986.

Giebel, H., H. Gutschale, and F. Wahl, "A system for automatic inspection of glass bottles using texture analysis procedures," *Proceedings, 6th Int. Conf. Pattern Recognition*, pp. 1068-1070, October 1982.

Goldberg, M., and S. Shlien, "A Clustering Scheme for Multispectral Images," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-8, No. 2, pp 86-92, February 1978.

Gray, P.D., A.C. Kilgour, and C.A. Wood, "Dynamic reconfigurability for fast prototyping of user interfaces," *Software Engineering Journal*, Vol. 3, No. 6, pp. 257-262, November 1988.

Gupta, R., W.H. Cheng, R. Gupta, I. Hardonag, and M.A. Breuer, "An Object-Oriented VLSI CAD Framework" *Computer*, pp. 28-37, May 1989.

Haralick, R. M., "Statistical and Structural Approaches to Texture," *Proceedings of IEEE*, Vol. 67, no. 5, pp. 786-804, 1979.

Haralick, R. M., and L. G. Shapiro, "SURVEY, Image Segmentation Techniques," *Computer Vision, Graphics, and Image Processing*, Vol. 29, pp. 100-132, 1985.

Haralick, R. M., and L. G. Shapiro, *Computer and Robot Vision: Volume 1*, Addison-Wesley Publishing Company, Inc., 1992.

Hashim, A.A., M. Lefley, and D. Spencer, "Automated visual inspection and classification of textile materials and garments," *Proceedings, Annual World Conf. Computers in the World of Textiles*, pp. 167-175, September 1984.

Hou, Y., "Developing a Flexible Range Sensing System for Industrial Applications," Master's Thesis, Bradley Department of Electrical Engineering, Virginia Tech, in preparation, 1993.

Hwang, V.S., L.S. Davis, and T. Matsuyama, "Hypothesis Integration in Image Understanding Systems," *Computer Vision, Graphics, and Image Processing*, Vol. 36, pp. 321-371, 1986.

Jacky, J.P., and I.J. Kalet, "An object-oriented programming discipline for standard Pascal," *Commun. ACM*, Vol. 30, No. 9, pp. 772-776, 1987.

Jacobson, J., "Industrial development of software with an object-oriented technique," *Journal of Object-Oriented Programming*, vol. 4, no. 1, pp. 30-41, March/April 1991.

Kent, W.K., and M.O. Shneier, "Eyes for automatons," *IEEE Spectrum*, pp. 37-45, March 1986.

Koivo, A. J., and C. W. Kim, "Robust Image Modeling for Classification of Surface Defects on Wood Boards," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 6, pp. 1659-1666, November/December 1989.

Lakatosh, B., "Defectoscopy of Wood," (Translated from Russian and available through NTIS).

Lavin, M.A., "Industrial Machine Vision: Where Are We? What Do We Need? How Do We Get It?," *Machine Vision: Algorithms, Architectures and Systems* (H. Freeman, ed.), pp. 161-185, Academic Press, Inc., San Diego, California, 1988.

Lamb, F., *Personal Communication*, 1990.

Lawrence, P.D., and K. Mauch, *Real-Time Microcomputer System Design: An Introduction* McGraw-Hill, Inc., 1987.

Lawson, H.W., "Philosophies for Engineering Computer-Based Systems," *Computer*, pp. 52-63, December 1990.

Lawton, D.T., and D.M. Mead, "A Modular Object Oriented Image Understanding Environment," *Proceedings of IAPR*, 10th International Conference on Pattern Recognition, Vol. 2, pp. 611-616, June 1990.

LeMoigne, J., and A.M. Waxman, "Projected Light Grids For Short Range Navigation of Autonomous Robots," *Proceedings of the 7th International Conference on Pattern Recognition*, pp. 203-206, 1984.

Leung, K.S., and M.H. Wong, "An Expert-System Shell Using Structured Knowledge," *Computer*, pp. 38-47, March 1990.

Levine, M.D., and A.M. Nazif, "Rule-Based Image Segmentation: A Dynamic Control Strategy Approach," *Computer Vision, Graphics, and Image Processing*, Vol. 32, pp. 104-126, 1985.

Lumia, R., L. G. Shapiro, and O. Zuniga, "A New Connected Components Algorithm for Virtual Memory Computers," *Computer Vision, Graphics, and Image Processing*, Vol. 22, pp. 287-300, 1983.

Luqi, "Software Evolution Through Rapid Prototyping," *Computer*, pp. 13-25, May 1989.

Luqi, and V. Berzins, "Rapidly Prototyping Real-Time Systems," *IEEE Software*, pp. 25-36, September 1988a.

Luqi, V. Berzins, and R. Yeh, "A Prototyping Language for Real-Time Software," *IEEE Transactions on Software Engineering*, October 1988b.

Luqi, and M. Ketabchi, "A Computer-Aided Prototyping System," *IEEE Software*, pp. 66-72, March 1988c.

Meisels, A., and S. Bergman, "Finding Objects on Aerial Photographs: A Rule-Based Low Level System," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 118-122, 1988.

Micallef, J., "Encapsulation, Reusability and Extensibility in Object-Oriented Programming Languages," *Journal of Object-Oriented Programming*, Vol. 1, No. 1, pp. 12-36, April/May 1988.

Mohan, L., and R.L. Kashyap, "An Object-Oriented Knowledge Representation for Spatial Information," *IEEE Transactions on Software Engineering*, Vol. 14, No. 5, pp. 675-681, May 1988.

Mullin, M., *Rapid Prototyping For Object-Oriented Systems*, Addison-Wesley Publishing Company, Inc., 1990.

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

Nazif, A.M., and M.D. Levine, "Low Level Image Segmentation: An Expert System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 5, pp. 555-577, Sept. 1984.

Nevatia, R., "Image Segmentation," in *Handbook of Pattern Recognition and Image Processing* (T.Y. Young and K.S. Fu, eds.), pp. 215-231, Academic Press, Orlando, Florida, 1986.

Nielsen, J., and J.T. Richards, "The Experience of Learning and Using Smalltalk," *IEEE Software*, pp. 73-77, May 1989.

Niemann, H., H. Bunke, I. Hofmann, G. Sagerer, F. Wolf, and H. Feistel, "A Knowledge Based System for Analysis of Gated Blood Pool Studies," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 3, pp. 246-259, May 1985.

Novini, A.R., "The Lighting and Optics Expert System For Machine Vision," *Proceedings of SPIE*, Machine Vision Systems Integration in Industry, Vol. 1386, pp. 2-9, November 8-9, 1990.

Okawa, Y., "Automatic Inspection of Surface Defects of Cast Metals," *Computer Vision, Graphics, and Image Processing*, Vol. 25, pp. 89-112, 1984.

Paul, D., W. Hattich, W.Nill, S. Tatari, and G. Winkler, "VISTA: Visual Interpretation System for Technical Applications - Architecture and Use," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 3, pp. 399-407, May 1988.

Pavlidis, T., *Structural Pattern Recognition*, Springer-Verlag, Berlin, 1977.

Perceptics, Corp., *RIPS Host Support User's Manual for VAX/VMS*, Perceptics Corporation, Knoxville, Tennessee, 1986.

Price, W.H., "The Photographic Lens," *Scientific American*, pp. 74-83, August 1976.

Purll, D.J., "Automated surface inspection with solid-state image sensors," *Proceedings of SPIE*, Vol. 145, pp. 18-25, March 1978.

Purll, D.J., "Solid-State Image Sensors," in *Automated Visual Inspection*, (B.G. Batchelor, D.A. Hill, and D.C. Hodgson eds.), pp. 255-293, Elsevier Science Publishing Company, Inc., New York, New York, 1985.

Rao, A.R., and R. Jain, "Knowledge Representation and Control in Computer Vision Systems," *IEEE EXPERT*, pp. 64-79, Spring 1988.

Roberts, L., "Machine Perception of Three-dimensional Solids," *Optical and Electro-Optical Information Processing* (J. Tippet et al., eds.), Cambridge MA: M.I.T. Press, 1965.

Rehkugler, G.E., and D.J. Aneshansley, "Vision Systems in Food Processing," *SENSORS: The Journal of Machine Perception*, Vol. 3, No. 6, pp. 6-13, June 1986.

Roman, G.C., M.J. Stucki, W.E. Ball, and W.D. Gillett, "A Total System Design Framework," *Computer*, pp. 15-26, May 1984.

Rosenthal, S., and L. Stahlberg, "Integrated approach to machine vision application development," *Proceedings of SPIE*, Machine Vision Systems Integration in Industry, Vol. 1386, pp. 158-162, November 8-9, 1990.

Sahoo, P. K., S. Soltani, and A. K. C. Wong, "SURVEY: A Survey of Thresholding Techniques," *Computer Vision, Graphics, and Image Processing*, Vol. 41, pp. 233-260, 1988.

Sanz, J.L.C., and D. Petkovic, "Machine Vision Algorithms for Automated Inspection of Thin-Film Disk Heads," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 6, pp. 830-848, November 1988.

Sawchuk, A.A., "Real-time correction of intensity nonlinearities in imaging systems," *IEEE Transactions on Computers*, vol.c-26,no.1,pp.34-39, January 1977.

Stefik, M.J., and D.G. Bobrow, "Object-Oriented Programming: Themes and Variations," *AI Magazine*, vol. 6, No. 4, pp.40-62, Winter 1986.

Stein, J.S, and Adeel Najmi, "Comparison of Conventional and Object-Oriented Approaches for Simulation of Manufacturing Systems," *Proceedings of IIE*, IIE Integrated Systems Conference & Society For Integrated Manufacturing Conference, pp. 471-476, November 1989.

Tanik, M.M, and R.T. Yeh, "Rapid Prototyping in Software Development," *Computer*, pp. 9-10, May 1989.

Turk, M. A., D. G. Morgenthaler, K. D. Gremban, M. Marra, "VITS - A Vision System for Autonomous Land Vehicle Navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 3, pp. 342-361, May 1988.

Verity, J.W., and E.I. Schwartz, "Software Made Simple: Will Object-oriented Programming Transform The Computer Industry?," *Business Week*, pp. 92-100, September 30, 1991.

Walsh, P.M., "Rapid system integration with symbolic programming," *Proceedings of SPIE*, Machine Vision Systems Integration in Industry, Vol. 1386, pp. 84-89, November 8-9, 1990.

Williams, T.D., "Image Understanding Tools," *Proceedings of IAPR*, 10th International Conference on Pattern Recognition, Vol. 2, pp. 606-610, June 1990.

Zavidovique, B., V. Serfaty, and C. Fortunel, "Mechanism to Capture and Communicate Image Processing Expertise," *IEEE Software*, pp. 37-50, November 1991.

VITA

Chong Teck Ng was born on November 2, 1957 in Melaka, Malaysia. He received his B.S. (1981) and M.S. (1984) degree in Electrical Engineering from Louisiana State University in Baton Rouge, Louisiana. After working as a programmer at the Remote Sensing and Image Processing Laboratory at LSU, he came to Virginia Polytechnic Institute and State University in 1987 and received his Ph.D. degree in Electrical Engineering in the Spring of 1993. During the course of his studies, he has been a Teaching Assistant, a Research Assistant, and a Graduate Project Assistant. At the time he received his Ph.D. he was employed by American Research Corporation of Virginia in Radford, Virginia, as a Research Scientist.

[illegible]