DEVELOPMENT OF AN INTELLIGENT THREE DIMENSIONAL OPTICAL INSPECTION SYSTEM

CHAN-TEH KUO

Bachelor of Science in Mechanical Engineering

Chung Yuan Christian University

June, 1989

Master of Science in Mechanical Engineering

Cleveland State University

December, 1993

Submitted in partial fulfillment of requirements for the degree

DOCTOR OF ENGINEERING in MECHANICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

May, 1999

ACKNOWLEDGEMENTS

I would like to give my sincere thanks to my advisor, Dr. Paul Lin for his guidance and suggestions to my research. The author also wishes to express thanks to dissertation committee members, Dr. Injazz Chen, Dr. John Frater, Dr. Ying-Hsin Liou and Dr. Taysir Nayfeh for their valuable comments and recommendations.

Finally, I would like to thank my wife Wei-Chin for help typing this dissertation. This dissertation is dedicated to her endless encouragement and support throughout my doctoral study.

DEVELOPMENT OF AN INTELLIGENT THREE DIMENSIONAL OPTICAL INSPECTION SYSTEM

CHAN-TEH KUO

ABSTRACT

Optical inspection is becoming more popular for quality assurance in manufacturing. Faster and more powerful computers help improve the speed of image processing and analysis. This dissertation presents the development of an intelligent three-dimensional optical inspection system. The main objective is to automate the inspection process.

This dissertation work consists of three major tasks: new calibration procedure, fuzzy logic based fringe center locator (FCL), and object recognition. The accuracy and reliability of fringe center detection are critical to three dimensional geometry determination, whereas object recognition is to recognize an object's shape and size. With the new calibration procedure, the value of calibration factor can be automatically adjusted when the camera angle changed.

The results indicate that two-dimensional objects can be successfully recognized, and the fringe centers can be automatically detected. The time needed to inspect the three dimensional geometry is about 30 seconds when using Matlab software. However, the same inspection can be completed in about 0.6 seconds if we convert the Matlab program into a C++ source code.

TABLE OF CONTENTS

Page

ABSTRACT iv

LIST OF TABLES ix

LIST OF FIGURES xi

CHAPTER

I GENERAL INTRODUCTION 1

1.1 Background and Significance 1

1.2 General Introduction 2

1.3 Literature Review 4

1.4 Motivation and Objectives Of Study 6

1.5 Dissertation Organization 7

II THE OPTICAL INSPECTION SYSTEM 9

2.1 Image Grabbing/Acquisition Hardware 9

2.2 Calibration and Image Processing 11

2.3 Image Processing Software 14

2.4 Applications 14

III THE TECHNIQUES AND ALGORITHMS 17

3.1 Machine Vision 18

3.1.1 Properties Of an Object in a Digitized Image 19

3.2 Fringe Projection Technique for 3-D Geometry Determination 20

3.3 Morphology – Thinning Method 21

3.4 Histogram 21

3.5 Vertical Line Enhancement 23

3.6 Object Recognition - Two-Dimensional Application in Machine Vision 24

3.6.1 Edge Detection in Object Recognition 26

3.6.2 Angle Of Sight (AOS) Signature in Object Recognition 26

3.7 Artificial Intelligence 30

3.7.1 Fuzzy Logic and Fuzzy Inference System 30

3.7.2 Fuzzy Inference System 33

3.7.3 Neural Networks 36

3.8 Calibration 38

3.8.1 Lens Properties 38

3.9 Design of Automated Inspection System 39

3.9.1 Algorithm for Fringe Center Locator by Means of Fuzzy Inference 40

Procedure for the Fuzzy-Logic Based Fringe Center Locator 41

Procedure for Reconstructing Discontinued Fringes 45

Finding the Exact Fringe Centers with Subpixel Resolution 46

3.9.2 Algorithms for Object Recognition 47

Image Threshold 47

Object Recognition by Means of Neural Network Mapping 48

Object Recognition by Means of Angle-of-Sight (AOS) Signatures 48

ROI Setting via Object Recognition 50

3.9.3 Algorithm for the New Calibration Procedure 50

Part I - Generation of a Calibration (Algorithm I) 51

Part II - Detection of Camera's Viewing Angle (Algorithm II) 54

Part III - Combination of the Two Algorithms 56

3.10 The Automated Inspection System 56

IV DESCRIPTION OF EXPERIMENTS 59

4.1 Fringe Center Locator (FCL) 60

4.1.1 Models Used in FCL Testing 60

4.1.2 Designing the Experiments 62

4.1.3 Different FCL and Subpixel Calculation Algorithms 64

4.2 Object Recognition Techniques 65

4.2.1 Objects Used in Object Recognition 66

4.2.2 Object Recognition by Means of Neural Networks 67

4.2.3 Object Recognition by Means of AOS Signatures 69

4.3 Auto-Calibration and Automatic Angle Adjustment 71

4.3.1 Establishing the Calibration Map 71

Part I - Using the Matlab Scripts 71

Part II - Using the Inspector 72

Surface Fitting 73

Performance comparison between Using Matlab and Inspector 73

4.3.2 Automated Camera Angle Detection Module 74

V RESULTS AND DISCUSSIONS 77

5.1 The Test Results of FCL and Subpixel Calculation..... 77

5.1.1 Comparison between Fuzzy Logic Based FCL and Traditional FCL..... 78

5.1.2 Comparison between Bisection-based numerical Method and Analytical Method 80

5.1.3 Comparison between Running a FORTRAN program and Matlab scripts 81

5.2 The Accuracy of fuzzy-logic based FCL Applied to Different Kinds of Surfaces 82

5.3 The Results of Object Recognition..... 89

5.3.1 Results of Testing the Neural Network Algorithm 90

5.3.2 Results of Testing the AOS Signature Algorithm..... 90

5.4 The Calibration Map and the Camera Angle Detection..... 94

5.5 Testing of the New Optical Inspection System 97

5.5.1 The Result on a Smooth Surface of a Clay Model..... 98

5.5.2 The Result on a Discontinued Surface 99

5.5.3 The Processing Time Using Different Algorithms 99

5.6 Discussions 104

VI CONCLUSION 106

REFERENCES 109

APPENDIX A Data for Establishing AOS Objection Recognition Look-up Table 114

APPENDIX B Look-up Table Used in the Matlab Script 115

APPENDIX C Results of finding angular distances (ICDs) for 32 tesing objects 116

LIST OF TABLES

TABLE Page

4-1 The 92 Fuzzy Rules for the Fuzzy FCL....63

4-2 The Programs and their Algorithms....64

4-3 The Parameters for Comparisons of Different Algorithms....64

4-4 Descriptions of Models Used in Object Recognition....67

4-5 The BPNN Training Data....68

4-6 The Look-up Table for AOS Signature Recognition Algorithm....69

4-7 The Performance of the Matlab Script and the Inspector in Finding Calibration Map for 25 Points....73

5-1 The Differences of Calculated Centers between Program Fr7.exe and Fuzzyfringe.m....79

5-2 The Differences of Calculated Centers between the Old and The New Systems on a Flat Surface....79

5-3 The Differences of Calculated Centers between the Old and The New Systems on a Very Curved Surface....80

5-4 The Differences of Calculated Subpixel Centers between Program Fr6.Exe and Fringe98a.M....81

5-5 The Differences of Calculated Centers between Using Fortran Program and Matlab Script....82

5-6 Result of Accuracy Test of Fuzzy FCL (Model 1, Fringe 3) 84

5-7 Result of Accuracy Test of Fuzzy FCL (Model 1, Fringe 6) 85

5-8 Result of Accuracy Test of Fuzzy FCL (Model 2, Fringe 1) 86

5-9 Result of Accuracy Test of Fuzzy FCL (Model 2, Fringe 8) 87

5-10 Result of Accuracy Test of Fuzzy FCL (Steel Block, Fringe 2) 88

5-11 Result of Accuracy Test of Fuzzy FCL (Steel Block, Fringe 7) 89

5-12 A Sample of Output by The NN Object Recognition Algorithm 91

5-13 The Result of Object Recognition through Neural Networks.....92

5-14 The Result of Object Recognition through AOS Signature.....93

5-15 Automatic Angle Detection Using Neural Network.....95

5-16 The Comparison of Processing Time between Bisection and Analytical Solution Sub-Pixel Algorithms.....101

5-17 The Comparison of Processing Time for 3-D Surface Reconstructed between Fortran Program and Matlab Script.....102

5-18 The Comparison of Computing Time between the Old and New Systems.....103

LIST OF FIGURES

FIGURE Page

2-1 The Current Optical Inspection System Setup 10

2-2 The Calibration Target 12

2-3 An Artificial Femur Bone with Optical Fringes 12

2-4 The Reconstructed Surface of a Femur Bone 13

3-1 Intensity Distribution of a Row Of an Image 22

3-2 A Histogram of a Row of an Image 23

3-3 A Cylinder and its AOS Signature 27

3-4 Two Different Angles of a Steel Block 28

3-5 The AOS of the Steel Blocks for 0° and 45° 28

3-6 A Photo of Two Different Size of Steel Blocks 29

3-7 AOS Signatures of the Two Different Sizes of Steel Blocks 29

3-8 The Membership Function for Tall 32

3-9 A FIS Flowchart for the Tipper Example 35

3-10 A Backpropagation Neural Network 37

3-11 Intensity Distribution of 5 Pixels across a Fringe (Pixel 1 and 7 are Background) 40

3-12 The Flowchart of the Fuzzy-Logic Fringe Center Locator 43

3-13 Some Properties of an AOS Signature 49

3-14 Distortion of Camera's Lens 51

3-15 A Typical Calibration Map.....53

3-16 The Printout Calibration Target with 25 Circles on it 53

3-17 Screen Shots of the Circle with Different Viewing Angles.....55

3-18 Calibration Maps with Different Yaw Angles Prepared for NN Training.....56

3-19 The Flowchart Of the New Calibration Procedure.....57

3-20 The Flowchart of the New Optical Inspection System.....58

4-1 A Clay Model (Model 1) with Projected Fringes 60

4-2 The Second Clay Model (Model 2) with Very Curved Surface.....61

4-3 A Steel Block with 3 Step Increases 61

4-4 A Graphical Representation of Four Types of Intensity Distribution.....62

4-5 The Models Used in Object Recognition.....66

4-6 The Plot (Epoch Vs. SSE) of the Training Process.....68

4-7 A Rough Dorm-Like Surface Generated by 25 Points 72

4-8 The Block Diagram for the Camera Angle Determination via the Trained Neural Networks 75

4-9 The Flow Chart of the Automated Inspection System 76

5-1 The Calibration Map Established by 49 Data Sets 94

5-2 The Mapping between the Calibration Circle's Position and Camera Angles 96

5-3 Two Calibration Maps Calculated From Different Camera Angle (Meshed: Yaw=0°,Pitch=0°; Solid: Yaw=3°, Pitch=-5°).....97

5-4 The Clay Model with and without Fringes.....98

5-5 The Reconstructed Surface of the First Demonstration 99

5-6 The Reconstructed Surface of the Steel Block and the Inspection Platform (exact height increase is 0.1"). 100

5-7 The Side View of the Reconstructed Surface along one of the Fringes (4th) 100

CHAPTER I

GENERAL INTRODUCTION

1.1 Background and Significance

We have developed an optical inspection system that can measure and reconstruct the three-dimensional surface of a given object accurately [Lin, et. al., 1990 and 1992]. This measuring accuracy of this system relies heavily on how well the projected fringe centers are located. Before the fringe centers can be accurately determined, it is necessary to approximately locate the centers, which is not easy to do using our existing techniques. These is because several parameters must be specified manually in order to automatically set the region of interest (ROI), and locate the fringe centers. In particular, setting the ROI is time consuming, and requires some knowledge and experience from the operator.

To automatically set the ROI, the shape of an object should be obtained first. As soon as we get the object's shape, it is to be recognized and useful data such as the object's dimensions can be extracted from the database. Using the shape recognition to set the ROI for the new fringe center locator (FCL) to automate an inspection process is the theme of this research.

In addition to automating the ROI setting, the calibration technique needs to be improved. In our optical inspection system, the positions of the camera and the projector are very crucial for measuring accuracy. Two angles are used in our computing algorithm, which are the camera's viewing angle, and the projecting angle. The camera should be set perpendicular to the object's plane, and the projecting angle is measured from the incident light to the direction of the surface normal of the object. The former must be set accurately, while the latter can be determined by analyzing the output result (i.e. 3-D surface data). As we all know, calibration is a time consuming process. Therefore, it is always desirable to avoid the necessity of having to recalibrate the inspection system as long as the camera angle is still within the allowable range.

1.2 General Introduction

Today many robots are used in manufacturing for automation. In order to make robots more versatile and efficient, some of them are equipped with sensors, such as cameras. Machine vision is one of the essential capabilities for a robot to function like human eyes. For example, if there are multiple parts under inspection, machine vision can make the distinction among them. Furthermore, if a machine vision system is equipped with artificial intelligence (such as neural network and fuzzy logic), the reliability of inspection can be much improved [Moon and Na, 1996]. Safety is another reason for using machine vision. In an autonomous warehouse, carts are moving on pre-designed routes and controlled by programs. It is assumed there is no people entering this warehouse. Machine vision, in this case, can be used to monitor the work area for

intruders and trigger an alarm.

Quality control is another important issue for today's manufacturers. Production lines are often operated at high speeds. A typical speed for bottling lines is between 3 and 30 containers per second [source: A.V.S. Inc. 1996], which is impossible for a human inspector to do bottle inspection this fast. In order to stop defective products immediately, a real-time or nearly real-time inspection system must be used. On a high-speed production line even a few minutes delay in detecting and correcting a problem can result in a large cost due to lost production time. Human inspectors can only achieve 80% reliability in visual inspection. Machine vision systems, in general, can perform over 95% reliability. Automatic high-speed inspection using machine vision is one of few available alternatives to assure the quality consistency.

Contact and non-contact measurements are used to perform quality inspection in manufacturing. Coordinate Measuring machines (CMM) uses a contact probe to detect and measure the 3-D geometry of an object, one point at a time. To many manufacturers, this is considered satisfactory. However, contact measurements are too time consuming. Among several non-contact methods, optical measurement is most accurate. Basically, optical techniques rely on surface reflection, determine the 3-D geometry based on the geometric relationships between the light source and the object under measurement.

It is desirable to automate the inspection process without the user's intervention. This will require the inspection system to be more intelligent, which can be achieved by adding artificial intelligence (AI). There are three major techniques in AI: artificial neural network (ANN), fuzzy logic (FL) and genetic algorithm (GA). In addition, some hybrid neural-fuzzy techniques can be used as well. In this case, the fuzzy rules can help to

decide the weights of a neural network and the output of the network can be fed back to change the fuzzy rules [Kawamura et. al., 1992].

Before a neural network can be used, the network must be trained with a certain amount of data [Fausett, 1994]. If a fuzzy inference system is chosen, a rule base has to be established in advance [Jang and Gulley, 1995]. In the past, computers were not fast enough, and the data storage capability was a big concern. But, today's computers are hundreds of times faster than before, and the data storage device is becoming cheaper and better. Using AI techniques has become the trend for making an inspection system more intelligent.

1.3 Literature Review

Machine vision is widely used in manufacturing industry today. 3-D Optical inspection and measurement system is important to today's manufacturers. The widely used structured light and our optical fringe projection are good techniques for 3-D inspection [Lin and Kuo, 1998]. For instance, the fringe projection technique can be employed to estimate the volume of a missing fragment of a surface [Jones and Plassmann, 1992]. To obtain accurate 3-D measurement with the structured light inspection system, an accurate moving mechanism must be used. Fringe projection technique observes the position change of the optical fringes can explicitly quantify the change of the third dimension of the object coordinate. The accuracy of the geometry measurement depends on how accurate the fringe centers are detected. The sub-pixel (i.e. a fraction of pixel or picture element) resolution was then developed to improve the accuracy [Lin, 1990].

The fringe-center locator algorithm used in our current optical inspection system is accurate and computationally efficient [Lin, 1990]. The intensity distribution across a fringe is similar to a Gaussian distribution. The fringe center is located exactly at a point where the intensity is at the maximum. A successive difference method was used to find the approximate center location, however, it requires the user to input some key parameter values to begin the inspection process.

Another application of using machine vision is to recognize objects from digitized images. For example, a possible application may be to distinguish overlapped parts [Breuel, 1996]. Through image analysis, some features of objects can be extracted. [Al-Kindi and Baul, 1991] defined the contour encoding, which is a collection of radii from the centroid to the object's edge, counterclockwise or clockwise. The uniform spaced polar encoding was utilized in their work. Matching the elements of the radius vectors of two objects was used for object recognition. When compared with the referenced feature in a database, an object can be recognized with a minimum squared error.

Artificial intelligence is becoming more popular. Technical books, such as “Neural Network Fundamentals with Graphs, Algorithms, and Applications” [Bose, 1996] and “Fundamental of Neural Networks”, give good description of neural networks. The user’s manual of the Matlab’s fuzzy logic toolbox from Mathworks corporation is a good resource to learn the applications about fuzzy logic and fuzzy inference system. Furthermore, a machine with artificial intelligence can recognize overlapped parts accurately [Law et. al., 1995].

Fuzzy approach with the angle-of-sight (AOS) signature proposed by [Popovic and Liang, 1994] is another way to recognize an object. By converting the AOS signature to

Cartesian coordinators, the outline of an object can be reconstructed. From the AOS signatures, one can know the shape and size of a regular geometry object. By matching the referenced and actual AOS signatures, the distinct feature of an object can be detected. [Popovic and Liang, 1994] also used a fuzzy logic approach to calculate the distance between the actual and the referenced AOS signatures. The object can be recognized quickly and accurately. However, the orientation of the object is limited to one direction. With fuzzy approach, a machine can even recognize those complex shapes, such as the handwritten Chinese characters [Cheng et. al., 1989].

With neural networks, a system can overcome the reliance on a functional form of the forecasting model and large historical database [Peng et. al., 1992]. If the features of an object are collected as input set, a trained neural network may be a good method to predict the output of unseen data. [Pomerleau, 1989] created an autonomous automobile, which was controlled by a neural network controller. The data were collected from a 256x256-pixel image of its environment, and were divided into an 8x8 matrix. Those 64 elements are sent to a well-trained neural network to control the vehicle. Another similar application is developed by [Hirota et. al., 1994]. They employed fuzzy technology to understand the dynamic image of a general road. This system could give the real-time road condition to the user or controller.

1.4 Motivation and Objectives of Study

We have developed a 3-D optical inspection system by means of fringe projection technique and subpixel fringe center calculation. The 3-D geometry determination is

based on detecting the fringe centers. However, fringe centers were sometimes misdetected. In this case, the inspection cannot continue until the user specifies some parameter values. Our original program used to work under the MS-DOS environment. Although it was translated into the equivalent Matlab script under the MS-Windows, the user still needs to enter some parameter values. Those are major obstacles in automating the system. Another problem is that the system cannot detect a small change of the camera angle.

The following four major objectives are to be achieved:

(1) To automate the existing optical inspection system.

(2) To accelerate the inspection process.

(3) To develop a reliable calibration and an auto-detection of camera's angle procedures.

(4) To develop an on-line fast optical inspection system embedded with machine intelligence.

1.5 Dissertation Organization

The three major tasks in this dissertation work are summarized as follows:

Task	Description
Calibration and camera's angle detection	Calibration is performed in the beginning of an inspection, and the calibration factor is adjusted according to the angle of the system's camera.
Fuzzy logic fringe center locator (FCL)	Optical fringes' centers are detected using fuzzy logic techniques. There is no need for the user to enter any parameter values.
Object recognition	Recognize an object, and collect geometric information for the object.

This dissertation is divided into six chapters. Chapter I is about general introduction, which includes the motivations and objectives of this research. Chapter II describes the optical inspection system currently used in our optical inspection laboratory, and the problem encountered.

Chapter III introduces the computing algorithms and the new techniques used in this research. In the beginning of this chapter, machine vision and its applications are introduced. Some useful techniques, which will be utilized in developing new algorithms, are described here. This chapter contains descriptions of neural networks, and fuzzy logic. The final part of this chapter gives the detailed descriptions of a modified optical inspection system embedded with AI, a new object recognition module and the new system calibration procedures. Chapter IV and Chapter V describe the experiments and the results of experiments, respectively. Chapter VI includes the conclusion and recommendations for future work.

CHAPTER II

THE OPTICAL INSPECTION SYSTEM

Like many other optical inspection systems, our system is personal computers (PC) based. We have the hardware and software to acquire and process images, respectively. This chapter contains four sections: the hardware, a brief description of calibration and processing procedures, the selection of software, and the applications of this inspection and measurement system.

2.1 Image Grabbing/Acquisition Hardware

The first step in image processing is to capture images. Recently, new equipment such as a black and white CCD camera, a frame grabber with PCI data bus, a diode laser projector and a Pentium II PC were installed in this inspection system. The camera is from Hitachi, model KP-M1 with the dimension of 44(W) x 29(H) x 72(D) mm. It uses the latest high-grade 2/3-inch image size CCD with a total of 410,000 pixels. The 16mm fixed focal length made by Javelin Electronics, can avoid possible image distortion caused by zoom lens. The new frame grabber "Meteor" is made by Matrox. With its four input lines, we are able to connect to four different inspection stations. The entire system setup

is shown in Figure 2-1.

Two possible light sources can be used, which are white light and red light. A Hi-Ne laser generates the red light source, which is consistent, coherent and more focused.

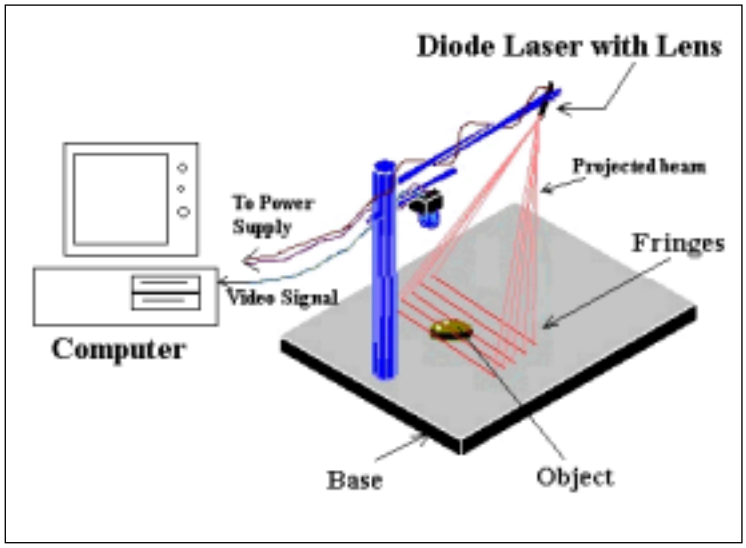

Figure 2-1. The Current Optical Inspection System Setup

With a special red light filter, the camera can capture images without turning the room lights off. However, the light intensity may be so low that it can only be applied to a small area at a short distance. Because of this disadvantage, a white light projector is sometimes used. Its average 80Watts light output gives a good working distance with a large illumination area. The measuring accuracy is almost as good as with a laser light source. However, the room light may interfere with the white light. Sometimes, the room light will have to be turned off.

Recently we added a new diode laser projector, DLS-500 series laser powered by a 9VDC battery, made by LASIRIS. The advantage of this laser projector is that it comes with special lenses. When a laser beam passes through a special lens, 33 or 99 fringes are projected onto the inspection platform. An alternative to the diode laser projector is a

grating called Ronchi ruling made of a piece of glass on which many equally spaced lines are printed.

A ring light made by Illumination Technologies Inc. was also added into our inspection system. It produces a direct, uniform and controllable light intensity. The new computer is a Pentium-II 266 Mhz personal computer. It has 80MB RAM for calculations and temporary data storage, and a 3GB hard disk drive to store image files, rule base and database.

2.2 Calibration and Image Processing

Before calibrating the optical system, we need to check the hardware setup. Firstly, the CCD camera's viewing axis must be perpendicular to the object's plane (i.e. surface of the inspection platform). Secondly, the angle between the projector and the object plane, called incident light angle, must be determined in advance. Thirdly, the fringes projected onto the inspection platform must be aligned with the vertical edge of the computer screen (i.e. y-axis).

To transfer the results in pixels into the real dimensions in mm or inch, a conversion called calibration factor must be determined. The calibration procedure used in our optical inspection system is straightforward. The calibration target (Figure 2-2) is a piece of opaque glass with dimensioned lines on it. There are several different spacings between lines in this target. The first step of calibration is to find the calibration factor (CF). A program was written to find the distance between any two parallel dark lines in a

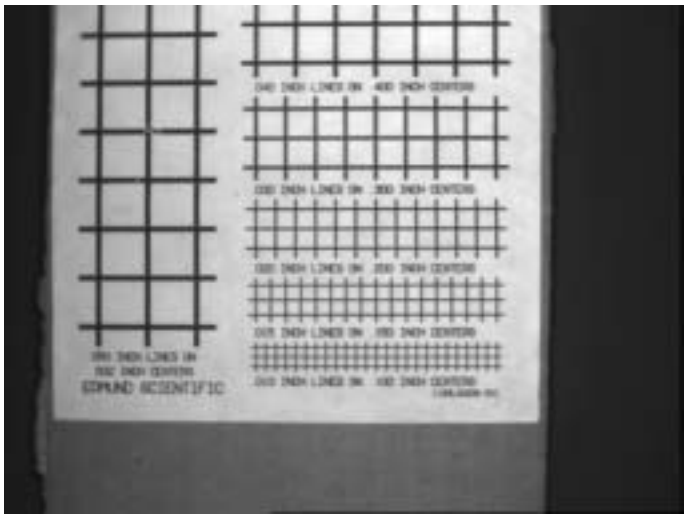

Figure 2-2. The Calibration Target

region. The calibration factor is the ratio of real dimension versus image dimension in the unit of mm/pixel or pixel/mm.

The second step is to place an object for inspection on top of the platform, and project fringes onto the object, as shown in the Figure 2-3. When parallel fringes are

Figure 2-3. An artificial Femur Bone with Optical Fringes

projected onto the object's surface, if the surface is curved, the fringes will be curved, and the fringe spacing could also vary from place to place. Reference line spacing between two lines should be known in advance. By comparing the difference between the curved line spacing and the referenced line spacing, the height change between two lines can be obtained [Kuo, 1993].

It is time consuming and complex to scan the entire image of a 3-D object, and reconstruct the 3-D surface, because some parameters must be specified in a program. The most important parameter is the ROI (region of interest). Even with a fast Pentium II PC, tens of seconds are needed just to scan a 640x480 pixels image. By setting the ROI, the image analysis can be focused on a smaller region to save time.

The aforementioned calibration process usually takes an experienced user several minutes. Afterward, the fringe centers' locations are fed into our 3-D surface reconstruction program. As an example, Figure 2-4 shows the reconstructed surface of a human femur bone.

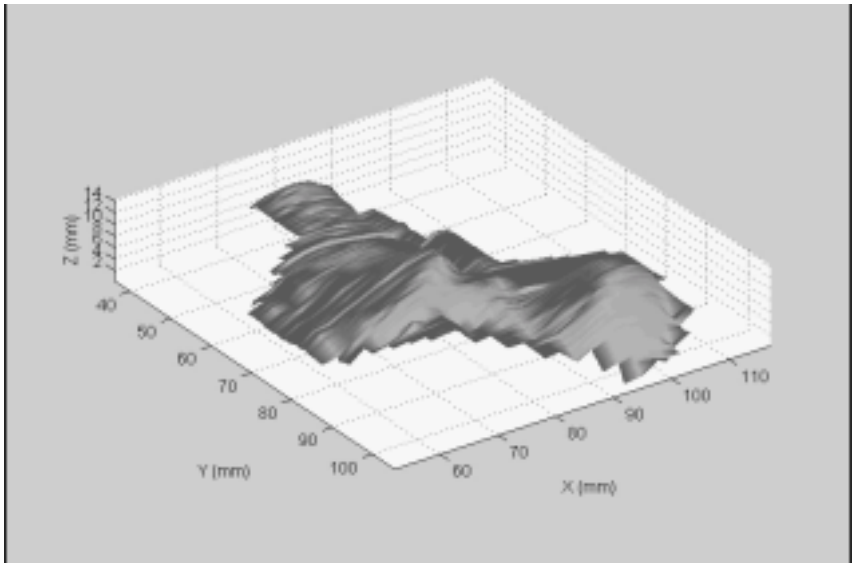

Figure 2-4. The Reconstructed Surface of a Femur Bone

2.3 Image Processing Software

The software used for image processing is Matlab's image processing toolbox, a matrix based mathematical software. It provides a simple environment to write script files. A script is executed for each application, and can be translated into C/C++ language to become a standalone executable program. According to a statement in the Matlab manual, with translated and compiled C source codes most do-loop script file can be executed 70 times faster than that written in Matlab language [Matlab compiler, 1995].

Since the toolbox can not capture a live video signal, extra commercial software was used. Commercial software called Inspector 2.1 developed by Matrox, works with Matrox's frame grabber nicely. It can capture images and perform some simple pre-processing tasks such as filtering. Afterward, the saved image can be read by Matlab for post-processing.

2.4 Applications

The following five application projects have been conducted by our laboratory this far.

(1) "Radius measurement via fringe projection technique using the edge detection with subpixel resolution" was a project supported by the Cleveland Advanced manufacturing program (CAMP), a division of the Ohio's Thomas Edison Program [Lin and Parvin, 1990]. In this work a methodology for automated non-contact radius measurements was developed in which a fringe was selected and the radius of a cylinder was calculated thereafter. Unlike Moire contouring technique, the fringe projection is not very sensitive to vibration. With the fringe projection technique, the

measuring accuracy was very good because more than 100 image points were used to determine the geometric property (radius, in this case).

(2) "Optical gaging of very short-term surface waviness" was another project supported by CAMP [Lin, Parvin and Schoenig, 1992]. Surface waviness is closely related to the change of the third dimension (i.e. the depth). Inspection of very short-term surface waviness of a thin and deformable solar panel was a challenging task. In this project, the RMS waviness of four samples was generally under 0.1 □m. The measuring accuracy of our optical inspection system was achieved by the subpixel resolution, whereas the repeatability was accomplished by the algorithm's low sensitivity to noise. The optical measurement results were verified by using a profilometer contact measuring device, and the RMS waviness obtained from the optical measurement was generally in agreement with those obtained from the profilometer.

(3) "Development of an optical technique for geometry determination of an irregular curved surface" was a project supported by Advanced Manufacturing Center (AMC) at Cleveland State University. A turbine blade, approximately 4 by 1.5 inch made of metal, was used as a typical curved surface. CMM (coordinator measuring machine), which is widely used in industry, can produce very accurate XYZ data for a flat surface. The optical measured data were compared with those obtained from a CMM. The comparison indicated that they were very much in agreement at many locations. In fact, the optical technique could measure some points in which their curvatures drastically changed. In contrast, the CMM's measurement was not that sensitive.

(4) "Three dimensional measurement of a rotating object by means of an optical technique" was a project aiming at dynamic measurements [Kuo, 1993]. With the help

of a high resolution VCR, many images could be captured and immediately frozen. The output was also verified by comparing the data obtained from the static measurement of a profilometer. The dynamic measuring accuracy of the rotating object was satisfied.

(5) "Deformation data analysis for dynamic testing of F-16 Bias and radial tires" was a project performed at the US Air Force Base [Lin, Kuo and Chawla, 1997]. Due to a potential danger of blowing up a loaded tire, it was necessary to use a non-contact technique. The tire deformation measurements were successfully completed, in which F-16 bias and radial aircraft tires were compared and some conclusions were drawn.

CHAPTER III

THE TECHNIQUES AND ALGORITHMS

Optical inspection uses machine vision and optical techniques. The key elements of machine vision are image processing and image analysis. The former requires a frame grabber, and the processing software. In contrast, the latter is application dependent and thus more challenging.

Accurate calibration is also important to an optical inspection system. Although many optical inspection systems are equipped with an auto-calibration module, they are lack of auto-compensation and self-adjusted functionality. To develop a system that can detect the changes of its environment, and automatically adjust the calibration factors is essential and challenging.

In this chapter, some techniques, which are the cores of this research, will be introduced first. The first section is machine vision, followed by object recognition, then neural networks and fuzzy logic. In order to enhance our existing algorithms for optical measurement, and to automate the inspection process, neural networks and fuzzy logic techniques will be utilized.

Some new algorithms developed in this work are presented in this chapter. The

first one is the new fringe center locator (FCL) using fuzzy logic. The second one is object recognition by means of angle-of-sight (AOS) signature. A new calibration procedure is also presented.

3.1 Machine Vision

Vision capability allows a machine to quickly respond to the change of its environment. Human vision is more versatile and faster than machine vision. A generally accepted formal definition of machine vision is described as follows [Jain, et. al., 1995]:

"**Machine vision** is the automatic acquisition of images by non-contact means and their automatic analysis to extract needed data for controlling a process or activity."

Although an image is two dimensional, it does contain some information that can be extracted and interpreted. Machine vision is a sophisticated process of extracting, characterizing, and interpreting information from images. Generally speaking, machine vision can be divided into six parts: grabbing, preprocessing, segmenting, describing, recognizing and interpreting. Grabbing is the process of capturing and digitizing the image of an object. Like human's eyes, machine uses a CCD camera to capture images, and a frame grabber to digitize and store images. Preprocessing deals with techniques such as noise reducing and detail enhancing. Segmenting is the process that partitions important objects of interest. Describing deals with the computation of features (e.g. shape) suitable for differentiating one type of object from another. Recognizing is the process that identifies these objects (e.g., wrench, and bolt), and interpreting assigns meaning to an ensemble for recognized objects. Machine vision not only recognizes 2-D

shape of an object, but also quantifies the object's size. For 3-D geometry determination, some optical techniques, such as stereo vision, Moier method [Pericles, 1969] or fringe projection may be added to extract the three dimensional geometric information from a 2-D image.

Properties of an Object in a Digitized Image

Position, size, color, and shape are some basic properties of an object. The first thing for a machine to recognize an object is to give some descriptions about the object such as "Where is it" or "What is its size". For many machine vision applications, the most important property of an object is its position. There are different ways to specify the position of an object, such as using its enclosing rectangle or centroid. In industrial applications, objects are usually placed on an inspection platform, and the position of the camera is known with respect to the platform. The simplest way to define the position of an object in an image is to specify the object's center in the image coordinates. If we consider the value of each pixel as the mass at that point, then the center of an object in a binary image is the same as the center of mass. Thus, we can calculate the position of the object the same way as for calculating the center of mass. The second property is the object's orientation. The orientation of an object can be determined by using the moment functions [Jain, et. al., 1995]. To define a unique orientation, the object's aspect ratio must not be equal to 1. In doing so, the orientation of the elongation axis can be used to define the orientation of the object. If an object has a symmetric shape, a unique orientation cannot be determined. However, some analytical equations for this situation were proposed by [Tsai and Chou, 1991].

3.1 Fringe Projection Technique for 3-D Geometry Determination

Fringe projection has been developed for 3-D optical metrology, in which we have developed some algorithms to accurately determine the 3-D geometry of an object [Lin and Parvin, 1990]. Fringe projection uses a single camera and a Ronchi ruling to generate fringes. The fringe projection combined with a subpixelation technique has proven to be an accurate method to reconstruct an object's 3-D geometry.

When optical fringes are projected onto the surface of an object, it is important to accurate locate the fringe centers. There are many ways to locate the fringe centers. For a curved surface, the spacing between any two adjacent fringes varies. By analyzing a row of intensity values, it is not difficult to approximately locate the fringe centers by means of a histogram. Another technique may also help finding the fringe centers through image enhancement such as morphology, the so-called thinning method.

When we examine a row of intensity values by human eyes, the approximate centers can be located easily. All this is accomplished by human intuition. Thus, an artificial intelligence-based (AI) approach might be a good candidate for finding the fringe center automatically. Among the existing AI techniques, neural networks and fuzzy logic techniques are gaining popularity, which will be discussed later. In the following sections, three image pre-processing algorithms finding fringe centers are described.

3.2 Morphology – Thinning Method

The feature of this filter is that thick lines may be thinned to one pixel wide after several operations [Matlab Inc. 1997]. The thin lines may or may not be close to the exact

fringe centers. The widths of all fringes become very small, and the thinned images will be in binary format, instead of gray-scale. The main advantage of using this filter is that one-pixel wide fringes can be used to approximately locate the fringe centers using a simple algorithm. However, this filter has the following disadvantages:

(1) It takes about 2.4 seconds to thin the 640x480 pixels image in a Pentium II 266Mhz PC, and sometimes the thinned lines are not one-pixel wide as expected. Some fringes are, indeed, two-pixel wide.

(2) A thinned image is often distorted, and the thinned fringes may be shifted from their original fringe centers. Thus, the measuring accuracy may be affected due to fringe center dislocation.

(3) Our fringe-center-locator program that works only for gray-scale files can not use the thinned binary image.

3.4 Histogram

A histogram shows the frequency distribution of data [Hoffman, 1998]. All elements in a vector are grouped according to their values. Each group is shown as one bin. The histogram's x-axis reflects the range of values in that vector. The histogram's y-axis shows the number of elements that fall within the groups; therefore, the y-axis ranges from 0 to the greatest number of elements deposited in any bin.

Figure 3-1 shows the intensity distribution of a row across several fringes, and Figure 3-2 shows the histogram of intensity of the selected fringes. In a histogram plot, the largest

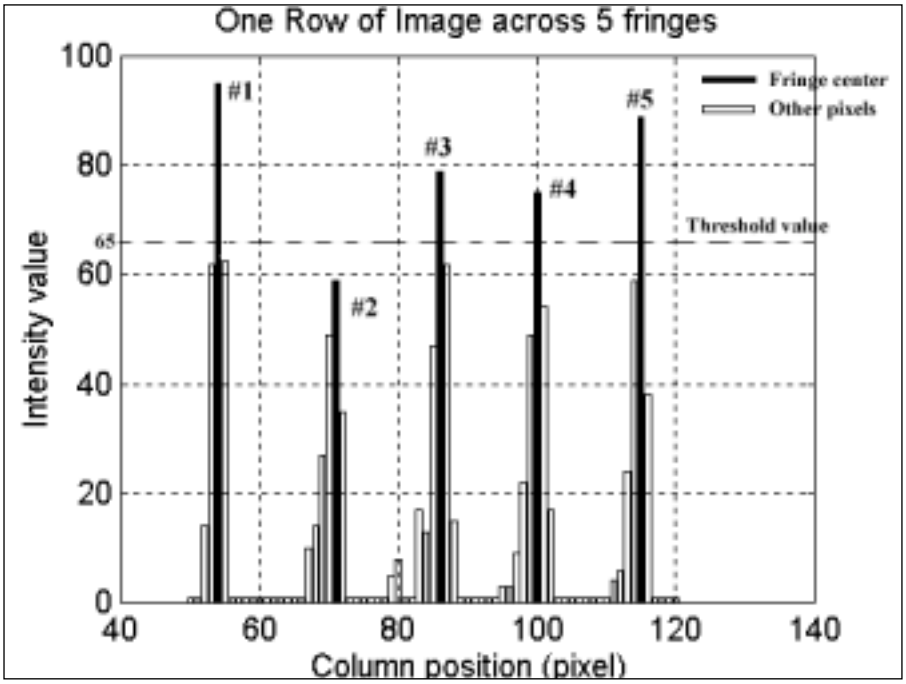

Figure 3-1. Intensity Distribution of a Row of an Image

intensity values are usually located at central fringe. The number in a bin indicates the number of fringes. But, this method is only suitable under certain conditions. In this example, since the intensity value of fringe 2 is lower than the preset threshold value, the center detection of fringe 2 will be missed. For a successful operation, the intensity values of central fringes have to be relatively larger than others. In other words, the intensity distribution is desired to be Gaussian-like. Unless the object is well illuminated, and the surface reflectivity is consistent, a Gaussian-like distribution usually does not appear in this kind of histogram.

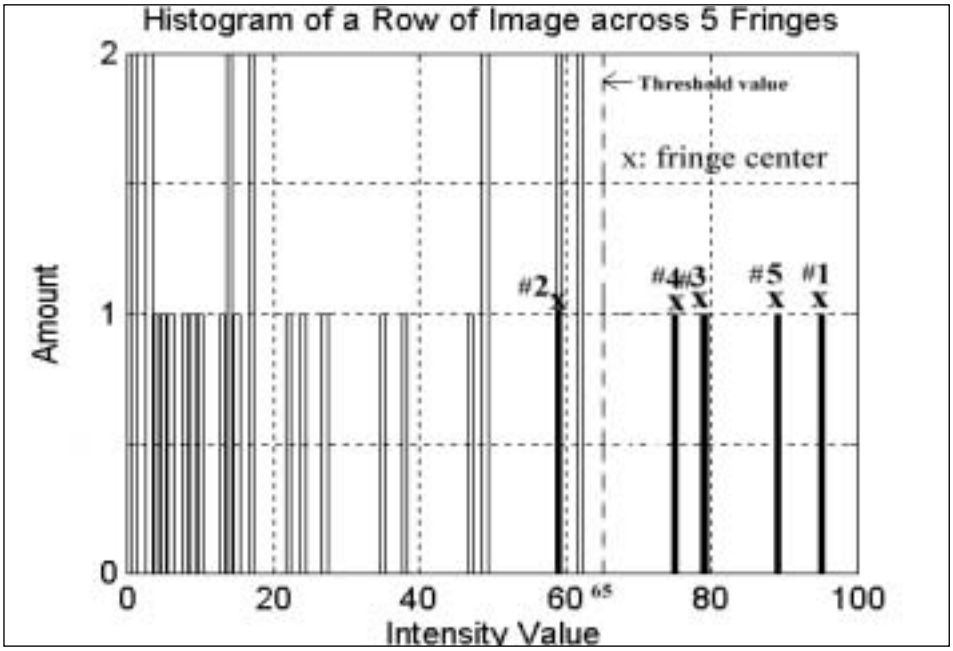

Figure 3-2. A Histogram of a Row of an Image

3.5 Vertical Line Enhancement

A filter alters each pixel's value from its current value, and the values of the neighboring pixels as well [source: JASC software, 1996]. The heart of a filter is an array of coefficients called a filter matrix or mask matrix. A filter processes an image pixel by pixel; each pixel's intensity value is multiplied by the coefficient in the matrix center, and any pixels within the matrix are multiplied by the corresponding coefficients. The sum of the products becomes the target pixel's new value. The new value is saved in a separate bitmap so that it does not affect the remaining pixels. The formula for this calculation is $F = \sum P_i C_i$, where F is the filtered value of the target pixel, P is a pixel in the grid, and C is a coefficient in the matrix.

Vertical line enhancement (VLE) is a common filter used in image processing. It uses a mask matrix to perform a mathematical operation with each pixel within the ROI. It has many advantages that can be used in our optical inspection system. They are:

(1) The intensity values across a well-illuminated line are Gaussian-like. Therefore, after the VLE filtering, a fringe becomes narrower, and the fringe center becomes brighter, while the edge of the fringe may be darker. Another advantage of using this filter is to remove noises such as white dots in the background.

(2) We can use this enhanced image to find the approximate fringe centers, and the exact fringe centers later. The accuracy of 3-D geometry determination heavily relies on to the quality of the filtered image.

(3) This enhancement is accomplished by a very simple 3x3 mask matrix as shown in the following:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

3.6 Object Recognition - Two-dimensional Application in Machine Vision

Let us consider an object recognition task, which is to take a snapshot of a table and return the location of all recognized utensils. The requirements for the recognition algorithms are size independence, robustness, short processing time, orientation independence and extensibility [IRL, 1998]. In object recognition, we need to describe an object completely and exactly [James, 1988]. Many of the early researchers relied on personal thoughts to suggest possible programmable methods. However, it seems likely that our knowledge of what happens during recognition is a long way. There is no simpler

recognition processing with human's thinking method. Our conscious processes are probably being supplied with information that is already the result of a great deal of processing. The only way to make any real progress in recognition is to try to find a general framework that does not depend on the particular problem in hand.

There are three major steps in object recognition - recognition, classification and understanding. After some features of an object have been extracted, the next step is to classify the object. Being able to classify an object into one of a number of groups is clearly something that might lead to a possible direction for object recognition. For example, character recognition is to classify a given image into one of the 26 letters of alphabet. Classification theory has a long history in many different fields, and it is important for anyone interested in pattern recognition. Image understanding is the technique of developing computer algorithms which create descriptions for particular purposes with given images [Firschein, 1995].

Understanding is usually in the context connection. For example, after some words are recognized, it is important to examine that they are fit in the sentences. Another example is in reading a satellite image. The machine should know what a house is or where the mountain ranges are. Furthermore, a fuzzy inference approach can perform the feeling processes of the human to the images [Chen, et. al., 1994]. With the help of the fuzzy technique, a vision system can give the road condition without considering the weather condition and daylight changing. [Nakagawa and Hirota, 1995]. Some traditional techniques are used in object recognition. They are edge detection, angel-of-sight (AOS) signatures. The following section describes these techniques.

Edge Detection in Object Recognition

When an object appears in an image, usually it has a clear contour or boundary. In a binary image, the boundary value is either 0 or 1. In a gray-scale and high contrast image, the boundary is located where the pixel intensities drastically change. However, gray-scale images are not always in high contrast. To extract an object from its background, some techniques such as image thresholding and filtering have been developed.

It is easy to perform edge detection for a binary image than a gray-scale image. Laplacian or Sobel filtering can retain the object's outlines, and remove others. Filtering and thresholding are the two popular methods for edge detection, which is the first step in object recognition.

Angle of Sight (AOS) Signature in Object Recognition

Recognizing an object in an image may be a difficult task. A special method called Angle-of-Sight (AOS) signature [Popovic and Liang, 1994] is adopted in this work for object recognition. Along with this method, a fuzzy logic technique is employed to more efficiently classify inspected objects into different categories.

There are several steps to translate an object into an AOS signature. The first step is to analyze the region in order to find the geometric properties of an object, such as area, centroid and angle of major axis [Jain, et. al., 1995]. The next step is to collect the perimeter pixels of the desired object, and transform them into polar coordinates. The radii, which measure from the center to any pixel on the perimeter, can be calculated for

every several angular degrees. The AOS signature is drawn as a plot of angle versus radius, as seen in Figure 3-3. Every object has its own signature.

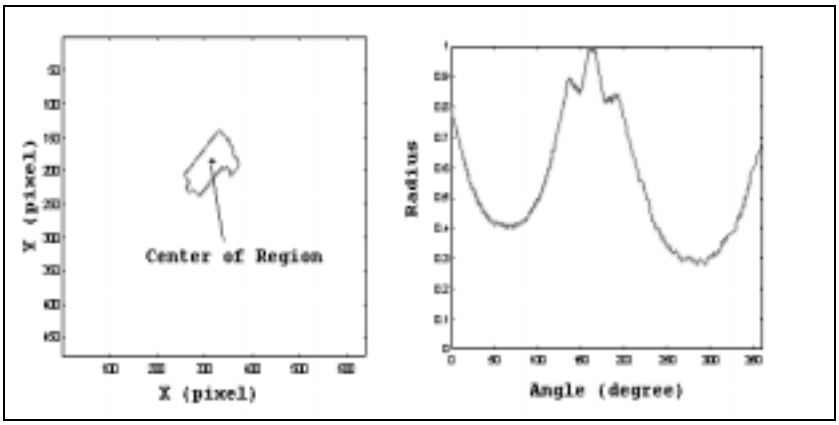

Figure 3-3. A Cylinder and its AOS Signature

It should be noted that the AOS technique can only used for a stationary object. The camera axis should pass through the object's center. AOS signature is orientation independent.

The AOS signature serves two purposes. Firstly, if the radii in the AOS signature were not normalized (i.e. all radii are divided by the maximum value), then the size of the object can be classified into different categories: large, medium and small. If the average radius is found large, it means all perimeter points are far from the center of the object. Of course, the objects must be in the same category such as square.

Secondly, if the radii were normalized, then the AOS signature can be used to recognize the shape of an object. In this case, small and large objects will be recognized as the same object as long as they belong to the same category such as square. Figure 3-4 and 3-5 show the signatures of an object before and after a rotation. The object is a square steel block with a trimmed corner. In the AOS signature plot, the "X" mark

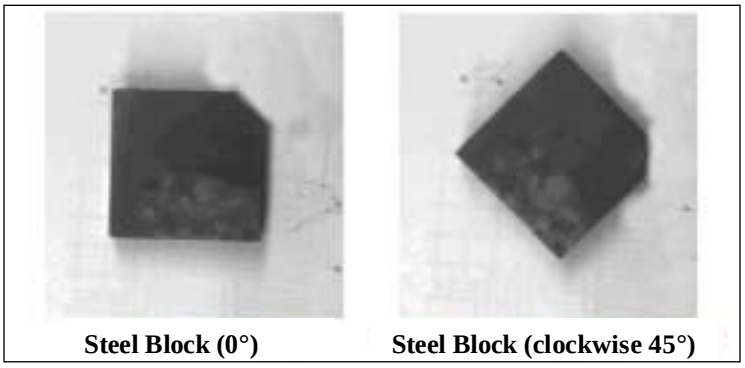

Figure 3-4. Two Different Angles of a Steel Block

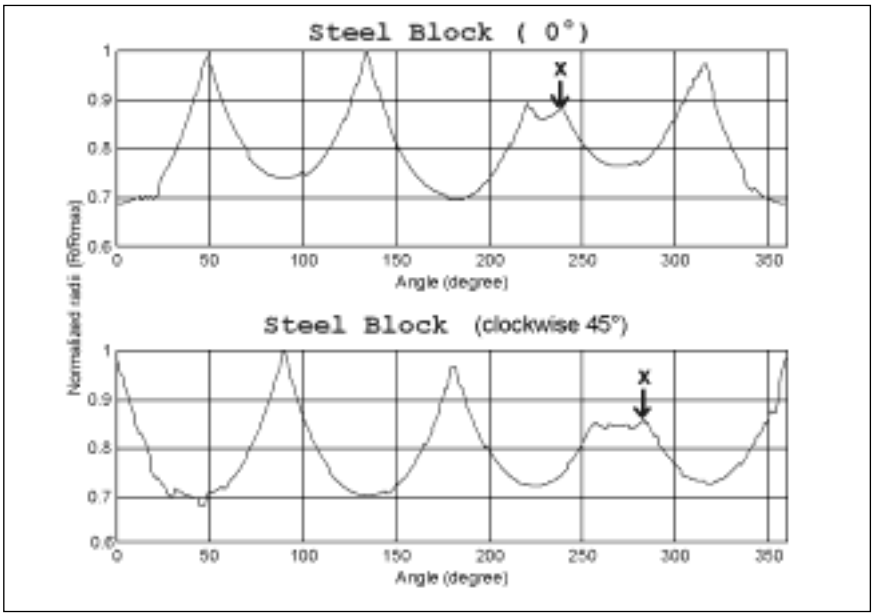

Figure 3-5. The AOS of the Steel Blocks for 0° and 45°

indicates a chamfer. If we shift the AOS signature of the rotated block by 45° forward, the two signatures will match perfectly. Figure 3-6 and 3-7 show two AOS signatures from different size of steel blocks. Apparently, those two signatures are almost the same

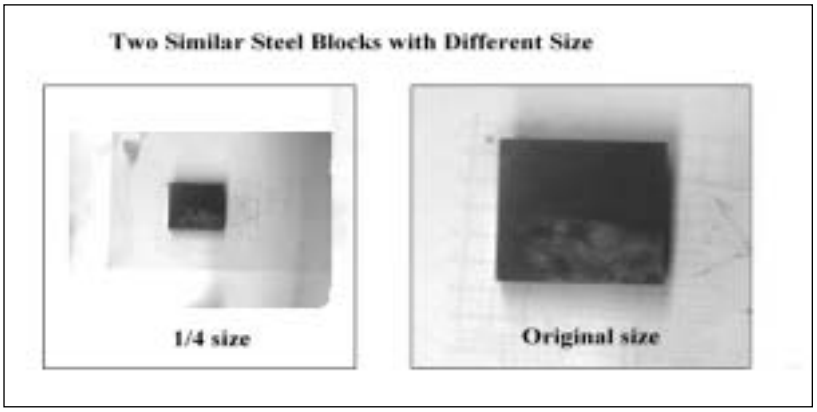

Figure 3-6. A Photo of Two Different Size of Steel Blocks

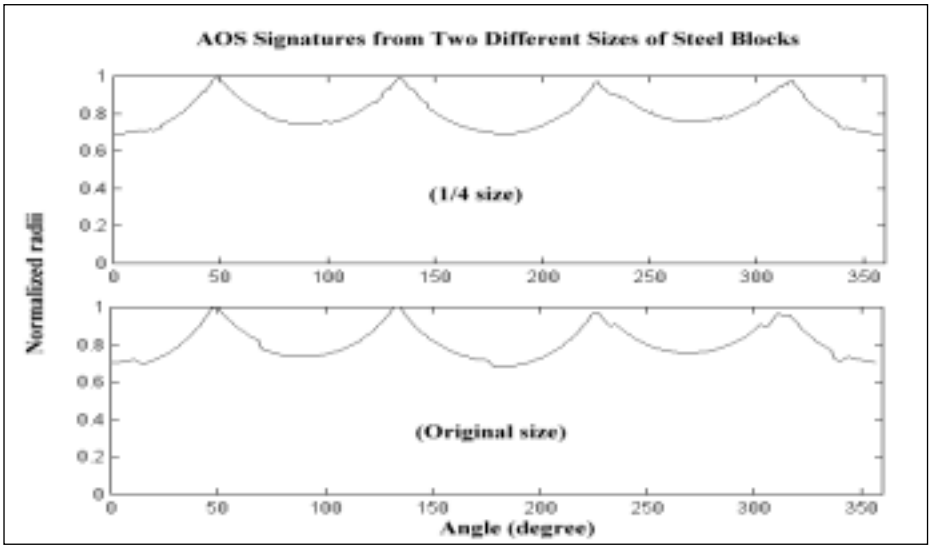

Figure 3-7. AOS Signatures of the Two Different Sizes of Steel Blocks

because they belong to the same category (i.e. square).

As mentioned in the previous chapter, the main goal of this research is to develop an intelligent inspection system. In order to achieve this goal, artificial intelligence is employed and described in the following sections.

3.7 Artificial Intelligence

Artificial intelligence (AI), a subfield of computer science concerns with the concepts and methods of symbolic inference by computer and symbolic knowledge representation for use in making inferences. AI can be seen as an attempt to model aspects of human thought on computers. Applications of AI are such as computer vision (building a system that can understand images as well as a human) and natural language processing (building a system that can understand and speak a human language as well as a human).

So far, many machines or systems are still run by human operators. Many simple things are easily understood by human, but very difficult for machines. Therefore, it is essential to give a machine some intelligence. The most popular and useful techniques in AI are fuzzy logic and neural networks.

Fuzzy Logic and Fuzzy Inference System

To find a fringe center by our current inspection system, the operator must specify some system parameters. It is certainly desirable to detect the fringe centers without human intervention. Fussy logic appears to be a possible candidate that can make this happen.

Fuzzy theories are based on the degree of uncertainty or truth. For instance, the statement, "John is 5'10" and he is tall" may or may not be true depending how we define tallness. In stead of using true or false, we may use words, such as very, somewhat or quite, to describe the degree of truth. For example, when we see black clouds in the sky, we will think that it might rain. How certain are we?

The traditional Boolean logic has only two values - true or false. In our daily life, most statements cannot be exactly true of false, and we assign the third value into our logic, such as "possible". When things are not clearly defined, the fuzziness is arisen. If we can not say "something must be wrong" or "what is the boundary between good and bad", fuzzy logic may be the best way to solve this situation [Rao V. and Rao H., 1993].

In Brule's article [Brule, 1994], Lukasiewicz described a 3-value logic, along with the mathematics in early 1900's. Later he explored four-valued logics, five-valued logics, and then declared that in principle there was nothing to prevent the derivation of an infinite-valued logic. In 1965, Dr. Lotif Zadeh [Zadeh, 1965] published "Fuzzy Sets" which described the mathematics of fuzzy set theory, and by extension fuzzy logic. He introduced it as a means to model the uncertainty of natural language. This theory proposed making the values False and True (or the membership function) operate over the range of real numbers [0.0 1.0]. In the above example, the truth-value of that statement may be about 0.8. By assigning a value (i.e. 5'10") to a so-called membership function "tall (5'10")" as shown in Figure 3-8, "tall" people can be well defined. The membership function can be a continuous curve or as simple as a set of some values.

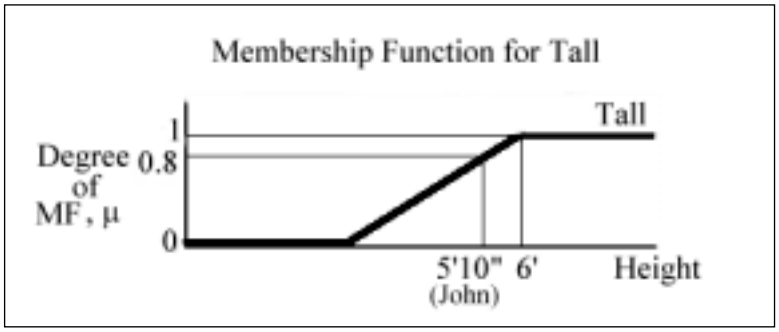

Figure 3-8. The Membership Function for Tall

Fuzzy logic is a superset of Boolean logic and dealing with the concept of partial truth -- truth values between "completely true" and "completely false". If the values are in crisp form, which is limited at 0 or 1, the fuzzy logic can be performed as the Boolean logic. When dealing with Fuzzy Logic, it is usually displayed as an ordered pair, such as (tall, 0.8) in the previous example. The first member of the ordered pair is a "candidate for conclusion" or "antecedent" in the set, and the second member is a value between 0 and 1, inclusive, called the "degree of membership" or "consequent" in the set.

Fuzzy Logic uses degree of truth to deal with a statement or proposition, which can be true to a certain degree. The degree of truth looks like a probability, but it is not quite the same. Probabilities for mutually exclusive events cannot add up to more than 1, but their fuzzy values could. However, fuzzy logic still behaves like probabilities, but have operations among several statements. The basic operation is union (or), intersection (and) and complement (not) and their definitions are:

truth(not x) = 1.0 - truth(x)

truth(x and y) = minimum (truth(x), truth(y))

truth(x or y) = maximum (truth(x), truth(y))

(There are other possible definitions for "and" and "or", e.g. using sum and product). If truth-values are restricted to 0 and 1 then these functions behave just like their Boolean counterparts. This is known as the "extension principle".

Fuzzy Inference System

With the definitions of labels and membership functions, a fuzzy inference system (FIS) can be established. FIS is very similar to the expert system. The difference is that its database is constructed by fuzzy rules. Fuzzy if-then rules or fuzzy conditional statements [Jang, 1993] are expressed as the form "*IF A THEN B*", where A and B are labels of fuzzy subsets characterized by appropriate membership functions.

The operations inside a FIS have four steps: fuzzify inputs, inference, composition, and defuzzification. According to different methods of composition, there are two types of fuzzy inference systems: Mamdani and Takagi-Sugeno Kang methods. An example describes the Mamdani method is:

If *road is straight*, then *speed is high*.

Where road and speed are fuzzy variables, straight and high are values that are characterized by membership functions. The antecedent and consequent parts are fuzzy equations.

The fuzzy inference process we have been referring to so far is known as Mamdani's fuzzy inference method. It is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani, which attempts to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from

experienced human operators. Mamdani-style inference, which is defined in the Fuzzy Logic Toolbox for Matlab, expects the output membership functions to be fuzzy sets. After the composition process, there is a fuzzy set for each output variable that needs defuzzification process.

Another form of fuzzy if-then rule, proposed by Takagi and Sugeno, has fuzzy sets involved only in antecedent part. By using this rule, we can describe the resistant force on a moving object as follows:

If *intensity value is high*, then *distance = function of intensity value*

Where, again, high in the antecedent part is a linguistic label characterized by a membership function. However, the consequent part is described by a nonfuzzy equation [Nomura, et. al., 1992] of the input variable, intensity value. It is possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. This is sometimes known as a singleton output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required to find the centroid of a two-dimensional shape. Rather than integrating across a continuously varying two-dimensional shape to find the centroid, we can just find the weighted average of a few data points. Sugeno systems support this kind of behavior.

In this section let us discuss the so-called Sugeno, or Takagi-Sugeno Kang method of fuzzy inference first introduced in 1985, which is similar to the Mamdani method in many respects. In fact, the first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The only distinctions are the fact that all output membership functions are singleton spikes, and the implication and

aggregation methods are fixed and can not be edited. The implication method is simply multiplication, and the aggregation operator just includes all of the singletons. The input for the defuzzification process is a fuzzy set and the output is a single number.

Here is an example of a FIS on finding the percentage of the tip according to the service and food of a restaurant. "Service" and "food" are defined as two input variables and "tip" is the output variable. Three membership functions (excellent, good and poor) are for "service" and two MFs (delicious and rancid) are for "food". "Tip" has two MFs (generous, average and bad). There are three rules for the FIS. They are:

the (tip If (service is poor) or (food is rancid) then (tip is cheap)
If (service is good) then (tip is average)
If (service is excellent) or (food is delicious) is generous)

In Figure 3-9, a graph demonstrates the operations of the tipper FIS.

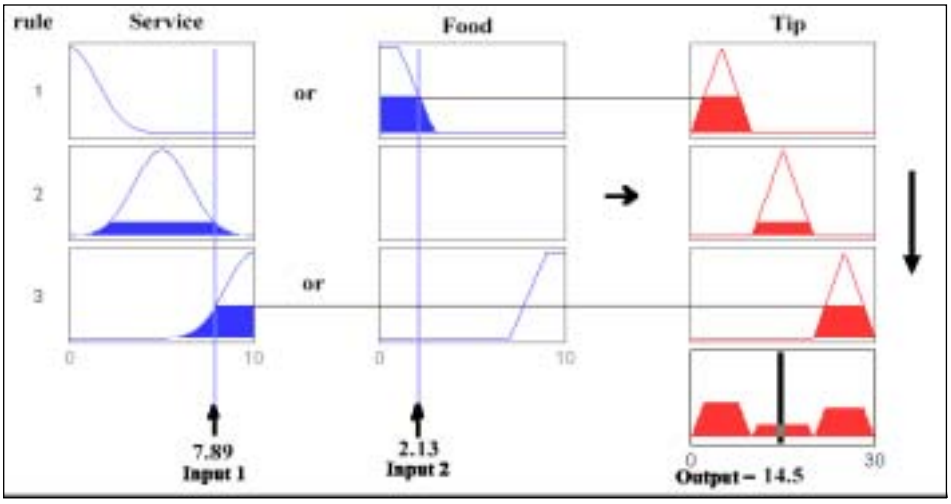

Figure 3-9. A FIS Flowchart for the Tipper Example

3.7.3 Neural Networks

It is easy for us to perform some tasks, such as recognition or classification, but they can be rather complex to modern computers. In order to make a machine (especially a computer) act like a human brain, artificial neural networks are used. A neural network is a processing device, either an algorithm, or actual hardware, whose design was inspired by the design and functioning of animal brains and components thereof.

There are many neurons (brain cells) inside a human brain. They use the axons connected to each other. A neuron can receive energy from other cells and emit a signal to its neighbors when the energy reaches a special level. Artificial neural networks (ANN) can be either hardware or software. Learning is the most important part of the neural networks. Neural networks learn from examples. Most neural networks have some sort of "training" rule whereby the weights of connections are adjusted based on presented patterns.

Neurons are often elementary non-linear signal processors (in the limit they are simple threshold discriminators). Another feature of ANN, which distinguishes them from other computing devices, is a high degree of interconnection, which allows a high degree of parallelism. Further, there is no idle memory containing data and programs, but rather each neuron is pre-programmed and continuously active. The term "neural net" should logically include biological neural networks, whose elementary structures are far more complicated than the mathematical models used for ANN. In principle, ANN can compute any computable function. For example, they can do everything a normal digital computer can do.

In practice, ANN is especially useful for classification and function approximation/mapping problems, which are tolerant of some imprecision. Almost any mapping between vector spaces can be approximated to arbitrary precision by feedforward ANN (which are the type most often used in practical applications) if we have enough data and enough computing resources.

To be more precise, a feedforward networks is usually designed with a single hidden layer. Such networks can also be trained as statistically consistent estimators of derivatives of regression functions and quantities of the conditional noise distribution. Feedforward networks with a single hidden layer which use threshold or sigmoid activation functions are universally consistent estimators of binary classifications under similar assumptions.

Usually neural networks are classified by their learning methods: supervised and unsupervised. The former is taught with desired outputs, and the later can classified the output the assigned categories. The Backpropagation (BP) NNs, as shown in Figure 3-10

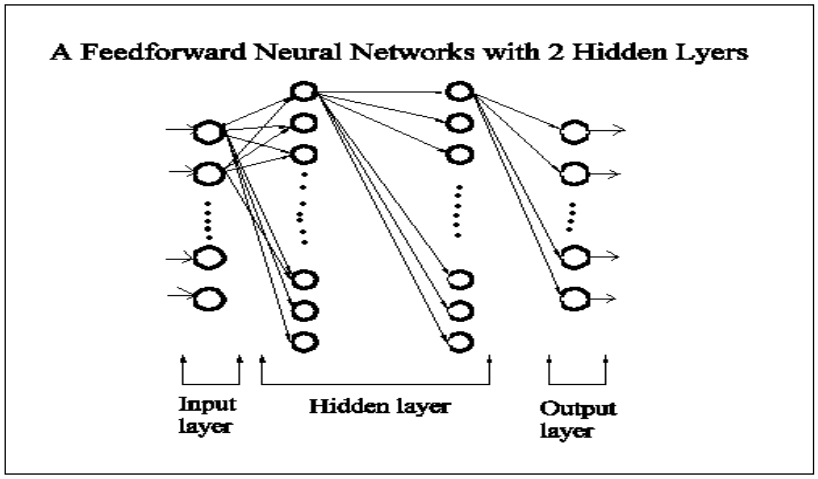

Figure 3-10. A Backpropagation Neural Network

is the most popular supervised learning method. Kohonen Self-organization map (SOM) is a typical unsupervised learning method.

In 1989, ALVINN (Autonomous Land Vehicle in Neural Networks) developed by Pomerleau [Pomerleau, 1989] set a world record for autonomous navigation distance. After the road condition is captured by a camera, the image is then divided into 30x32 small elements. Each element is the averaged intensity value of its 16 neighbor pixels. There are 30 different parameters to control the vehicle. By feeding the images to a neural network (30x32 input nodes), the mapped 30 steering parameters were used to drive that car automatically.

3.8 Calibration

Before an optical system can be used in a product line, the system needs to be calibrated first. If a relationship between the lens and the object position can be found, a look-up table can be established to find the calibration factor. The lens position (the distance to the object) and the camera orientation are the crucial factors in calibration. Furthermore, it is important to keep the camera's properties (i.e. position and orientation) constant all the time. However, the properties might change if someone inadvertently touches and repositions the camera.

Lens Properties

The function of a camera lens is to form an image. The most fundamental characteristic of a lens is its focal length [Tissue, 1996]. Generally speaking, this is the distance from the center of the lens to the image plane. The longer the focal length is, the

larger the image of an object will be. Focal length and image size is therefore directly proportional [Craven, 1982]. For example, if we replace a lens of 50mm focal length by that of 100mm focal length, the latter image will be exactly twice the size of the former. In addition, if we move the object to close to the lens, the size of the image will be bigger. The surface curvature of lens, however, might be slightly distorted due to the manufacturing defects.

3.9 Design of a New Automated Inspection System

Three modules are developed. The first module is the algorithm for a fringe center locator (FCL) using fuzzy inference. The second module is the object recognition algorithm, and the third one is the auto-calibration procedure.

In order to meet the current and future needs, an inspection system must be intelligent and automated. The system currently used in our Robotics and Optical Inspection Laboratory is manually operated. The fringe projection technique is the core of the system, but the program cannot locate those fringes by itself without human intervention. The user must specify some parameter values prior to running the computer program to locate fringe centers. Since such a task is relatively easy for a human operator, it should be possible for an intelligent machine to do the same. In this work, a fuzzy inference system is developed to function as a new fringe center locator (FCL).

Besides the FCL, some object recognition algorithms are also essential. A simple and fast recognition algorithm is developed in the research. Before an inspection system can be put to work in a product line, calibration should be done first. Even so, as time

goes by, the imaging error may become bigger and bigger. Therefore, it is desirable to develop a self-recalibration procedure.

Algorithm for Fringe Center Locator by Means of Fuzzy Inference

The intensity values across a fringe are analogous to a Gaussian distribution [Lin, et. al., 1990]. When scanning the region of interest (ROI), all fringe centers can be accurately determined using our in-house developed algorithm provided that every scanned fringe is a Gaussian distribution alike. This will require at least five pixels (picture elements) per fringe, which usually has the highest intensity value in the middle, and the value gradually decreased toward each of the two ends, as shown in Fig 3-11.

Before using fuzzy logic rules, the five pixels within a fringe should be normalized (i.e. by dividing all intensity values by the maximum value). Afterward, each pixel is

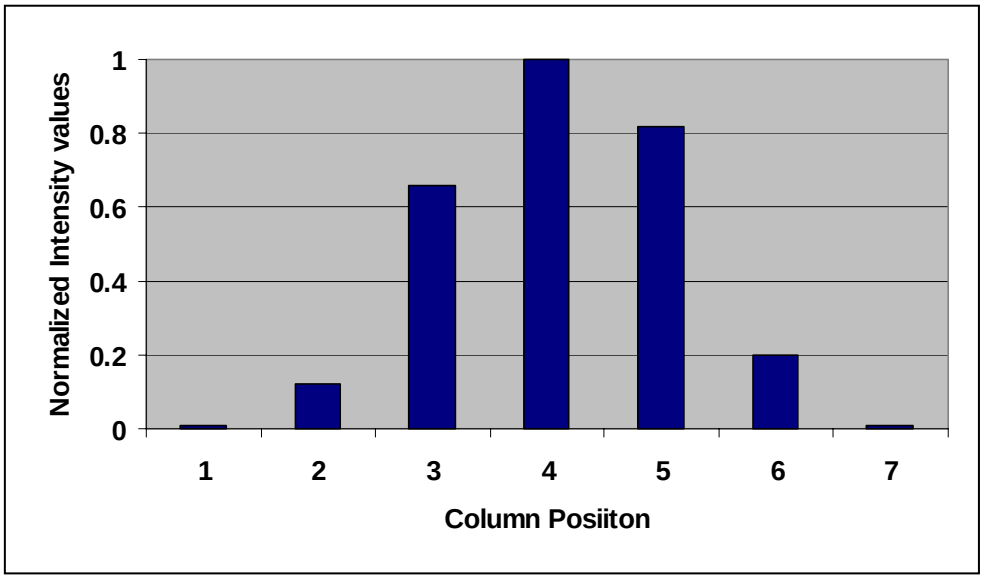

Figure 3-11. Intensity Distribution of 5 Pixels across a Fringe (pixel 1 and 7 are background)

classified into dark, light and bright categories. For example, one of fuzzy rules is stated as follows:

IF (point 1 is dark) **and** (point 2 is light) **and** (point 3 is bright) **and** (point 4 is light) **and** (point 5 is dark) **THEN** (Curve is Gaussian-like)

Usually the exact fringe center is located in the neighborhood of the third point (i.e. between points 2 and 3, or between points 3 and 4). However, in some cases, the center could be located beyond the expected neighborhood. In this case, the distribution curve will be skewed, which will require different rules in the fuzzy inference system.

Each image file can be considered as a matrix. For example, a 640x480 pixels image has a matrix consisting of 480 rows and 640 columns. The intensity values in this matrix are between 0 and 255 for an 8-bit gray-scale image. As mentioned in section 3.7.2, the output of a Sugeno fuzzy inference system is a singleton spike (i.e. a single value) which is suitable for our fuzzy rules. Thus, a Sugeno fuzzy inference system is employed to develop our algorithms. The outputs for the FIS are 0, 1, 2 and 5 for non-center, upward-edge, downward-edge and center, respectively, while 3 and 4 are reserved for the future development. The first step is to translate the intensity values into the fuzzy output values.

Procedure for the Fuzzy-Logic Based Fringe Center Locator:

The size of a ROI is assumed m by n (m rows by n columns). Point P(i,j) is a point inside the ROI, whose intensity value is I(i,j). After loading the designed fuzzy rule base, the following steps illustrate the procedure for FCL. The flowchart is shown in Figure 3-

12.

Step 0. Let i = 0 and j = 0 (upper left corner of the ROI)

Step 1. Do loop from steps 2 to 7 while j <= m

Step 2. Do loop from steps 3 to 6 while i <= n

Step 3. Choose 5 continuous pixels whose intensity values form a vector V = [I(i-2,j), I(i-1,j), I(i,j), I(i+1,j), I(i+2,j)].

Step 4. Normalized the vector through dividing each element by the maximum value of those 5 values.

Step 5. Apply fuzzy interface to the vector. The output is 0, 1, 2, or 5.
A new fringe center matrix (FC_{mxn}) containing only these four values
is generated.

Step 6. Let i = i + 1, go back to step 2

Step 7. Let j = j + 1, go back to step 1

Step 8. Set up a new matrix (FC_1) with the same size of FC_{mxn}. Scan the matrix FC. As soon as a row vector such as [2 5 1] is found anytime, a row vector [0 1 0] is output to the FC_1 in the related position. Note that Matrix FC_1 is the new matrix contains only 1s and 0s.

Step 9. From the FC_1, find the column positions of 1s to form the matrix of fringe Centers row by row.

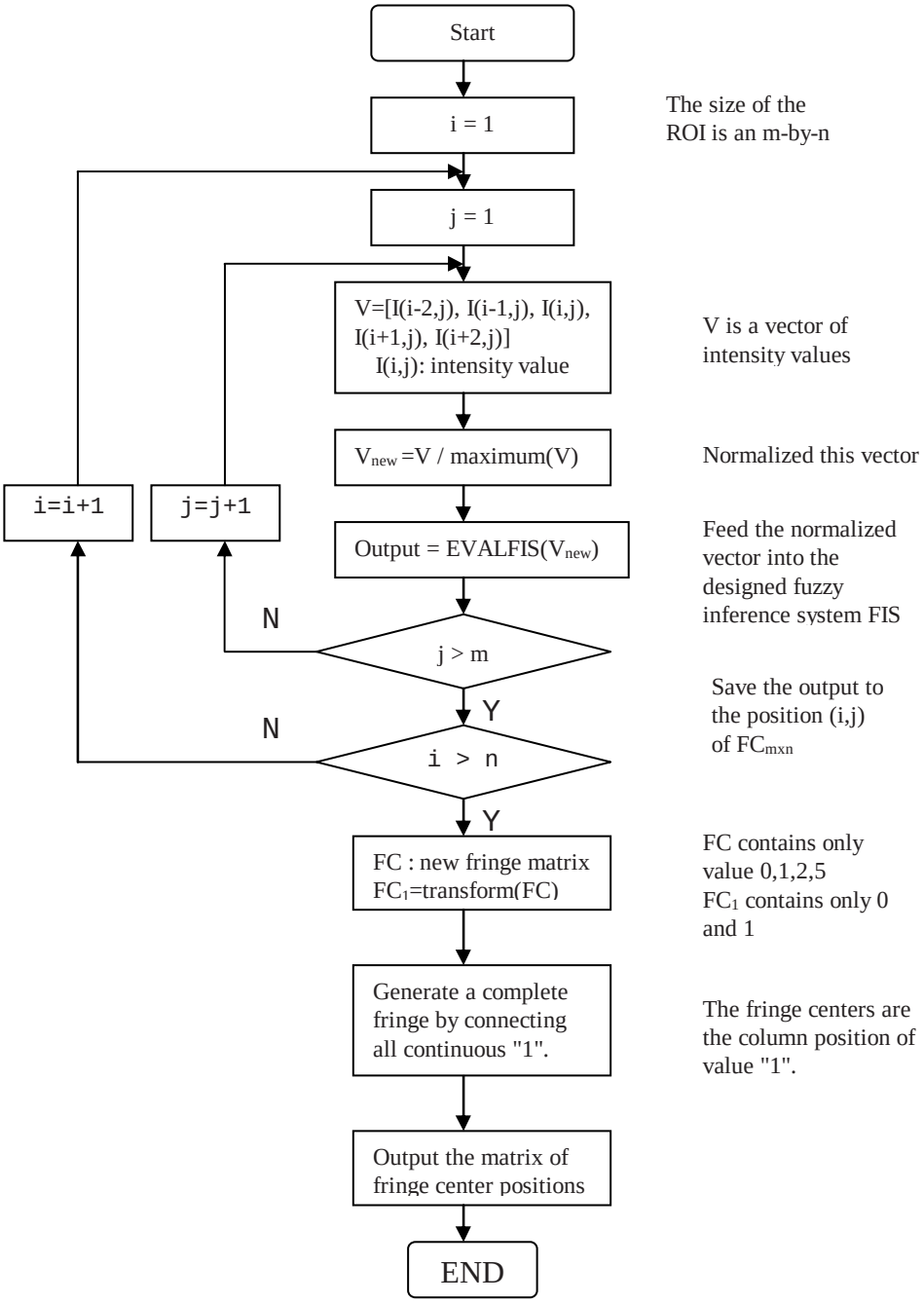

Figure 3-12. The Flowchart of the Fuzzy-Logic Fringe Center Locator

The following example illustrates the procedure. A fuzzy interface system is employed for the entire image, but only one fringe with three rows is selected for the purpose of illustration. In Eq. (3.1) shows the matrix of intensity values is converted into one that contains only 0, 1, 2, and 5.

$$\begin{bmatrix} 0 & 75 & 150 & 235 & 180 & 90 \\ 23 & 130 & 220 & 140 & 43 & 0 \\ 15 & 115 & 200 & 100 & 7 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 2 & 5 & 1 & 0 \\ 0 & 2 & 5 & 1 & 0 & 0 \\ 0 & 2 & 5 & 1 & 0 & 0 \end{bmatrix} \qquad (3.1)$$

Afterward, [2 5 1] are converted into [0 1 0], while others are converted into [0 0 0]. Thus, a temporary matrix with values of only 0s and 1s are generated as shown in Eq.(3.2).

$$\begin{bmatrix} 0 & 0 & 2 & 5 & 1 & 0 \\ 0 & 2 & 5 & 1 & 0 & 0 \\ 0 & 2 & 5 & 1 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (3.2)$$

Based on one-pixel resolution, the approximate fringe centers are at the column position indicated with the value of 1. As can be seen in Eq.(3.3), the fringe centers are located in columns 4, 3 and 3 for the 1st , 2nd and 3rd row, respectively.

$$\begin{array}{l} \textit{column}\ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad \textit{fringe center} \\ \textit{row}\,1 \\ \textit{row}\,2 \\ \textit{row}\,3 \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix} \qquad (3.3)$$

Sometimes a fringe is discontinuous due to a drastically change in geometry or simply system noise. Eq.(3.4) shows a matrix contains 3 fringes. The 3rd row of the 3rd

fringe center is missing. Apparently, we need to develop another procedure to reconstruct the discontinued fringes.

$$\begin{array}{l} \textit{column}\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12 \\ \textit{row}\ 1 \\ \textit{row}\ 2 \\ \textit{row}\ 3 \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 8 & 12 \\ 3 & 7 & 12 \\ 3 & 7 & 0 \end{bmatrix} \quad \textit{output matrix} \qquad (3.4)$$

Procedure for Reconstructing Discontinued Fringes

1. Find the row near center (i.e. 2nd row of Eq.(3.4)) with the maximum number of fringes (i.e. 3 fringes).
2. There are two loops in this step:

   2-1. Loop1: from the chosen row (m) to the 1st row

   Let the temporary (m-1)th row vector equal the m-th row vector, then choose 1st element from the row (m-1) to compare with all m-th vector's element. Put the element into the position with minimum difference (<= 2). Repeat the above operation until all elements from (m-1)th vector are used.

   2-2. Loop2: from the chosen row (m) to the last row

   Repeat the same procedure as in loop 1

The height of all fringes is m rows, and there are two fringes in this example. After regrouping the matrix, a new matrix consisting continuous fringes is reconstructed and shown in Eq.(3.5).

$$\underset{old\ matrix}{\begin{bmatrix} 4 & 8 & 12 \\ 3 & 7 & 12 \\ 3 & 7 & 0 \end{bmatrix}} \rightarrow \underset{new\ matrix}{\begin{bmatrix} 4 & 8 & 12 \\ 3 & 7 & 12 \\ 3 & 7 & 12 \end{bmatrix}} \qquad (3.5)$$

Finding the Exact Fringe Centers with Subpixel Resolution

In the calculation of the sub-pixel fringe center, two methods were used. They are the bisection numerical method and the analytical method by symbolically differentiating the Gaussian distribution function. The solution thus obtained through the analytical method is closed-form.

The Bisection method can find the root of f'(x) (i.e. the solution for f'(x) = 0, where f(x) represents a convolution sum of intensity values [Lin and Parvin, 1990] Gaussian distribution function). In this method, all pixels across a fringe are used, and it can accept any width of a fringe. This method, however, is a little time consuming due to many iterations, and the accuracy is not as good as that obtained from the second method.

The second method has the following closed-form solution (Eq. (3.6)).

$$f(I_i) = \frac{e^{-0.5}(I_1 - I_{-1}) + 2e^{-2}(I_2 - I_{-2})}{I_0 - 3e^{-2}(I_2 - I_{-2})} \qquad (3.6)$$

This method instantly finds the sub-pixel fringe center. However, the fringe width is desirable to be 5 pixels or more, but no less than 3 pixels. If the fringe width is more than 5-pixel, only 5 pixels near the approximate fringe center are selected. Usually, an

inspection system can be arranged to produce at least 5 pixels.

Algorithms for Object Recognition

In automating the optical inspection process, it is desired to the capability of object recognition. The two advantages for adding the object recognition module to the inspection system are.

(1) Let the system recognize what the current object is, and retrieve its property from a database.

(2) Helps set up the ROI for the 3-D geometry measurement of a chosen object.

Image Threshold

The image files are all in gray-scale mode (i.e. black and white). Image threshold is necessary to distinguish the object from its background, and the threshold value is usually preselected based on experience [Parker, 1991]. The Laplacian or Sobel edge enhancement filters were not used in the research because the detected edges may be distorted. Imagine threshold is a quick and accurate way to find the edge of an object if the background is simple. A single value of threshold is desired in this research. The rule is as follows.

$$Intensity\ value,\ I = \begin{cases} 1, & if\ \ I > threshold \\ 0, & if\ \ I \le threshold \end{cases}$$

Object Recognition by Means of Neural Network Mapping

While analyzing the image of an object, some descriptors can be extracted. The most useful descriptors are area, perimeter, compactness, number of corners, and radii (maximum, minimum and average). By studying these descriptors, a relationship between the shape of an object and the descriptors can be found. A backpropagation neural network (BPNN) is utilized in this research due to its simple training algorithms and fast mapping.

After a preliminary study, three descriptors are chosen as the input variables of the neural network. They are the area, perimeter and elongation ratio. Each object is assigned by a special coded number. For instance, the rectangle, circle, square and hexagon are coded as 1, 2, 3 and 4, respectively. Thus, a BPNN can be designed to have three input neurons (or nodes) and one output neuron.

Object Recognition by Means of Angle-of-Sight (AOS) Signatures

In order to increase the accuracy of object recognition, an addition recognition algorithm is needed. AOS signature provides useful information of the object, such as the maximum, minimum and average radii as functions of angles. Figure 3-13 shows an example of AOS signature for a rectangle. In fact, the four corners can be easily observed. If the corners are located at θ_i (i = 1,2,3 and 4), the distances between two adjacent corners are defined as $\Delta\theta_j$ (inter-corner distance, ICD), where $\Delta\theta_j = |\theta_{j+1} - \theta_j|$ and j=1,2 and 3, where $\Delta\theta_4$ is calculated by $\Delta\theta_4 = (360° - \theta_4) + (\theta_1 - 0°)$. In this example, the four ICDs are (72.74°, 105.33°, 74.97°, 106.94°), and the average, maximum and minimum radii are 65.79, 83.33 and 48.29 pixels, respectively. The object is recognized as a rectangle by the fact that there are four corners and $\Delta\theta_1 = \Delta\theta_3$ and $\Delta\theta_2 = \Delta\theta_4$.

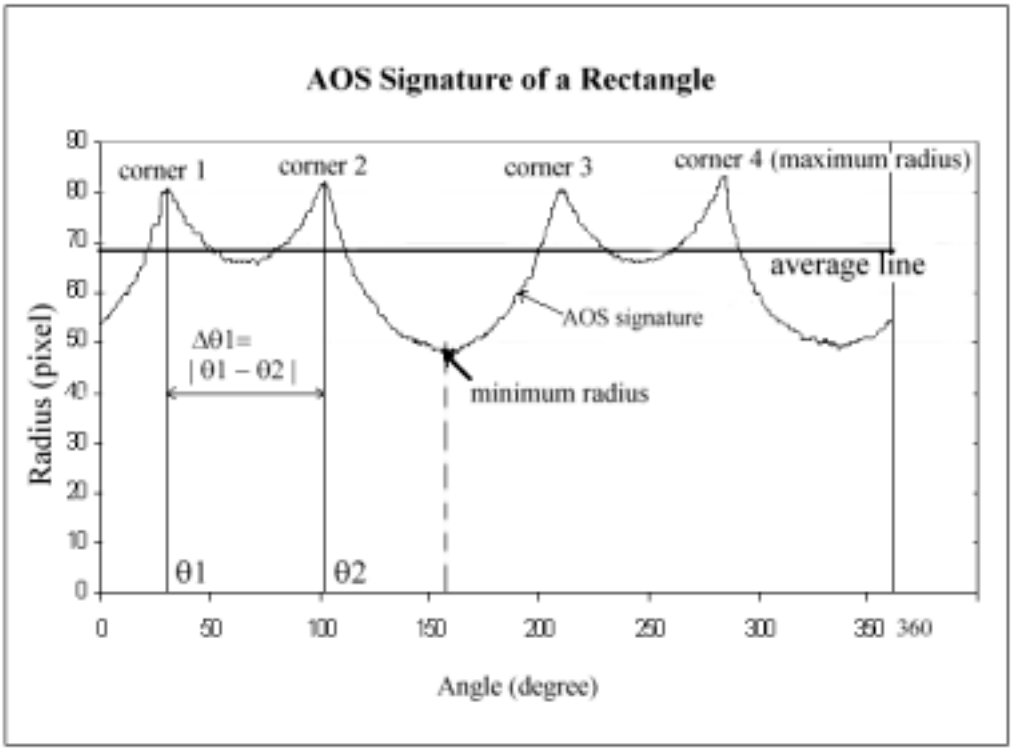

Figure 3-13. Some Properties of an AOS Signature

Furthermore, a database or a look-up table, such as Table 4-6, should be established. In the look-up table, each column indicates the shape of the part along with the average, maximum, minimum radii, number of corners and the distances between corners.

When an object is first inspected, its properties are extracted from the object's AOS signature, and compared with those from the look-up table. Four steps are needed to complete this recognition procedure.

Step 1. Compare the average radius.

Step 2. Compare the maximum and the minimum radii.

Step 3. Calculate the ICDs between corners (the distance between peaks in the AOS signature) and find the matched data set.

Step 4. Determine the number of corners.

By comparing the object's features determined by the above steps, an object can be recognized by matching the features with those stored in a data set in which every regular-shape object is assigned by a reference part number.

ROI Setting via Object Recognition

There are many ways to set up an ROI (Region of Interest) for an optical system. The simplest shape of ROI is a rectangular. When the object is recognized, its size and shape are determined. By using the perimeter's maximum and minimum y values, the boundaries of a ROI in the vertical direction can be set. Likewise, the boundaries of a ROI in the horizontal direction can be set by using the maximum and minimum x values. Another method is to use the exacted contour of the object. However, it is more difficult to do.

Algorithms for the New Calibration Procedure

As mentioned earlier, the calibration is much affected by the position and the angle of the camera. A study containing two parts is conducted in order to develop a new calibration procedure. The first part is to find the calibration factor map, which is related to the object’s position in the image viewed by the camera. The second part is to find a way to detect the camera’s yaw and pitch angles more easily. When facing the camera from the front end of the platform, the yaw angle is defined as positive when camera moves to the left, and negative when it moves to the right. Likewise, the pitch angle is defined as positive when the camera moves upward, and negative when it moves

downward. Rotation about the camera's axis (known as roll motion) is unlikely, and therefore is not considered in this study. Part I and Part II are describe as follows.

- **Part I - Generation of Calibration Map (Algorithm I)**

System calibration is a very essential procedure for any optical inspection system. In our traditional calibrating procedure, the calibration factor was determined manually by the user. In this proposed new calibration procedure, a mapping table is generated in advance. When an image is captured, the calibration factors (CF) are determined according to the position of the part in the image. Obviously, the center of the object is not always at the center of the image. When the object deviates a little from the image center, the CF should be changed accordingly.

Figure 3-14 shows that the calibration targets (circles) are placed at different

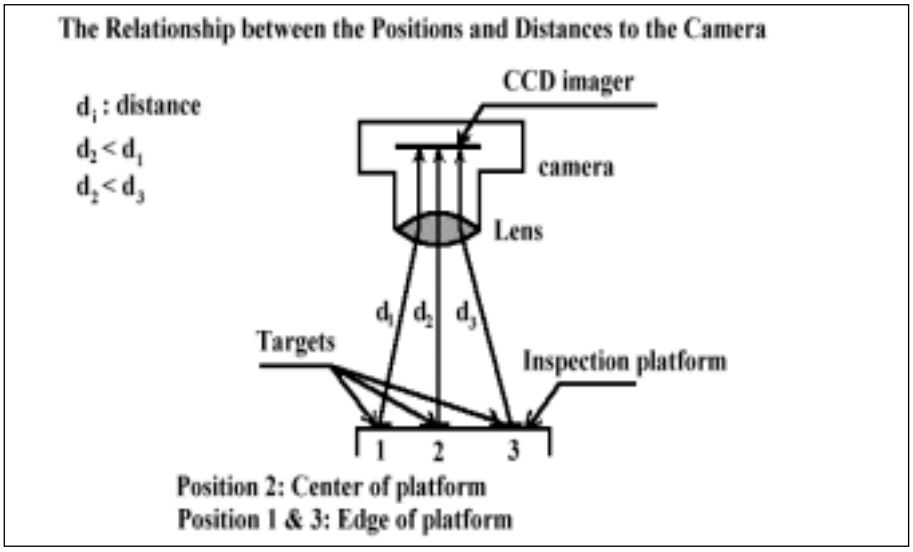

Figure 3-14. Distortion of Camera's Lens

positions on the inspection platform, where distance d_2 along the centerline is the shortest. Because the size of the object in the CCD imager is reversibly proportional to the

distance, the size of the calibration circle is the largest at the center. Furthermore, the areas of circles viewed at different distances are different as well.

After the system is completely set up, the camera should be kept perpendicular to the inspection platform. A series of circles are utilized here as calculation targets. It is not difficult to find the relationship between the area (or perimeter) of a circle and the calculated radius. The area of a circle can be calculated by Area = □* $(R_I / 2)^2$ pixel², and the radius $R_i = 2 * (\text{Area} / \square)^{1/2}$ pixels. The exact radius of a known circle (R_r) is used. Therefore, the calibration factor can be defined as R_r / R_i (inch/pixel).

When the image of the calibration target is captured, the circle's centroid (x_c,y_c) and the CF can be found. In fact, we can plot a calibration map in three dimensions (i.e, x_c, y_c, and CF). In other words, the calibration factor (CF) can be mapped by a function f.

$$CF = f(x_{obj} , y_{obj})$$

When an object is placed on a platform, the object's centroid (x_{obj}, y_{obj}) should be arranged to be coincident with the image center. Then the object's CF is calculated and used for an inspection later. In order to establish the calibration map, two different methods are used. The first method was done by a Matlab program. Here only a 5x5 matrix and a 7x7 matrix are considered. The calibration circle was initially placed at the upper left corner (0,0) and the image was captured and saved. Then the circle was moved to the next row until it touches the right edge of the platform (or edge of the screen). A total of 25 or 49 points are collected row by row. A dorm-like surface can be generated, in which the base consists of 640x480 pixels (same as the resolution of VGA), and the z direction represents the calibration map (See Figure 3-15). The advantage of this method is that all calculations are in the same Matlab environment.

The second method uses commercial software called Inspector 2.1 (developed by Matrox). Twenty-five same size circles are printed on a white paper as shown in Figure 3-

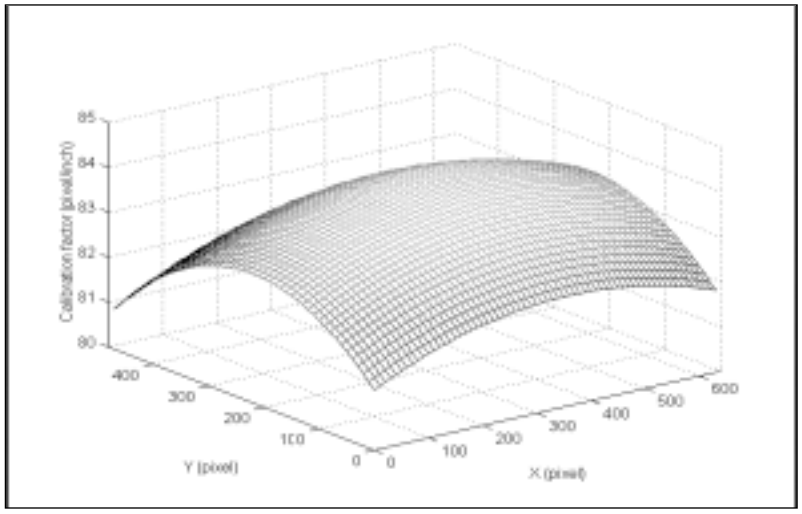

Figure 3-15. A Typical Calibration Map

16. By using the Autocad software, the generated circles are guaranteed to be perfectly round. We placed a template under the camera, and captured all circles in the same image. Then, using the "blob analysis" tool in the Inspector, the centroids and CFs (converted from the area of the circle) can be obtained and, a 5x5 calibration map can be

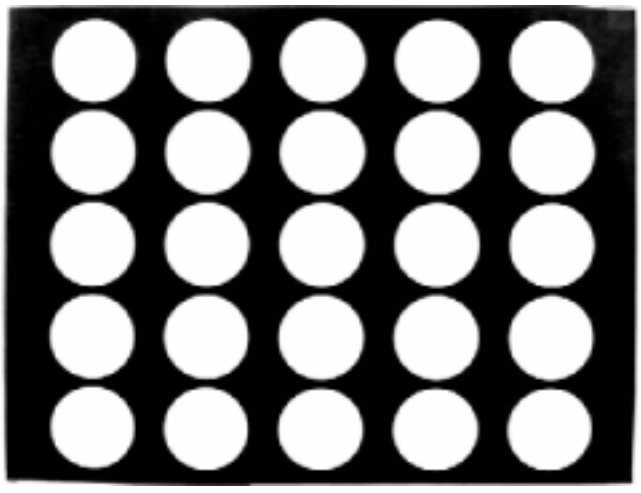

Figure 3-16. The Printout Calibration Target with 25 Circles on it

established. This method is much faster than the first one. However, it is an external program, and separated from our main program in Matlab. In other words, the Inspector and the Matlab software cannot communicate.

Since those 25 points do not cover the entire surface of the platform, a surface fitting is necessary. In the beginning, a simple neural network (BPNN) was used to perform the surface fitting. Before a network can be used, we need to collect enough data to train the BPNN. But, it is time consuming and incompatible with Part II development (camera's viewing angle detection) which is to be described in the following section. When the angle of the camera changed, and is detected from the part II, the calibration map should be regenerated. In order to regenerate quickly the map, we fit the map by a second-order polynomial in x and y (i.e. $z = ax^2 + bxy + cy^2 + dx + ey + f$).

- **PART II - Detection of Camera's Viewing Angle (Algorithm II)**

The optical inspection system is designed to work in a production line that is usually noisy and vibratory. After a long period of time, the camera may turn loose. Since the camera angle is important to the measuring accuracy, an algorithm is developed in order to detect the camera's angular position at any time. If the position is within the allowable range, the estimated angle through neural network mapping is entered the program to adjust the calibration factor. If the position is out of range ($> +5°$ or $< -5°$), the operator should adjust the camera to its zero position. Figure 3-17 shows the various camera's viewing angles. Two possible methods to find the camera's viewing angle are considered here. The first one is simple, and requiring fewer calculations than the second one. By examining Fig. 3-16, the viewing angle □, can be determined by □ = $\tan^{-1}(S_i / d)$.

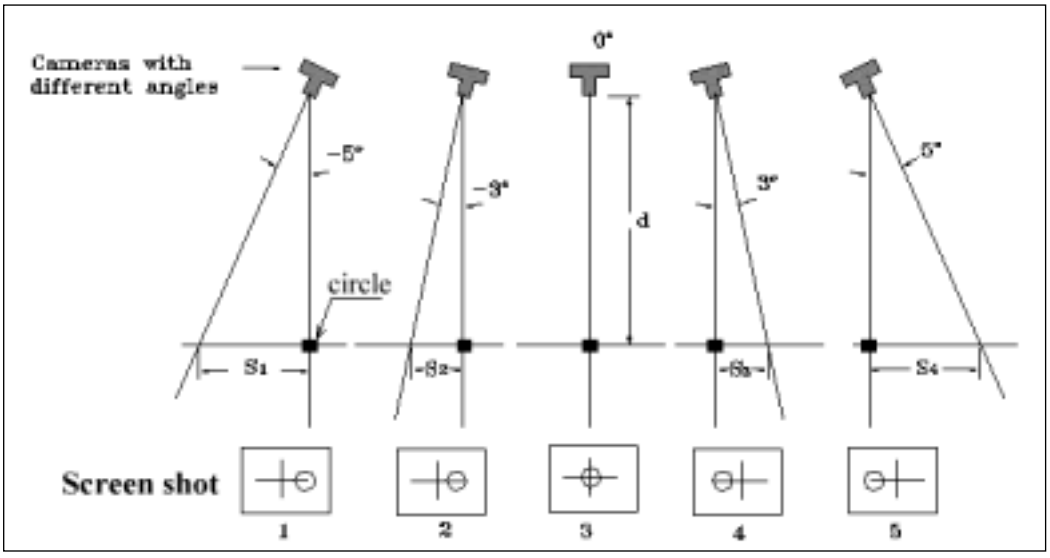

Figure 3-17. Screen Shots of the Circle with Different Viewing Angles

A circle can be placed at the pre-marked center, which is the same as the screen center. Note that S_i is in pixels, and can be converted into inch or mm using the calibration factor.

The second method is more complicated. A new map similar to the calibration map is prepared for each special camera angle (yaw angle = -3°, 0° and 3°) as shown in Figure 3-18. A backpropagation neural network was used for training. Each network was trained for a specific angle. The input node is the position of a point, and the output neurons are the CF values. When a circle is inspected, its centroid (x_c,y_c) and CF can be obtained. By entering the (x_c,y_c) values into the pretrained five NNs, five calibration factors (CF_{nn}) can be obtained. The viewing angle is determined by finding the minimum value of those five $|CF_{nn} - CF|$.

However, these sets of NNs were very difficult to train. It took a lot of time to

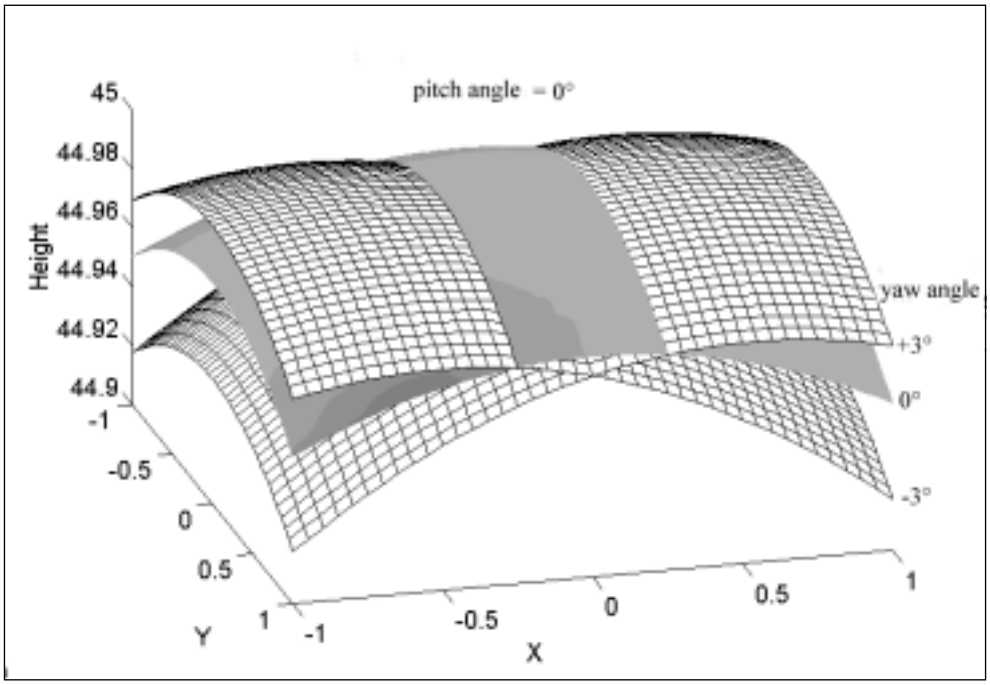

Figure 3-18. Calibration Maps with Different Yaw Angles Prepared for NN Training

prepare enough calibration maps with different viewing angles. The design and test of this complicated neural network set (5 NNs) are shown in the next chapter.

- **PART III- Combination of the Two Algorithms**

Figure 3-19 shows a flowchart of the combination of the two calibration algorithms applied to the inspection system.

3.10 The Automated Inspection System

When the aforementioned calibration is completed, the system should have at least the following capabilities- Fringe center locator (FCL) by means of fuzzy inference

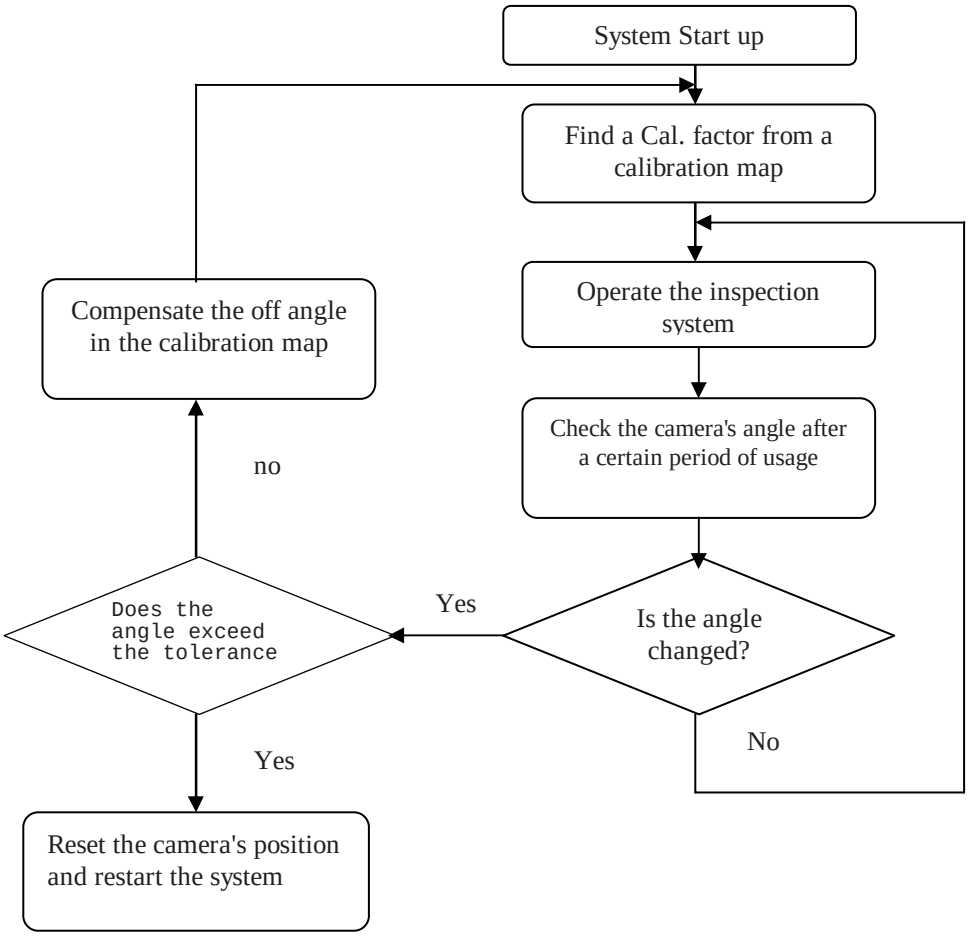

Figure 3-19. The Flowchart of the New Calibration Procedure

system (FIS), object recognition with neural networks or Angle-of-Sight signatures, and an automated calibration. The framework of this system is described as follows.

When an object is placed on an inspection platform, it is usually near the center of the viewing area (the area of platform is larger than the viewing area). An image without fringes on it is first captured. After the features of the object have been extracted, the object can be recognized through NN mapping and Angle-of-Sight signatures. If necessary, the object will be inspected in 3-D. In this case, fringes will be projected onto the object. The flowchart of the new optical inspection system is shown in Figure 3-20.

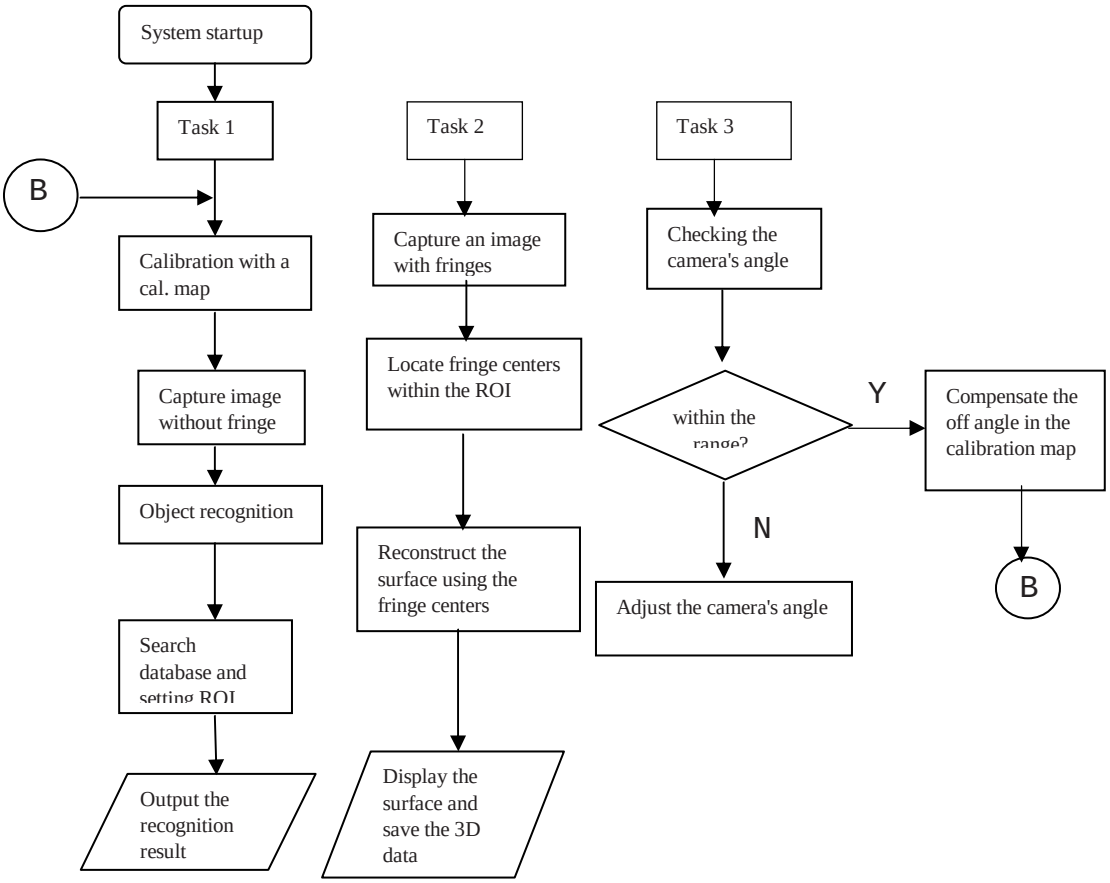

Figure 3-20. The Flowchart of the New Optical Inspection System

CHAPTER IV

DESCRIPTION OF EXPERIMENTS

As mentioned earlier, three new algorithms were developed in this work. This chapter describes how the experiments were designed and conducted to verify these three algorithms.

The first section describes the application of fuzzy logic (FL). The new inspection system will be tested and the results will be compared between the new and old systems. The second section is about the application of object recognition algorithms. There are two recognition algorithms employed in the new system. The first one is by means of neural network mapping, and the second one is via the angle-of-sight (AOS) signatures. The results from using these two different algorithms are reported separately.

The last section shows the results of using the new calibration procedure. A calibration map was generated and modified according to the estimated camera viewing angles. At the end of this chapter, some discussion will be presented.

4.1 Fringe Center Locator (FCL)

The most important part of our 3-D geometry inspection system is to locate optical fringe centers. The algorithm for the fringe center locator contains two steps. The first step is to find the approximate fringe center based on one-pixel resolution. Afterward, sub-pixel resolution is employed to determine the exact fringe center locations. The following sections describe the new experiments of locating fringe centers.

4.1.1 Models Used in FCL Testing

There are three different models used in this experiment. Two of them are made of clay, which the third one is made of steel with three step increases. The surface of the clay is not very reflective, which minimizes efforts in dealing with lighting. The first clay model is irregular in shape, but smooth in surface, as shown in Figure 4-1. The reason for choosing the clay model is that it contains two distinct regions representing different shapes or depths. The left one is somewhat flat, while the right one is more curved. Figure 4-2 shows the second clay model, which is shell-like. It contains a highly curved

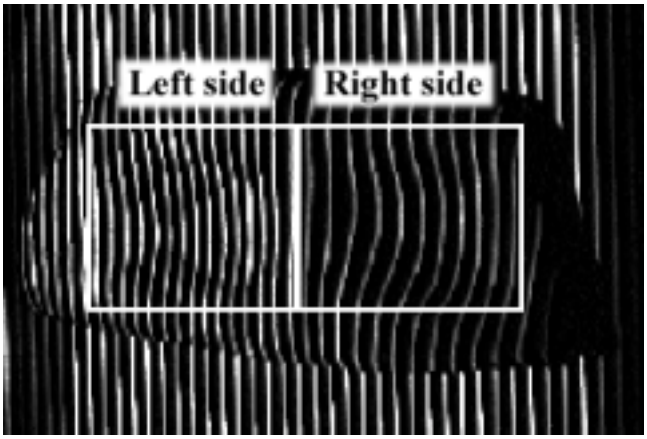

Figure 4-1. A Clay Model (model 1) with Projected Fringes

surface, very similar to the surface of a cylinder. This model was designed to test the new fuzzy logic algorithm detecting the fringe centers when fringe lines are very curved.

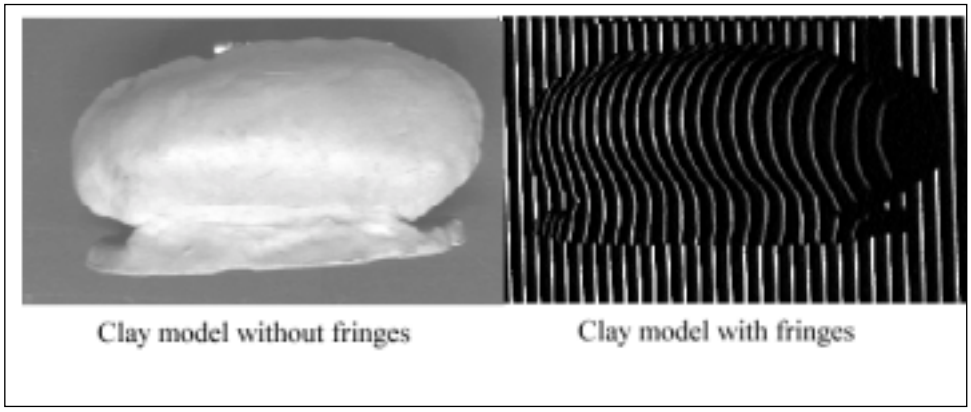

Figure 4-2. The Second Clay Model (model 2) with Very Curved Surface

The last model is a block of steel as shown in Figure 4-3. The surface is flat, and therefore the fringes are parallel straight lines. There are three step increases (0.5", 0.5",

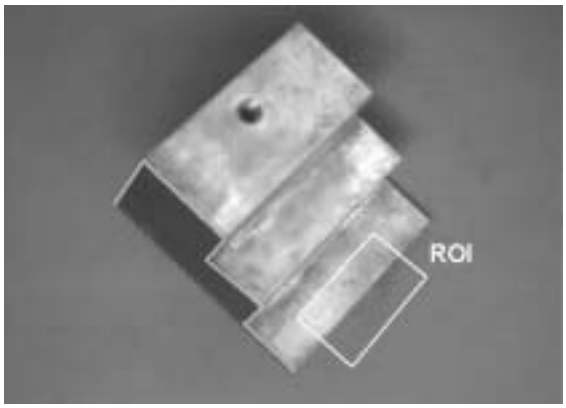

Figure 4-3. A Steel Block with 3 Step Increases.

0.1") made in this model. Between steps, fringes are shifted or broken. Thus, the shift has to do with the step increase. The lowest step (0.1") was chosen to test the ability of the new FCL in recovering the broken fringes.

4.1.2 Designing the Experiments

In order to verify the performance and accuracy of the new fuzzy-logic based FCL, four different experiments were comparisons were designed to verify that new algorithm can obtain the same results as the old ones. The first comparison is between the "successive difference (SD)" and the Fuzzy Logic FCL. These two algorithms are used to find the one-pixel resolution fringe center. In the fuzzy logic based FCL, ninety-two rules were established in the rule base as shown in Table 3-1. Note that it was not desirable to design a full factorial rule base (i.e. 3^5=243 rules) [Kasabov, 1995], because such a rule base may slow down the inspection and even reduce the performance of the system. In order to make the fuzzy FCL be fitted to any intensity distribution curve, the rule base is designed according to Figure 4-4.

The second comparison is between the bisection method [Lin and Parvin, 1990] and the analytical method (close-form solution) method. They were used to calculate the accurate sub-pixel fringe centers. The results were obtained using two different methods and the CPU time for the calculations were recorded for comparisons.

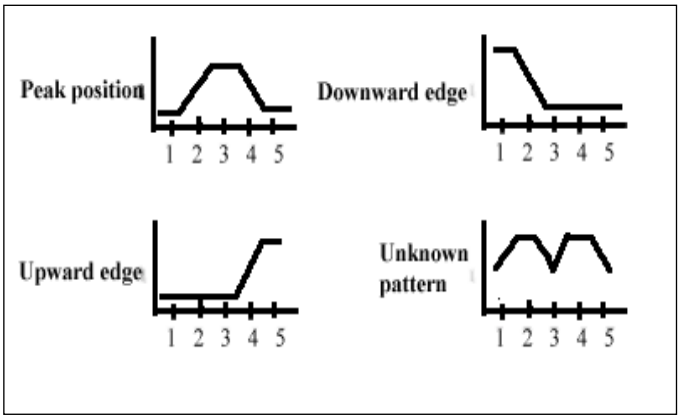

Figure 4-4. A Graphical Representation of Four Types of Intensity Distribution

TABLE 4-1 The 92 Fuzzy Rules for the Fuzzy FCL

Peak position	Unknown patterns	Upward edge	Downward edge
1 2 3 2 1, 1 (1) : 1	3 2 1 2 3, 2 (1) : 1	1 1 1 2 3, 4 (1) : 1	3 2 1 1 1, 5 (1) : 1
1 2 3 3 1, 1 (1) : 1	3 2 1 2 1, 2 (1) : 1	1 2 2 3 3, 4 (1) : 1	3 2 2 1 1, 5 (1) : 1
1 3 3 2 1, 1 (1) : 1	3 2 2 2 3, 2 (1) : 1	1 1 2 2 3, 4 (1) : 1	3 2 2 2 1, 5 (1) : 1
1 1 3 2 1, 1 (1) : 1	3 1 1 1 3, 2 (1) : 1	1 1 2 3 3, 4 (1) : 1	3 3 2 1 1, 5 (1) : 1
1 2 3 1 1, 1 (1) : 1	3 1 1 1 2, 2 (1) : 1	1 1 2 3 2, 4 (1) : 1	2 3 2 1 1, 5 (1) : 1
1 1 3 3 1, 1 (1) : 1	2 1 1 1 3, 2 (1) : 1	1 1 1 3 2, 4 (1) : 1	2 3 1 1 1, 5 (1) : 1
1 3 3 1 1, 1 (1) : 1	3 3 1 3 3, 2 (1) : 1	1 1 1 3 3, 4 (1) : 1	1 3 2 1 1, 5 (1) : 1
1 2 2 3 1, 1 (1) : 1	2 1 1 1 2, 2 (1) : 1	1 2 2 2 3, 4 (1) : 1	2 2 2 2 1, 5 (1) : 1
1 3 2 2 1, 1 (1) : 1	2 2 1 2 2, 2 (1) : 1	1 2 2 2 2, 4 (1) : 1	3 3 3 1 1, 5 (1) : 1
1 1 3 3 2, 1 (1) : 1	3 1 1 2 3, 2 (1) : 1	1 1 2 3 1, 4 (1) : 1	1 3 1 1 1, 5 (1) : 1
1 2 3 3 3, 1 (1) : 1	3 2 1 1 3, 2 (1) : 1	1 1 1 3 1, 4 (1) : 1	1 2 1 1 1, 5 (1) : 1
1 2 3 2 2, 1 (1) : 1	3 1 1 1 1, 2 (1) : 1	1 1 1 2 1, 4 (1) : 1	
2 2 3 2 1, 1 (1) : 1	1 1 1 1 3, 2 (1) : 1	1 1 3 3 3, 4 (1) : 1	
2 2 3 2 1, 1 (1) : 1	3 3 1 1 1, 2 (1) : 1		
2 3 3 3 3, 1 (1) : 1	2 1 1 1 1, 2 (1) : 1		
2 3 3 3 2, 1 (1) : 1	1 1 1 1 2, 2 (1) : 1		
2 3 3 3 1, 1 (1) : 1	1 1 1 2 2, 2 (1) : 1		
1 3 3 3 1, 1 (1) : 1	2 2 1 1 1, 2 (1) : 1		
3 3 3 3 1, 1 (1) : 1	2 2 2 1 1, 2 (1) : 1		
3 3 3 3 2, 1 (1) : 1	1 1 2 2 2, 2 (1) : 1		
3 3 3 2 1, 1 (1) : 1	1 1 1 1 1, 2 (1) : 1		
1 2 3 1 1, 1 (1) : 1	2 2 2 3 2, 2 (1) : 1		
2 3 3 1 1, 1 (1) : 1	2 2 3 2 2, 2 (1) : 1		
3 3 3 3 3, 1 (1) : 1	1 3 1 1 2, 2 (1) : 1		
2 2 2 2 2, 1 (1) : 1	1 2 1 1 3, 2 (1) : 1		
1 1 3 1 1, 1 (1) : 1	1 3 1 1 3, 2 (1) : 1		
	3 1 3 1 3, 2 (1) : 1		
	3 1 2 1 3, 2 (1) : 1		
	2 1 2 1 2, 2 (1) : 1		
	2 1 3 1 2, 2 (1) : 1		
	2 2 3 1 1, 2 (1) : 1		
	3 3 1 1 2, 2 (1) : 1		
	3 1 1 2 1, 2 (1) : 1		
	3 1 1 2 2, 2 (1) : 1		
	1 3 1 3 1, 2 (1) : 1		
	1 2 1 2 1, 2 (1) : 1		
	1 2 1 3 1, 2 (1) : 1		
	1 3 1 2 1, 2 (1) : 1		
	3 1 2 2 1, 2 (1) : 1		
	3 2 3 1 1, 2 (1) : 1		
	3 1 1 3 1, 2 (1) : 1		
	1 1 2 1 1, 2 (1) : 1		

Verbal expression of the rules:

3 2 3 1 1, 2 (1) :1 => **If** [I(1) is bright] **and** [I(2) is light] **and** [I(3) is bright] [I(4) is dark] **and** [I(5) is dark] **then** [pattern is known]

":1" stands for an AND logic operation, value =1 meaning that all input variables are equally important

"(1)" is weight, default =1

The next part of the experiment is to verify if the fuzzy FCL can find the exact fringe centers, and the final part of the experiment is to test if the new system can reconstruct the broken fringes.

4.1.3 Different FCL and Subpixel Calculation Algorithms

Our old algorithms for 3-D inspection had been verified before. The first experiment is to verify if the accuracy of the new algorithms is as good as the old ones. Table 4-2 and 4-3 list the detailed descriptions of programs, models and ROI used for testing.

Table 4-2 The Programs and their Algorithms

Program name	File name	FCL algorithm	Sub-pixel algorithm	Language
P1	FR6.EXE	Successive Difference	Numerical (Bisection)	FORTRAN
P2	FR7.EXE	Successive Difference	Analytical	FORTRAN
P3	Fringe98a.m	Successive Difference	Analytical	MATLAB
P4	Fuzzyfringe.m	Fuzzy logic	Analytical	MATLAB

Table 4-3 The Parameters for Comparisons of Different Algorithms

Algorithms for Comparisons	ROI (starting row, starting col., width, height)	Model name	Number of fringes detected
P2 and P4	(259,156,98,80)	Model 1 (Both)	7
P1 and P3	(259,156,98,80)	Model 1 (Both)	10
P2 and P3	(259,156,98,80)	Model 1 (Both)	10
P3 and P4	(180,55,145,145)	Model 1 (Left one)	12
P3 and P4	(180,195,175,145)	Model 1 (Right one)	11

The next experiment was to verify if the fuzzy logic based FCL can detect the approximate fringe centers accurately. All three models were used, and two fringes were selected from each model. By manually examining the intensity values of the selected

fringes, the position that has the maximum intensity can be easily identified. For example, if a row of intensity is [0, 7, 75, 225, 248, 9, 0], the approximate center will be at column 4 with the intensity value of 255.

In the experimental results, the difference between the approximate center (identified by the user) and the fuzzy fringe center is defined as follows. We assume that approximate fringe center is at column 4. If the fuzzy fringe center is at column 4, the difference will be reported as 0 (4 - 4 = 0). If the Fuzzy FCL center is at column 5, the output will be reported as 1 (5 - 4 = 1). Fringe #2 and #5 were selected from model 1 (two different surfaces), Fringe #2 and #7 were selected from model 2 (Very curved surface), and Fringe #3 and #7 were selected from model 3 (flat surface with broken fringes).

The model 3 (steel block) was also used in the final experiment. We selected the ROI that covered the lowest step and the inspection platform (Fig 4-3). A reconstructed 3-D surface was displayed on the screen and a cross-section view along the fourth fringe of this surface was output to the screen. The results can be seen in the next chapter.

4.2 Object Recognition Techniques

Object recognition (OR) plays an important role in this research work. The centroid, size and perimeter of the object will be extracted first. After the object is recognized, more detail information can be retrieved from the database, and the region of interest (ROI) can be set accordingly.

A backpropagation neural network (BPNN) was chosen to perform the object recognition. Since the recognized object is coded in a binary mode (i.e. 0 or 1), the

BPNN is well suited. Another object recognition technique uses the angle-of-sight (AOS) signatures. Combining these two techniques could make object recognition more accurate and reliable.

4.2.1 Objects Used in Object Recognition

In this experiment, a total of 16 objects were used (Figure 4-5). These objects were made very thin to avoid shadows. The detailed description of these objects is shown in Table 4-4.

As can be seen in Fig.4-5, the first three objects are of regular shapes: circle (part #1), rectangle (part #2), and triangle (part #3). All objects were purposely made to have rough edges (to simulate the geometric errors).

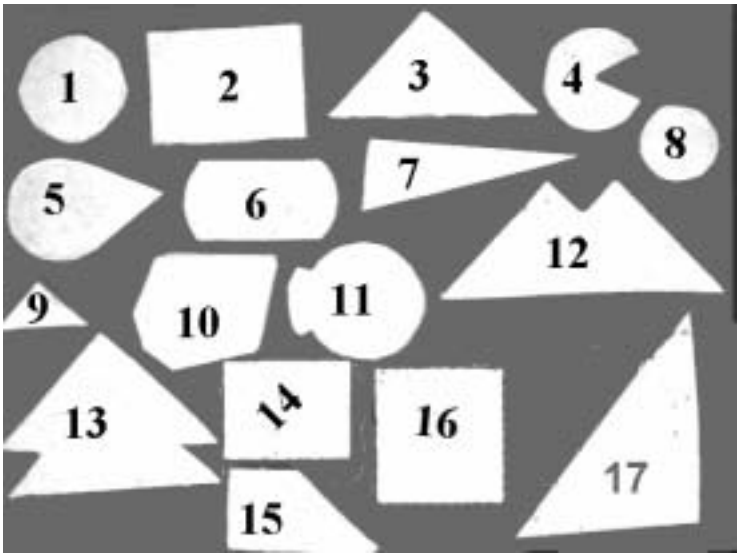

Figure 4-5. The Models Used in Object Recognition

TABLE 4-4 Descriptions of Models Used in Object Recognition	
Object Number	Descriptions
1,2,3	Regular shape (referenced/standard) 1:circle, 2:rectangle, 3:triangle
4,5,8,11	Circle-like 4,5,11: variation in shape, 8: variation in size (smaller)
6,14,15,16	Rectangle-like
7,9,12,13,17	Triangle-like 12,13: variation in shape, 9: variation in size (smaller)
10	unknown object

4.2.2 Object Recognition by Means of Neural Networks

A 3-inputs and 1-output BPNN was trained for object recognition. The three input parameters are the area, axis ratio and perimeter. The full screen has 640 by 480 pixels (307,200 pixel^2), so the maximum area of a recognized object should be no more than 307,200 pixel^2. The maximum perimeter of a rectangle is the whole screen whose perimeter is 2 x (640+480) pixels (2,240 pixels). There are three objects used in this type of object recognition, circular, rectangular and triangular. The ratios of minimum radius to maximum radius are equal to 0.5, 1 and 0.3, respectively. The input data were normalized in advance. The area was divided by 30720 instead of 307,200, and the perimeter was divided by 2240 such that the normalized area and perimeter values will not be too small.

The output of the neural network mapping is the part number of each object. It was found possible to simply use integers for part numbers. The rule of thumb for calculating the number of hidden layer is given by the following formula:

$$\frac{(number\ of\ input\ neurons)+(number\ of\ output\ neurons)}{2}+\sqrt{number\ of\ training\ patterns}$$

From the training data listed in Table 4-5, 58 training patterns were extracted from the acquired images. Each data set includes the normalized area and perimeter, and the

Table 4-5 The BPNN Training Data

Input node	3	Output node	1
Activation function	Hidden layer: Tan-Sigmoid function Output layer: Linear line function		
Nodes in hidden layer (s1)	10		
Maximum training epoch	5000	Learning rate	0.01
Stopped when SSE reaches	0.008		
Number of training patterns	58 (20 circles,20 rectangles,18 triangles)		

ratio of the maximum radius to the minimum radius. The total training time took about 20 seconds. The plot of training-epochs versus sum-square-errors is shown in Figure 4-6.

After the training, the network is ready for object recognition. Thirty-two different parts were selected. As a result, objects can be recognized regardless of their

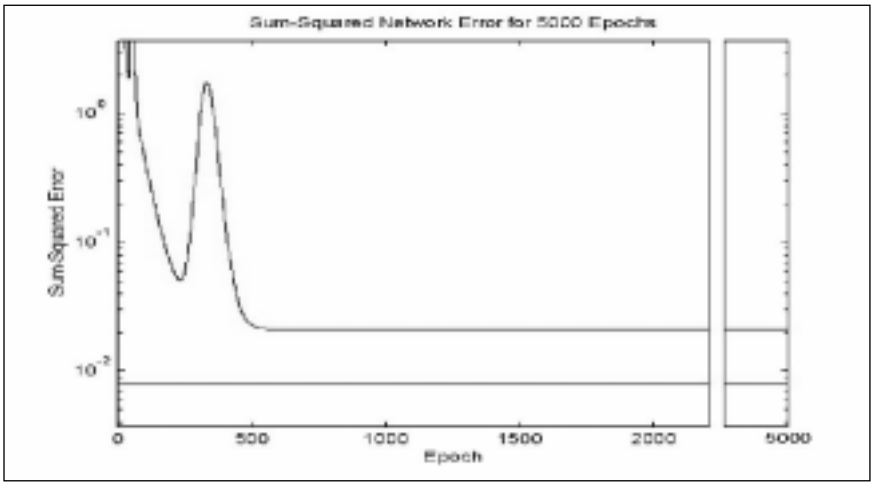

Figure 4-6. The Plot (Epoch vs. SSE) of the Training Process

size. The shape is the only thing that matters. In fact, the object does not have to be complete in shape in order to be recognized.

4.2.3 Object Recognition by Means of AOS Signatures

In using the angle of sight (AOS) signature recognition method, the first thing to do is to make a look-up table. For each referenced part (1,2 and 3), 10 samples for each referenced part are analyzed, and their average values are shown in Table 4-6. A look-up table was used in the program. Their original values are listed in the Appendix. The

Table 4-6 The Look-up Table for AOS Signature Recognition Algorithm

Part #	Average radius	Maximum radius	Minimum radius	Number of corner	Inter-corner Distance (ICD) (ascending sorted)
1	46 (0.9484)	48.5 (1.00)	43.5 (0.8969)	0	none
2	65 (0.7831)	83 (1.00)	48.3 (0.5819)	4	71,74,104,106
3	57 (0.600)	95 (1.00)	30.2 (0.3178)	3	7,142,210

Unit of all radii is in pixel, numbers in the parenthesis are normalized with respect to the maximum radius
Values in each cell are the average of 10 samples
Part #1: circle, part #2: rectangle, part #3: triangle

same 32 parts tested in the NN recognition algorithm are used for an accuracy comparison between these two algorithms.

To perform the AOS signature analysis, a Matlab script file was used. Four rules are listed below, Where R_{ave}, R_{max} and R_{min} represent the actual values defined in the look-up table, R_{ca}, R_{cmax} and R_{cmin} are the calculated values from the actual testing parts, and R stands for radius. The vector *RE(i)* is the result, and *i* is the rule number. The values set in all rules are based on some preliminary tests earlier. In order to recognize both

standard and non-standard sizes, the normalized AOS signatures were used. The six rules are listed below:

Rule 1. $| R_{ave} - R_{ca} |$ = A1; Original values;
If (A1 ≤ 5) THEN the part is a circle; otherwise *RE* = lookuptable(min(A1)).

Rule 2. $| NR_{ave} - NR_{ca} |$ = A1; Normalized values;
If (A1 ≤ 0.104) THEN the part is a circle; otherwise *RE* = lookuptable(min(A1)).

Rule 3. $| R_{ca} - R_{cmax} |$ = A2 and $| R_{cmin} - R_{ca} |$ = B2; Original values;
IF (A2 ≤ 5) AND (B2 ≤ 5) THEN the part is a circle and *RE* = 1;
IF (20 ≤ A2 ≤ 5) AND (20 ≤ B2 ≤ 5) THEN the part is a rectangle and *RE* = 2;
IF (40 ≤ A2 ≤ 20) AND (40 ≤ B2 ≤ 20) THEN the part is a triangle and *RE* = 3;
OTHERWISE *RE* = 0;

Rule 4. $| NR_{ca} - NR_{cmax} |$ = A2 and $| NR_{cmin} - NR_{ca} |$ = B2; Normalized values;
IF (A2 ≤ 0.104) AND (B2 ≤ 0.104) THEN *RE* = 1;
IF (0.28 ≤ A2 ≤ 0.16) AND (0.28 ≤ B2 ≤ 0.16) THEN *RE* = 2;
IF (0.45 ≤ A2 ≤ 0.35) AND (0.34 ≤ B2 ≤ 0.24) THEN *RE* = 3;
OTHERWISE *RE* = 0;

Rule 5. Assign the number of corners according to the rule 4.
(RE = 1) -> 0; (RE = 2) -> 4; (RE = 3) -> 3; otherwise 5.

Rule 6. Examine all ICDs; Original values;
The difference of each calculated $\Delta\theta_i$ must be smaller than 10 pixels

In the rule 5, the number of corners is from the rule 4, which helps find the angular distances in rule 6 efficiently without spending time in finding the number of corners from the object's AOS signature. Using a database to recognize an object was discussed in Pichumani's dissertation [Pichumani, 1996]. When the above six rules are applied, the matched part number will be assigned in the output. Otherwise, a message of "unknown object" will be displayed. The test results will be shown in the next chapter, in which a comparison of the results between using the NN and the AOS signature is also given.

4.3 Auto-Calibration and Automatic Angle Adjustment

In this part, two calibration targets were used. A circle made of opal glass was used as a calibration target for establishing the calibration map and the automatic angle adjustment. Its diameter is exactly 1.000 inch (25.4 mm) and the thickness is 0.125 inch. Another one is a computer printout paper with 25 circles on it for establishing the calibration map. Each circle is exactly 1.000 inch (25.4 mm) in diameter. The following describes the procedures for auto-calibration and automatic angle adjustment.

4.3.1 Establishing the Calibration Map

The detailed description about the relationship between the area or the perimeter of the circle and the calibration factor (CF) in Chapter III was given earlier, which concluded that the CF could be assumed proportional to the area or perimeter. As mentioned in Chapter III, two methods were used to establish the calibration map. The first one is using the Matlab scripts for the calculation. The second one is using the commercial software, Inspector. These two are described in the detail next.

Part I - Using the Matlab scripts

In this experiment, the inspection platform was equally divided into 25 nodes (5 rows by 5 columns). The calibration circle was placed at each mesh point, and its image was captured and saved. There are a total of 25 images, and each image contains only one calibration circle. Note that the Matlab's image processing toolbox can only deal with one object at a time. The Matlab script read those images one by one, and extracted the centroids, areas and perimeters of those circles. In order to prove that these 25 circles

were enough to generate an accurate calibration map with surface fitting, another set of 49 points (7 rows by 7 columns) were done for the sake of comparison.

Part II - Using the Inspector

The previous map needs 25 movements of the calibration circle and 25 image files. In order to improve performance of this procedure, a computer printout paper with 25 circles was utilized here. When the paper was placed on the inspection platform properly, all 25 circles were fitted into the 640x480 pixels screen. After the image containing the 25 circles was captured, the centroids, areas and perimeters were extracted by the "blob-analysis" toolbox provided by the Inspector. Those data were copied to other software and saved as a data file in order to establish the calibration map. Those 25 points can only generate a rough dorm-like surface as shown in Figure 4-7.

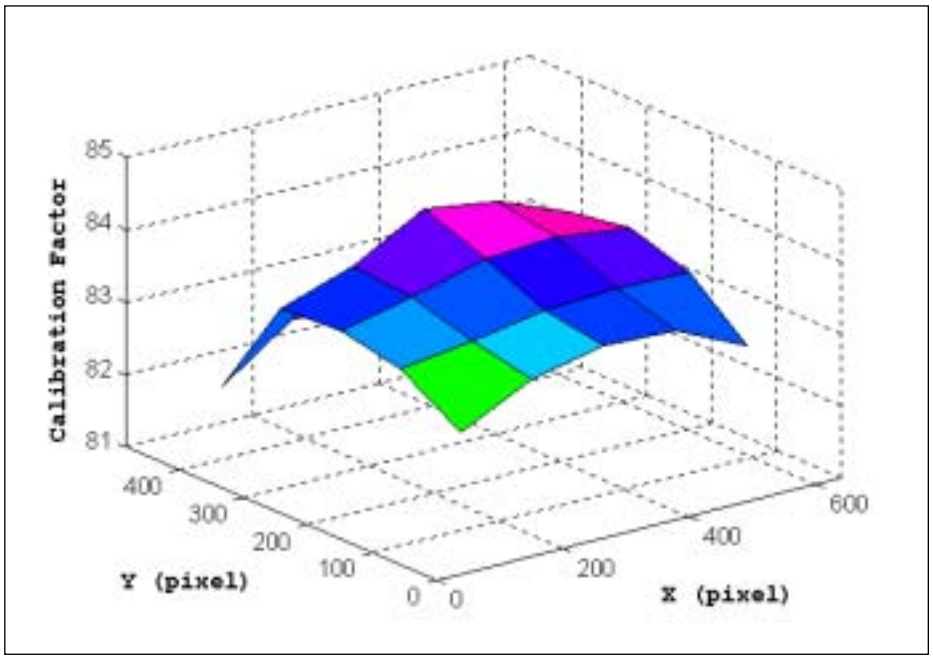

Figure 4-7. A Rough Dorm-like Surface Generated by 25 points

Surface Fitting

The calibration map serves as a function for obtaining the calibration factor when the centroid (x,y) of an object is entered. A dorm-like surface can be approximated by a second-order polynomial in 3-D. This polynomial can be expressed by Eq.(4-1).

$$z = f(x,y) = ax^2 + bxy + cy^2 + dx + ey + f \quad \ldots\ldots \quad \text{Eq.(4-1)}$$

The boundary conditions are: $0 \leq x \leq 640$ and $0 \leq y \leq 480$ and the centroid position is (x_c,y_c), where $0 \leq x_c \leq 640$ and $0 \leq y_c \leq 480$.

Performance Comparison between Using Matlab and Inspector

There are pros and cons for using both methods. In using the Matlab script, it is time consuming to establish the calibration map.

However, it is more convenient to use the commercial software, the inspector, in our inspection system. It has a friendly user interface, and it is easy to operate. The calculation time is fast. It takes about 3 seconds to obtain all the data points. The only disadvantage is that it is an external program to our Matlab main program. It needs an "copy-and-paste" function to send the data directly to the Matlab. Table 4-7 shows the performance comparison between these two methods.

Table 4-7 The Performance of the Matlab Script and the Inspector in Finding Calibration Map for 25 points

Procedures	Time	Software	Time	Software
Capture and save image	25 sec.	Inspector	1 sec.	Inspector
Find centroids, area	12 min.	Matlab	1 sec.	Inspector
Establish cal. map	1 sec.	Matlab	manually	Excel
Total	12 min. 26 sec.		Depend on user	

4.3.2 Automated Camera Angle Detection Module

For an automated inspection system, it is always desirable to have the system automatically detect the change of the camera-viewing angle and automatically compensate the calibration factor for the angle change. The test results will be shown in the next chapter.

The first method was using the relationship between the camera's angles and the center position of the calibration circle appeared on the screen. For example, if the circle appears at (110, 50), the camera's angle is (-7°, -5°). This relationship is based on the calculation of trigonometry described in Chapter III. The distance between the camera and the platform is measured as 375 mm (d_c). We assume that the yaw angle is θ and pitch angle is ϕ and the position of the circle on the screen is (x, y) in the unit of mm. The centroid measured by the program is (X_c ,Y_c) in pixel, and the center of the screen is (320,240). The relationship can be expressed by the following equations.

$$d_c * \tan(\theta) = x = (X_c - 320) / CF \quad \ldots\ldots \quad \text{Eq.(4-2)}$$

$$d_c * \tan(\phi) = y = (Y_c - 240) / CF \quad \ldots\ldots \quad \text{Eq.(4-3)}$$

or

$$\theta = \tan^{-1}[(X_c - 320) / (CF * d_c)] \quad \ldots\ldots \quad \text{Eq.(4-4)}$$

$$\phi = \tan^{-1}[(Y_c - 240) / (CF * d_c)] \quad \ldots\ldots \quad \text{Eq.(4-5)}$$

The allowable range is -7° to 7° for the camera's yaw angle and -5° to 5° for the pitch angle. The circular target was initially coincident with the center of the screen when the yaw and pitch angles are zeros. Then, the camera was randomly tilted to different angles within the range. In this procedure, the circular target always stayed at the same position.

The calculated position used the Eq.(4-4) and Eq.(4-5). The yaw and pitch angles were manually measured. The centroids were measured from the screen in the unit of pixels.

The second method used a set of 5 neural networks to interpret the relationship between the position of the circular target and the camera angle. Figure 4-8 shows the block diagrams for determine the camera's angle. In this case, five calibration maps with different yaw angles (pitch angle is fixed at 0°) were used to train five backpropagation neural networks (BPNNs). The testing results are shown in the next chapter.

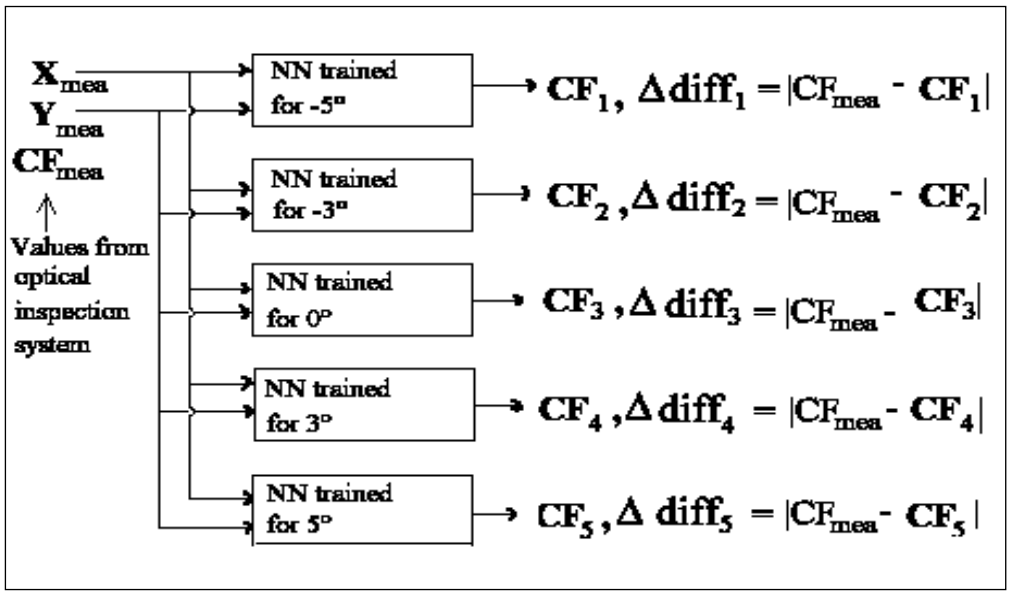

Figure 4-8 The Block Diagram for the Camera Angle Determination via the Trained Neural Networks

Fig. 4-9 shows a flowchart of the automated inspection system step by step. The CPU time at the end of each step is recorded, and at the end of step 7, all the CPU times are accumulated.

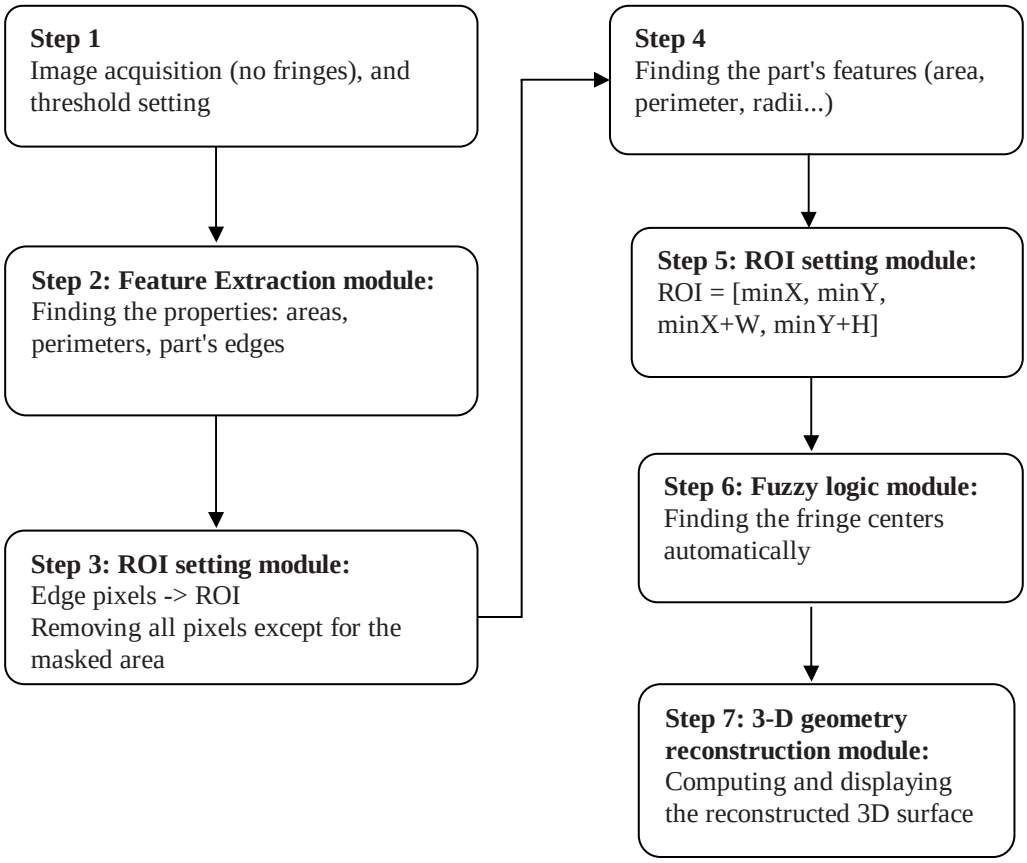

Figure 4-9. The Flowchart of the Automated Inspection System

CHAPTER V

RESULTS AND DISCUSSIONS

This chapter shows the results of testing the developed algorithms and techniques. This dissertation work includes the following three major tasks:

(1) Automated fringe center locator (FCL)

(2) Object recognition using two different techniques

(3) New calibration procedure, and automated detection of camera's angle

The actual CPU time was then recorded so as to evaluate the efficiency of each task in terms of the elapsed time.

5.1 The Test Results of FCL and Subpixel Calculation

FCL stands for fringe center locator, and subpixel means a fraction of image pixel (picture element). Three models were used in this testing. The results show the differences between using the old and new algorithms.

5.1.1 Comparison between Fuzzy Logic Based FCL and Traditional FCL

Table 5-1 shows the comparison of two fringe center locating methods, namely the fuzzy-logic based FCL and the traditional FCL. In this comparison, the fuzzy-logic based FCL was written in Matlab, whereas the traditional FCL was written in FORTRAN program run under the MS-DOS environment. The comparison was made for 80 rows of image data. The average difference of each fringe location is between 0.95 to 1.06 pixels, whereas the RMS (root mean squared) difference is between 0.97 and 1.11 pixels.

Matlab is Window-based software. It also provides a graphical user interface (GUI) development environment. In Matlab, our programs are in scripts. A comparison between the fuzzy-logic based FCL script and the traditional FCL script was made. Both scripts were run in Matlab 5.1. Model 1 was used, in which two ROIs (left and right side) were set on this model. The results are shown in Table 5-2 and 5-3, and there are 12 fringes on the left side of the clay and 11 fringes on the right side of the clay (see Fig. 4-1). All fringes contain 145 rows. The average difference is below 0.0005 pixels and 0.007 pixels, on the left and right sides, respectively. The accuracy difference may have resulted from the fact that the surface of the right side is more curved.

Table 5-1 The Difference of Calculated Centers between Program fr7.exe and fuzzyfringe.m

Fringe #	Average	Standard Deviation	# of Rows	Sum Sq. Error	RMS error
1	1.003886	0.35294	80	90.46374	1.063389
2	0.973664	0.288511	80	82.41755	1.014997
3	1.062199	0.340508	80	99.421	1.114793
4	0.994933	0.27161	80	85.01931	1.030893
5	0.953479	0.210654	80	76.23543	0.976188
6	1.023296	0.298295	80	90.80013	1.065365
7	1.052653	0.244917	80	93.38504	1.080423

(1) fr7.exe uses successive difference to find fringe centers under MS-DOS environment.
(2) fuzzyfringe.m uses fuzzy logic based FCL to find fringe centers under Matlab environment.
(3) All units are in pixel.
(4) Fringe centers are in sub-pixel resolution.
(5) Each fringe uses 80 rows to compute the average.

Table 5-2 The Differences of Calculated Centers between the Old and the New Systems on a Flat Surface

Fringe #	Average	Standard Deviation	# of Rows	Sum Sq. Error	RMS error
1	0	0	145	0	0
2	0	0	145	0	0
3	0.000254	0.003056	145	0.001354	0.003056
4	0	0	145	0	0
5	0.001379	0.013481	145	0.026447	0.013505
6	0	0	145	0	0
7	0.000782	0.007652	145	0.008521	0.007666
8	0	0	145	0	0
9	0.000485	0.005842	145	0.004948	0.005842
10	0.000323	0.003887	145	0.002191	0.003887
11	0.000485	0.005842	145	0.004948	0.005842
12	0	0	145	0	0

(1) Left Side ROI (low curvatures) of the Clay Model 1 was used in this table.
(2) Old system uses program fringe98a.m and old fringe center locator (FCL).
(3) New system uses program fuzzyfringe.m and fuzzy logic based FCL.
(4) All units are in pixel.
(5) Fringe centers are in sub-pixel resolution.
(6) Each fringe uses 145 rows of data to compute the average.

Table 5-3 The Differences of Calculated Centers between the Old and the New Systems on a very Curved Surface

Fringe #	Average	Standard Deviation	# Of Rows	Sum Sq. Error	RMS error
1	0	0	145	0	0
2	0.00703	0.028757	145	0.126248	0.029406
3	0.007935	0.034141	145	0.176972	0.034697
4	0.004392	0.020493	145	0.063274	0.020677
5	0.005881	0.024743	145	0.093172	0.025006
6	0.005406	0.026415	145	0.104717	0.026422
7	0.006533	0.028157	145	0.120359	0.028233
8	0.014486	0.067914	145	0.694602	0.0676
9*	0.351201	1.253706	145	244.2206	1.263413
10	0.005027	0.028276	145	0.118798	0.027774
11	0.006271	0.035798	145	0.190243	0.035034

* The intensity values are too low for the lower 15 rows of the fringe.

(1) Right Side ROI (very curved) of the Clay Model 1 was used in this table.
(2) Old system uses program fringe98a.m and old fringe center locator (FCL).
(3) New system uses program fuzzyfringe.m and fuzzy logic based FCL.
(4) All units are in pixel.
(5) Fringe centers are in sub-pixel resolution.
(6) Each fringe uses 145 rows of data to compute the average.

5.1.2 Comparison between Bisection-based Numerical Method and Analytical Method

In the past, we started with a bisection method to numerically determine the subpixel fringe canters. Later, we switched from the numerical method to an analytical method from which a close-form solution was used. Table 5-4 shows the difference between these two methods. 10 fringes are selected for comparison, where each fringe contains 80 rows. The average difference is between 0.0112 and 0.009 pixels, whereas the RMS (root mean squared) difference is between 0.109 and 0.136 pixels.

Table 5-4 The Differences of Calculated Subpixel Centers between Program fr6.exe and fringe98a.m

Fringe #	Average	Standard Deviation	# of Rows	Sum Sq. Error	RMS error
1	0.112156	0.059761	80	1.288451	0.126908
2	0.108498	0.08322	80	1.488861	0.136421
3	0.104509	0.072224	80	1.285863	0.12678
4	0.090265	0.067101	80	1.007519	0.112223
5	0.094183	0.055552	80	0.953429	0.109169
6	0.093343	0.053139	80	0.92011	0.107244
7	0.089887	0.059631	80	0.927291	0.107662
8	0.10326	0.06475	80	1.184224	0.121667
9	0.091878	0.06186	80	0.977641	0.110546
10	0.097222	0.054843	80	0.993775	0.111455

(1) Fr6.exe uses bisection to find subpixel fringe centers.
(2) Fringe98a.m uses symbolic derived function to find fringe centers.
(3) All units are in pixel.
(4) Fringe centers are in sub-pixel resolution.
(5) Each fringe uses 80 rows to compute the average.

5.1.3 Comparison between Running a FORTRAN Program and Matlab Scripts

Our new programs are run under the Matlab 5.1 environment. The original programs were in FORTRAN source code, and then converted to the equivalent Matlab scripts. A comparison between the Fortran program and the Matlab script is given in Table 5-5. There are 10 fringes selected from the ROI, in which each fringe contains 80 rows. Apparently, there is almost no difference in fringe center detection between these two environments. The average difference is about 2.5×10^{-5} pixels, and the root mean squared difference is about 2.5×10^{-5} pixels. The small difference most likely resulted from run-off errors.

Table 5-5 The Differences of Calculated Centers between Using Fortran Program and Matlab Script

Fringe #	Average	Standard Deviation	# of Rows	Sum Sq.	RMS error
1	2.3E-05	1.51E-05	80	6.03E-08	2.75E-05
2	2.48E-05	1.4E-05	80	6.49E-08	2.85E-05
3	2.4E-05	1.62E-05	80	6.7E-08	2.89E-05
4	2.68E-05	1.54E-05	80	7.62E-08	3.09E-05
5	2.31E-05	1.41E-05	80	5.83E-08	2.71E-05
6	2.57E-05	1.39E-05	80	6.8E-08	2.92E-05
7	2.46E-05	1.31E-05	80	6.18E-08	2.78E-05
8	2.71E-05	1.46E-05	80	7.55E-08	3.07E-05
9	2.57E-05	1.46E-05	80	6.96E-08	2.95E-05
10	2.38E-05	1.4E-05	80	6.07E-08	2.75E-05

(1) Fr7.exe is a Fortran program.
(2) Fringe98a.m is a Matlab script.
(3) Both programs use the same algorithms.
(4) All units are in pixel.
(5) Fringe centers are in sub-pixel resolution.
(6) Each fringe uses 80 rows of data to compute the average.

5.2 The Accuracy of Fuzzy-Logic Based FCL Applied to Different Kinds of Surfaces

In this experiment, the approximate fringe centers (based on one-pixel resolution) were used so that the result could be verified by the user's observation. Also, only the fuzzy-logic based FCL script was used. The column positions of fringes were calculated by the Fuzzy FCL, where the intensity values were analyzed and the column positions were selected manually.

In Table 5-6 and 5-7, fringes 3 and 6 are selected from the image of model 1. There were 22 rows in this comparison. The fuzzy-logic based FCL located the same centers as observed, because the surface's curvature was moderate. In Table 5-8 and 5-9,

fringes 1 and 8 were chosen from the image of model 2. In this comparison, all 23 rows still had the same column positions as the actual approximate fringe centers, although the surface was highly curved. The fuzzy-logic based FCL, in this case, had a 100% reliability on the highly curved surface. Fringe 2 and 7 were selected from the image of a steel block, as shown in Table 5-10 and 5-11. The steel block is a flat and relatively shiny surface. All 20 rows had the same column positions as the actual approximate fringe centers. Again, the FCL had a 100% reliability on this block.

Table 5-6 Result of Accuracy Test of Fuzzy FCL (Model 1, Fringe 3)

Intensity matrix: Selected Column:[27 - 34]; Total Rows = 22; ROI=[65,204,127,106]

	intensity (Approximate centers are in bold faces)								
Fuzzy FCL column positions	**Column =>**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**difference (pixel)**
30	↓ Intensity Value →	10	15	71	**255**	238	0	0	0	0
30		3	15	65	**252**	230	0	0	0	0
31		0	4	45	213	**251**	0	0	0	0
31		0	0	22	193	**255**	52	0	0	0
31		0	0	5	159	**255**	104	0	0	0
31		6	0	1	124	**255**	170	0	0	0
31		2	4	11	113	**255**	199	0	0	0
31		0	8	17	99	**233**	202	0	0	0
32		0	8	14	83	224	**255**	0	0	0
32		2	12	10	68	238	**255**	0	0	0
32		0	15	8	50	218	**255**	37	0	0
32		0	8	7	38	174	**255**	99	0	0
32		0	0	2	28	142	**255**	169	0	0
32		0	0	0	20	122	**255**	227	0	0
32		0	0	0	16	100	**255**	249	0	0
32		0	0	2	11	72	**255**	253	0	0
33		0	0	1	8	46	222	**255**	0	0
33		0	0	0	4	34	190	**255**	0	0
33		0	0	0	0	28	183	**255**	46	0
33		0	0	0	3	21	174	**255**	101	0
33		0	0	0	8	21	147	**255**	172	0
33		0	0	0	9	22	96	**255**	237	0

Accuracy = 100%

Table 5-7 Result of Accuracy Test of Fuzzy FCL (Model 1, Fringe 6)

Intensity matrix: Selected Column:[57 - 64]; Total Rows=22; ROI=[65,204,127,106]

	Intensity (Approximate centers are in bold faces)									
Fuzzy FCL column positions	**Column =>**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**Difference (pixel)**
60		0	21	94	**255**	197	0	0	0	0
60		0	11	66	**255**	239	0	0	0	0
61		0	5	39	255	**255**	0	0	0	0
61		0	3	24	255	**255**	0	0	0	0
61		0	3	17	204	**255**	35	0	0	0
61		0	2	6	145	**255**	154	0	0	0
61		0	1	2	102	**255**	247	0	0	0
61		0	0	7	75	**255**	248	0	0	0
62	**Intensity Value →**	0	0	10	59	219	**255**	17	0	0
62		0	0	8	42	177	**255**	109	0	0
62		0	0	6	28	124	**255**	186	0	0
62		0	0	7	18	78	**255**	234	0	0
63		0	0	8	15	53	255	**255**	0	0
63	**← Intensity Value**	0	0	8	13	38	255	**255**	0	0
63		0	0	4	11	28	255	**255**	0	0
63		0	0	0	16	30	222	**255**	40	0
63		0	0	0	16	25	208	**255**	43	0
63		0	0	0	8	11	181	**255**	93	0
63		0	0	0	3	2	132	**255**	185	0
63		0	0	0	0	0	104	**254**	229	0
63		0	0	0	0	0	93	**241**	236	0
64		0	0	0	0	0	72	213	**255**	0

Accuracy=100%

Table 5-8 Result of Accuracy Test of Fuzzy FCL (Model 2, Fringe 1)

Intensity matrix: Selected Column:[9--18]; Total Rows =23; ROI=[155,206,167,112]

	Intensity (Approximate centers are in bold faces)											
Fuzzy FCL column positions	**Column =>**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**Difference (pixel)**
11		20	110	**255**	241	2	0	0	0	0	0	0
11		14	116	**255**	250	0	0	0	0	0	0	0
12		23	104	255	**255**	0	0	0	0	0	0	0
12		31	80	208	**237**	55	0	0	0	0	0	0
12		31	53	150	**229**	124	0	0	0	0	0	0
12		17	36	159	**255**	156	0	0	0	0	0	0
12		4	27	169	**255**	158	0	0	0	0	0	0
12		7	20	136	**255**	166	0	0	0	0	0	0
12		13	14	95	**236**	222	42	0	0	0	0	0
13	↓ Intensity Value →	11	14	64	203	**255**	111	0	0	0	0	0
13		7	20	48	154	**255**	144	0	0	0	0	0
13		1	13	34	109	**226**	183	0	0	0	0	0
14		0	8	20	74	215	**252**	28	0	0	0	0
14		0	12	14	47	202	**255**	77	0	0	0	0
14		0	7	11	30	152	**255**	129	0	0	0	0
14		0	1	7	23	101	**203**	163	0	0	0	0
15		0	2	4	15	62	147	**167**	62	0	0	0
15		0	3	1	3	27	99	**170**	114	0	0	0
16		0	0	0	0	10	63	158	**159**	2	0	0
16		0	0	0	0	9	36	113	**179**	94	0	0
17		0	0	0	4	7	18	65	163	**184**	28	0
17		0	0	0	3	4	12	40	124	**195**	102	0
17		0	0	0	2	6	11	25	80	**155**	137	0

Accuracy=100%

Table 5-9 Result of Accuracy Test of Fuzzy FCL (Model 2, Fringe 8)

Intensity matrix: Selected Column:[121--130]; total Rows = 23; ROI=[155,206,167,112]

	Intensity(Approximate centers are in bold faces)											
Fuzzy FCL column positions	**Column =>**	**121**	**122**	**123**	**124**	**125**	**126**	**127**	**128**	**129**	**130**	**Difference (pixel)**
122		175	**249**	137	0	0	0	0	0	0	0	0
122		153	**214**	133	0	0	0	0	0	0	0	0
122		133	**225**	158	11	0	0	0	0	0	0	0
122		99	**206**	190	61	0	0	0	0	0	0	0
123		71	161	**193**	111	0	0	0	0	0	0	0
123		52	115	**175**	150	26	0	0	0	0	0	0
123		32	86	**176**	171	25	0	0	0	0	0	0
123		19	76	**163**	152	29	0	0	0	0	0	0
124	↓ Intensity Value →	13	59	130	**153**	75	0	0	0	0	0	0
124		4	36	114	**181**	127	1	0	0	0	0	0
124		0	22	95	**177**	162	41	0	0	0	0	0
125		0	14	62	151	**180**	77	0	0	0	0	0
125		0	8	39	115	**164**	115	19	0	0	0	0
126		0	2	22	71	131	**152**	85	0	0	0	0
126		8	0	4	31	97	**159**	122	19	0	0	0
127		11	2	0	9	67	139	**144**	86	0	0	0
127		0	1	0	6	47	108	**156**	126	0	0	0
127		0	4	3	3	25	74	**140**	131	20	0	0
128		2	5	3	0	8	45	98	**114**	71	26	0
129		0	3	2	0	3	27	64	102	**115**	74	0
129		0	5	4	0	2	14	42	93	**142**	114	0
130		0	4	3	0	2	6	19	64	134	**146**	0
130		0	0	2	0	3	9	10	32	88	**136**	0

Accuracy=100%

Table 5-10 Result of Accuracy Test of Fuzzy FCL (Steel Block, Fringe 2)

Intensity Matrix: Selected column: [20-24]; Total Rows = 20; ROI=[246,218,126,78]]

	Column position (Approximate centers are in bold faces)						
Fuzzy FCL column positions	**Column =>**	**20**	**21**	**22**	**23**	**24**	**Difference (pixel)**
22	↓ Intensity Value →	55	255	**255**	89	0	0
22		51	255	**255**	77	0	0
22		59	255	**255**	62	0	0
22		66	255	**255**	62	0	0
22		65	255	**255**	71	0	0
22		59	255	**255**	64	0	0
22		63	255	**255**	38	0	0
21		77	**255**	241	13	0	0
21		84	**255**	239	0	0	0
21		81	**255**	252	0	0	0
22		75	255	**255**	0	0	0
22		78	255	**255**	0	0	0
22		95	255	**255**	0	0	0
22		111	255	**255**	0	0	0
22		116	255	**255**	0	0	0
22		117	255	**255**	0	0	0
22		117	255	**255**	0	0	0
22		120	255	**255**	0	0	0
22		120	255	**255**	0	0	0
22		108	255	**255**	0	0	0

Accuracy=100%

Table 5-11 Result of Accuracy Test of Fuzzy FCL (Steel Block, Fringe 7)

Intensity Matrix: Selected Column:[90-94]; Total Rows = 20; ROI=[246,218,126,78]

	Intensity (Approximate centers are in bold faces)						
Fuzzy FCL column positions	**Column =>**	**90**	**91**	**92**	**93**	**94**	**difference (pixel)**
92		9	104	**252**	154	0	0
92		12	108	**255**	163	0	0
92		15	114	**255**	163	0	0
92		15	118	**255**	166	0	0
92		11	124	**255**	178	0	0
92		7	124	**255**	183	0	0
92		8	115	**255**	169	0	0
92		11	110	**243**	144	0	0
92		12	116	**228**	125	0	0
92		13	127	**226**	110	0	0
92		16	130	**221**	103	0	0
92		18	126	**218**	106	0	0
92		19	128	**224**	111	0	0
92		22	137	**241**	122	0	0
92		24	149	**255**	130	0	0
92		27	169	**255**	126	0	0
92		33	192	**255**	120	0	0
92		36	205	**255**	115	0	0
92		34	206	**255**	84	0	0
92		30	179	**198**	40	0	0

Intensity Value → (↓)

Accuracy=100%

5.3 The Results of Object Recognition

In object recognition, two algorithms are used. They are artificial neural network and angle-of-sight (AOS) signatures.

5.3.1 Results of Testing the Neural Network Algorithm

In training a neural network, a total of 58 data sets (i.e. training patterns) were used for training. After the training, the neural network was ready for object recognition. A total of 32 images of different objects were captured by a camera, and their properties (centroid, area, perimeter …etc.) were extracted for recognition. Program "testmybp.m" generated a set of output as shown in Table 5-12, and Table 5-13 summarizes the output. The success rate of recognition was 87.5% in which 4 out of 32 parts could not be recognized.

5.3.2 Results of Testing the AOS Signature Algorithm

The same 32 images were also used in testing the AOS signature algorithm. Three properties namely maximum, minimum and average radii, and the AOS signatures were recorded. For instance, circle, rectangle and triangle were registered as model number 1, 2 and 3, respectively. The final output was a vector of six elements: average radius, normalized average radius, maximum and minimum radii, normalized maximum and minimum radii, number of corners matched, and angular distance (inter-corner distance). For example, a vector [2 2 2 2 2 2] means that a rectangular object has been identified. An object may be identified as an unknown object if the AOS signature did not match anything in our database. The final results are shown in Table 5-14. The success rate of recognition was 100%.

Table 5-12 A Sample of Output by the NN Object Recognition Algorithm

Original Object shape	Normalized Area	R_{min}/R_{max}	Normalized Perimeter	Desired Result	Object's Orientation	Screen output from NN recognition
Unknown	0.255685	0.499826	0.164511	2	80.9144	Unknown
rectangle	0.407977	0.580896	0.210003	2	84.3382	rectangle (2)
triangle	0.267757	0.313402	0.198693	9	95.6773	triangle (3)
circle	0.210636	0.888994	0.137905	1	48.7844	circle (1)
triangle	0.267696	0.312391	0.205390	9	96.8170	triangle (3)
rectangle	0.407977	0.586160	0.221166	2	83.1554	rectangle (2)
triangle	0.266748	0.320321	0.209759	3	95.1683	triangle (3)
Unknown	0.054401	0.293637	0.092210	9	43.4613	triangle (3)
Unknown	0.110544	0.903714	0.098952	9	35.4591	circle (1)
Unknown	0.165678	0.118179	0.151948	9	51.3469	Unknown
Unknown	0.182060	0.341190	0.040279	9	118.2394	Unknown
Unknown	0.274935	0.514703	0.168949	9	49.8751	Unknown
Unknown	0.499480	0.486717	0.296739	9	107.1068	Unknown
Unknown	0.212225	0.376794	0.162118	9	82.3382	Unknown
Unknown	0.440587	0.276273	0.284138	9	127.3875	Unknown
circle	0.233464	0.892683	0.140060	1	50.0170	circle (1)
rectangle	0.443652	0.581442	0.215247	2	84.8294	rectangle (2)
circle	0.233691	0.894917	0.140124	1	50.1978	circle (1)
Triangle	0.297428	0.330877	0.044815	3	94.5868	Triangle (3)
triangle	0.302865	0.335216	0.198680	3	95.3298	triangle (3)
rectangle	0.446029	0.588080	0.226097	2	84.3069	rectangle (2)
Unknown	0.324967	0.600760	0.177360	9	63.0648	Unknown
circle	0.233919	0.902185	0.140800	1	49.6668	circle (1)
rectangle	0.445801	0.574443	0.218684	2	85.4106	rectangle (2)
Unknown	0.528060	0.273821	0.294317	9	79.9876	rectangle (2)
triangle	0.293001	0.327202	0.209867	3	95.7615	triangle (3)
rectangle	0.445573	0.579107	0.223028	2	85.2577	rectangle (2)
rectangle	0.441829	0.582309	0.225001	2	83.7701	rectangle (2)
circle	0.233464	0.901175	0.140908	1	49.8050	circle (1)
Unknown	0.308138	0.759767	0.168975	9	62.8977	Unknown
Unknown	0.311947	0.612963	0.174233	9	85.2476	Unknown
Unknown	0.426172	0.685917	0.207428	9	83.7979	rectangle (2)

Only three different shape of objects (circular, rectangular and triangular) are to be recognized.
Unknown: Object cannot be recognized, which means that the object is other than circular, rectangular and triangular.

Table 5-13 The Result of Object Recognition through Neural Network

Order	Object #	Recognized as	Result	Order	Object #	Recognized as	Result
1	5	unknown	1	**16**	1	1	1
2	2	2	1	**17**	2	2	1
3	3	3	1	**18**	1	1	1
4	1	1	1	**19**	3	3	1
5	3	3	1	**20**	3	3	1
6	2	2	1	**21**	2	2	1
7	3	3	1	**22**	16	unknown	0
8	9	3	1	**23**	1	1	1
9	8	1	1	**24**	2	2	1
10	4	unknown	1	**25**	17	2	0
11	7	unknown	1	**26**	3	3	1
12	14	unknown	0	**27**	2	2	1
13	13	unknown	1	**28**	2	2	1
14	15	unknown	1	**29**	1	1	1
15	12	unknown	1	**30**	11	unknown	1
				31	6	unknown	1
				32	10	2	0

"1" for correctly recognized, and "0" for incorrectly recognized.
Test Result: out of 32 samples, 28 recognized and 4 failed

Part#1, #2 and #3 are references.
Part#4 through part#15 are defected
Program used: trainmybp.m for training, testmybp.m for testing

Table 5-14 The Result of Object Recognition through AOS Signature

Reference		Results of all 6 rules (displayed in model number)						Result: model number			
Order	Part #, model #	1	2	3	4	5	6	AOS	Success	BPNN	Success
1	5,0	3	0	0	0	3	0	unknown	Y	unknown	Y
2	2,2	2	2	2	2	2	2	2	Y	2	Y
3	3,3	3	3	3	3	3	3	3	Y	3	Y
4	1,1	1	1	1	1	1	1	1	Y	1	Y
5	3,3	3	3	3	3	3	3	3	Y	3	Y
6	2,2	2	2	2	2	2	2	2	Y	2	Y
7	3,3	3	3	3	3	3	3	3	Y	3	Y
8	9,3	0	3	0	3	3	2	3_s	Y	3	N
9	8,1	0	1	1	1	1	1	1_s	Y	1	N
10	4,0	0	0	0	0	2	0	unknown	Y	unknown	Y
11	7,0	0	0	0	0	3	1	unknown	Y	unknown	Y
12	14,2	3	2	2	2	2	2	2_s	Y	unknown	Y
13	13,0	0	0	0	0	2	3	unknown	Y	unknown	Y
14	15,0	1	0	0	0	3	0	unknown	Y	unknown	Y
15	12,3	0	3	3	3	3	0	3_s	Y	unknown	Y
16	1,1	1	1	1	1	1	1	1	Y	1	Y
17	2,2	2	2	2	2	2	2	2	Y	2	Y
18	1,1	1	1	1	1	1	1	1	Y	1	Y
19	3,3	3	3	3	3	3	3	3	Y	3	Y
20	3,3	3	3	3	3	3	3	3	Y	3	Y
21	2,2	2	2	2	2	2	2	2	Y	2	Y
22	16,2	2	0	0	0	2	2	2_s	Y	unknown	Y
23	1,1	1	1	1	1	1	1	1	Y	1	Y
24	2,2	2	2	2	2	2	2	2	Y	2	Y
25	17,3	0	3	3	3	3	0	3_s	Y	2	N
26	3,3	3	3	3	3	3	3	3	Y	3	Y
27	2,2	2	2	2	2	2	2	2	Y	2	Y
28	2,2	2	2	2	2	2	2	2	Y	2	Y
29	1,1	1	1	1	1	1	1	1	Y	1	Y
30	11,0	3	0	0	0	1	2	unknown	Y	unknown	Y
31	6,0	3	0	0	0	2	2	unknown	Y	unknown	Y
32	10,0	3	0	0	0	2	2	unknown	Y	2	N

The results marked with "_s" means different size. (i.e. 2_s = part #2 with a different size)
Reference Part # 1, 2 and 3: standard; 5 through 15: unknown or variant size
Reference model # 1 (circle), 2 (rectangle), 3 (triangle), 0 (not regular shape)
Values in column 3,4,5,6,7 and 8: model numbers output by AOS signature object recognition method
Values in column 9 and 11: "unknown" means "not regular shape"
Note 1: a variant size circle
Note 2: a variant size rectangle
Note 3: a variant size triangle

5.4 The Calibration Map and the Camera Angle Detection

Figure 5-1 shows the calibration map consisting of 49 (7 by 7 matrix) calibration

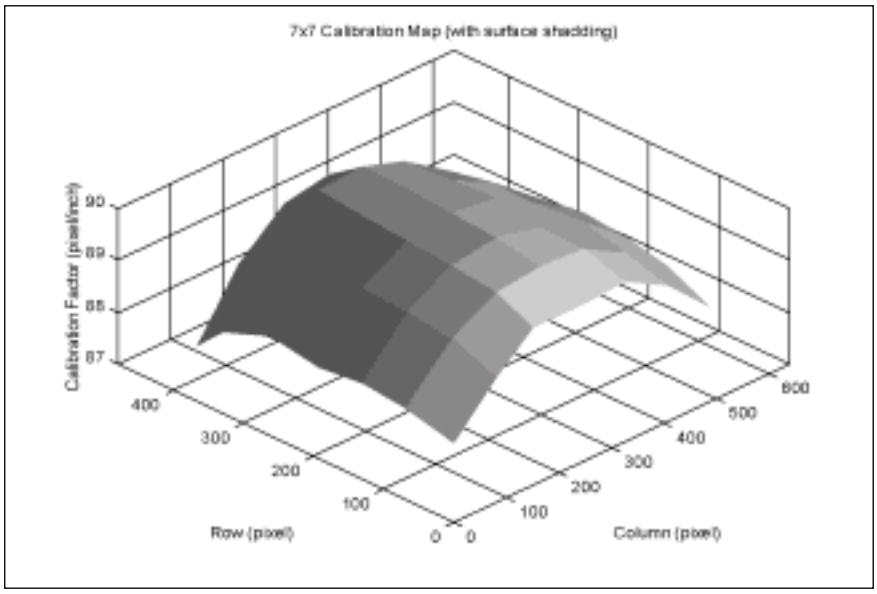

Figure 5-1. The Calibration Map Established by 49 Data Sets

factors (CFs). It can be seen in Fig. 5-1, the shape of the map resembles a second order polynomial. Note that the procedure for generating this map was described earlier in Chapter IV.

In terms of automatic angle detection, two methods were considered, and also described in chapter IV. A neural network was first used to find the camera angle when entering a position (x,y) and a calibration factor (CF_{mea}) in our optical inspection system. Based on the block diagram for determining the camera's angle in Chapter IV, we consider the closest difference (diff.) and the second closest difference between the CF_{nn} (CF from NN) and CF_{mea} (CF from inspection system). The success rate of the neural network prediction was 74% as shown in Table 5-15. This result is for the change of yaw angle only, while the pitch angle was fixed at 0°.

Table 5-15 Automatic Angle Detection using Neural Network

	Data set 1-12											
Exact angle (deg.)	-5	-5	-5	-5	-5	-5	-5	-5	-3	-3	-3	-3
Smallest $\Delta diff_i$	-3	-3	-3	-3	3	-3	3	-3	-5	-5	-3	3
2nd smallest $\Delta diff_i$	-5	-5	-5	-5	5	0	0	-5	-3	-3	-5	-3
Matched status	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y
	Data set 13-24											
Exact angle (deg.)	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Smallest $\Delta diff_i$	3	-3	-5	-5	-5	-3	0	-3	-5	3	3	-3
2nd smallest $\Delta diff_i$	5	3	-3	-3	-3	3	-5	-5	-3	-3	0	3
Matched status	N	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y
	Data set 25-36											
Exact angle (deg.)	-3	-3	-3	-3	0	0	0	0	0	0	0	0
Smallest $\Delta diff_i$	-3	-3	-3	-3	0	3	0	3	-3	0	0	0
2nd smallest $\Delta diff_i$	3	0	-5	-5	-5	-3	5	-3	0	3	-3	5
Matched status	Y	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y
	Data set 37-48											
Exact angle (deg.)	0	0	0	0	0	0	0	0	0	0	0	3
Smallest $\Delta diff_i$	5	5	-5	-5	0	-5	-3	3	3	5	3	3
2nd smallest $\Delta diff_i$	0	3	-3	-3	-5	-3	0	0	0	0	-5	5
Matched status	Y	N	N	N	Y	N	Y	Y	Y	Y	N	Y
	Data set 49-60											
Exact angle (deg.)	3	3	3	3	3	3	3	3	3	3	3	3
Smallest $\Delta diff_i$	0	5	5	5	3	0	0	-3	3	3	0	5
2nd smallest $\Delta diff_i$	5	0	3	3	5	3	3	-5	0	0	-3	0
Matched status	N	N	Y	Y	Y	Y	Y	N	Y	Y	N	N
	Data set 61-72											
Exact angle (deg.)	3	5	5	5	5	5	5	5	5	5	5	5
Smallest $\Delta diff_i$	-3	5	5	5	0	-3	0	5	5	5	5	3
2nd smallest $\Delta diff_i$	-5	3	3	3	-5	3	5	0	3	3	3	5
Matched status	N	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y
	Data set 73-81											
Exact angle (deg.)	5	5	5	5	5	5	5	5	5			
Smallest $\Delta diff_i$	0	0	5	5	5	0	0	3	5			
2nd smallest $\Delta diff_i$	5	3	3	3	3	5	5	-5	3			
Matched status	Y	N	Y	Y	Y	Y	Y	N	Y			

All numbers in the boxes indicate angles (degree).

$\Delta diff_I = | CF_{mea} - CF_{nn} |_i$, where i = 1, 2, 3, 4, 5 and θ_i= -5°,-3°,0°,3°,5° respectively.

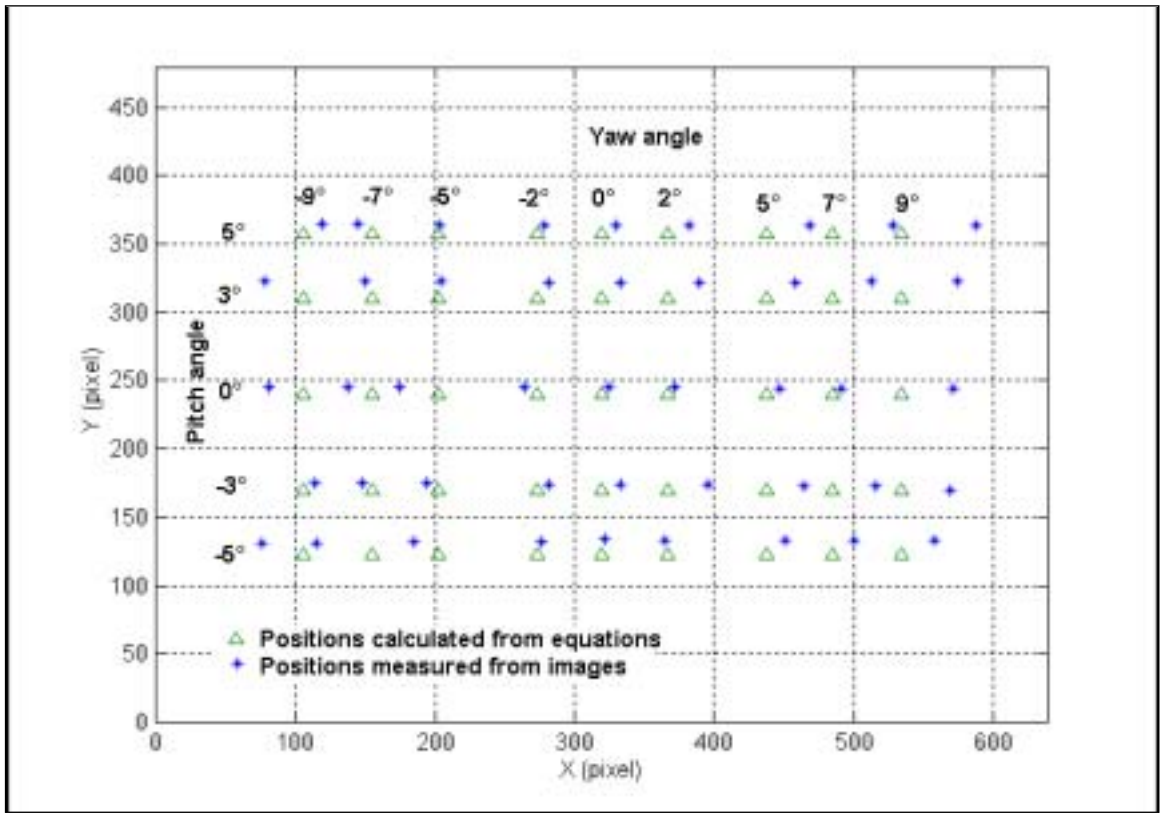

Figure 5-2. The Mapping between the Calibration Circle's Position and Camera Angles

CF_{mea} is calibration factor from optical inspection system
CF_{nn} is calibration factor from the i-th neural network
***If the "smallest" or "2nd smallest" matched the exact angle, the "Matched status" will be marked as "Y".**
Total tests = 81 sets
Total data sets with angle matched = 60 sets
Success rate = 74%

Another calculation uses the equations in the previous chapter. The angle is determined by a simple principle of trigonometry (Eq. 4-4 and 4-5). Figure 5-2 shows the mapping relationship between the calibration circle's center position and the camera angle. As we can see, some measured positions do not match the calculated positions. The

discrepancy could be mostly due to the measured angular error, in which a simple protractor was used to manually adjust the camera's angle between -5° and 5°.

The next task is to plot the relationship between the centroid of the circle and the camera angle. Figure 5-3 shows the two plots representing two different camera angles. The figure also shows how the calibration map changes as the camera's angle varies. The meshed one is for the camera at yaw=0° and pitch=0°, whereas the shaded one is for the

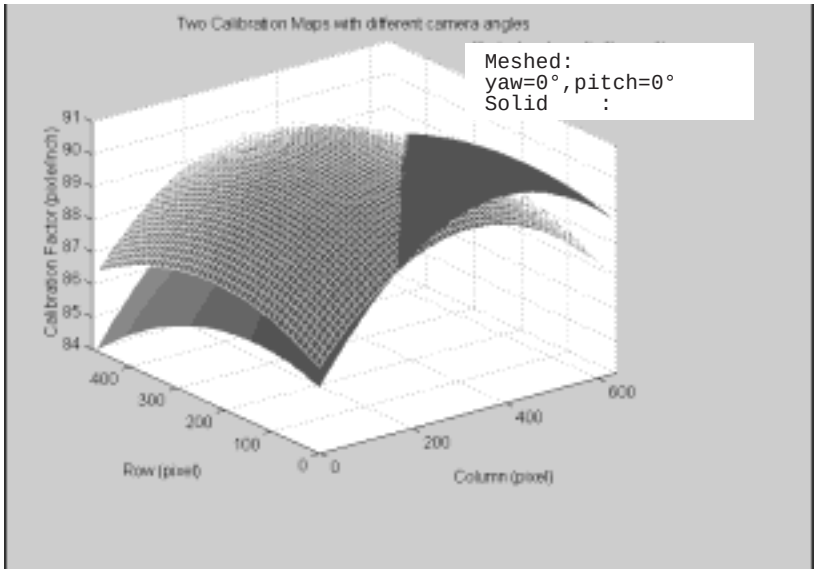

Figure 5-3. Two Calibration Maps Calculated from Different Camera Angle (meshed: yaw=0°,pitch=0°; solid: yaw=3°, pitch=-5°).

camera at yaw=+3° and pitch=-5°. The 49 data were fitted into a second-order polynomial (Eq. 4-1). With the new angle (3°,-5°), the original calibration map surface is rotated to form a new surface (shown as shaded surface in Fig. 5-4). As long as the yaw and pitch angles are within the allowable range (-5°,5°), a new calibration map can be established.

5.5 Testing the New Optical Inspection System

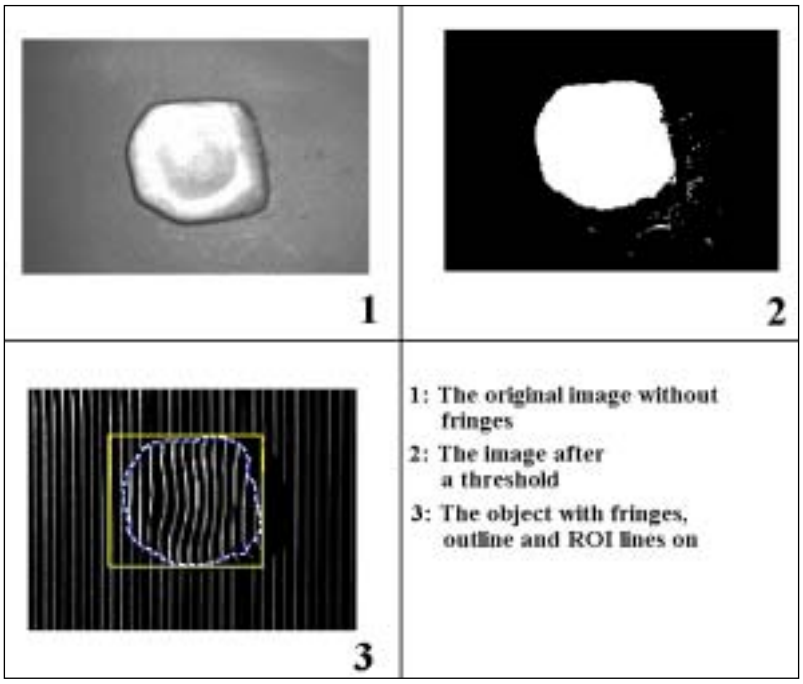

Figure 5-4. The Clay Model with and without Fringes.

In order to test the new optical inspection system, two tests on two different models were conducted.

5.5.1 The Result on a Smooth Surface of a Clay Model

Figure 5-4 shows the image of a clay model. In this figure, part (1) is the model without optical fringes, while the second one is its filtered image after a threshold.

The image with a threshold is then used to obtain the centroid, area and perimeter. The outline is also converted into the AOS signature for object recognition and ROI (region of interest) setting. The third one is the model with fringes on (outlines/edges, and ROI). The solid line is the ROI, and the dashed line is the outline of the model. The intensity values inside the ROI were fed to the inspection system to reconstruct the surface of the model for further analysis. Figure 5-5 shows the rendered reconstructed surface with surface fitting.

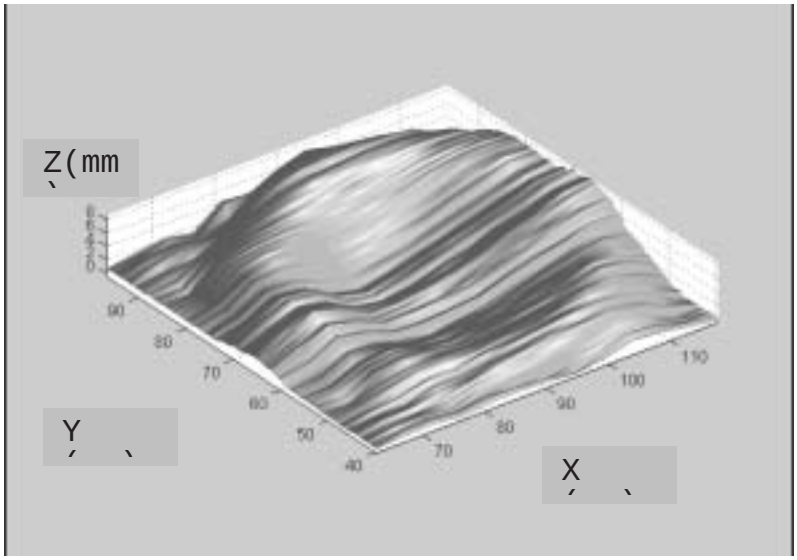

Figure 5-5. The Reconstructed Surface of the First Demonstration

5.5.2 The Result on a Discontinued Surface

The second model is a steel block with three step increases (see Fig. 4-3). The first bottom step with the thickness 0.1 inch is selected for testing. Optical fringes are projected onto the surface and the inspection platform.

Figure 5-6 shows the reconstructed surface of the model and inspection platform. Figure 5-7 shows the side view of the model. The line shown in Fig. 5-7 is along one of fringes (the 4^{th}), and the measured height increase is 0.09".

5.5.3 The Processing Time Using Different Algorithms

In order to know how much improvement the new algorithms have made over the old ones, the total CPU time for each sub-process were recorded for analysis. A total of 7.5 seconds time saving was achieved by running the new Fortran program as shown in Table 5-16.

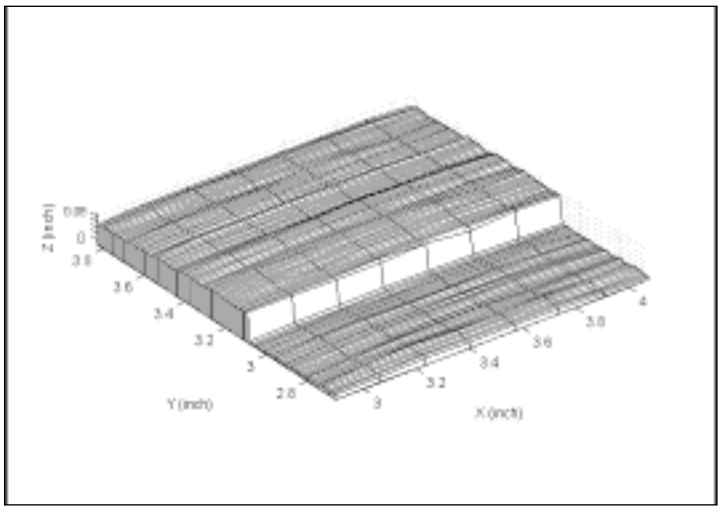

Figure 5-6. The Reconstructed Surface of the Steel Block and the Inspection Platform (exact height increase is 0.1").

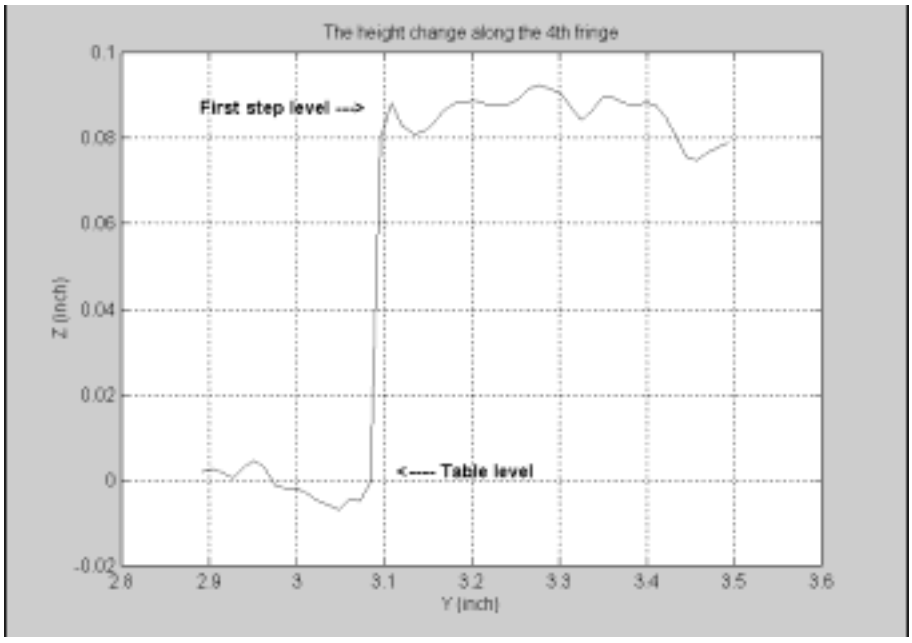

Figure 5-7. The Side View of the Reconstructed Surface along One of the Fringes (4^{th}).

In the new system, all of our programs were converted to Matlab script format. However, the Matlab uses an interpreting method to execute a program, which is a major

drawback. As shown in Table 5-17, a Matlab script needs 62.5 seconds to complete the 3-D calculation as compared with 55 seconds accomplished by the Fortran program.

Table 5-16 The Comparison of Processing time between Bisection and Analytical Solution Sub-pixel Algorithms

Machine: 486 66Mhz CPU PC

Finding Fringe centers

Program name	FR6.EXE	FR7.EXE
ROI (150x128)	1 min 10 sec	1 min 10 sec
Read image file	1 sec	1 sec
Input parameters	2 minutes	2 minutes
Find centers in Subpixel	8 sec	1 sec
Save positions	30 sec	30 sec
Combine data	1 minute	1 minute
Total	**4 min 49 sec**	**4 min 42 sec**

(1) 12 fringe, 150 row per fringe
(2) Time was measured under the MS-DOS environment
(3) FR6.EXE uses the bisection method to compute the fringe centers in subpixels
(4) FR7.EXE uses an analytical close-form solution to compute fringe center in subpixels
(5) Both programs require the operator to enter the system parameters
(6) Both programs are in Fortran source codes

Table 5-17 The Comparison of Processing time for 3-D Surface Reconstruction between Fortran program and Matlab Script

Task: 3-D Surface Reconstruction		
Software environment	**Fortran**	**Matlab**
Parameters entry	54 sec	54 sec
Calculation	1 sec	8.5 sec
Total	**55 sec**	62.5 sec

(1) ROI size: 14 fringes, 188 row per fringe
(2) Fortran program: FR7.EXE
(3) Matlab script: Fringe98a.m
(4) Both programs use the same algorithms

The last comparison is between the new and old inspection systems. This test was completely under the Matlab environment. The Matlab's Fuzzy Logic and Image Processing Toolboxes were used to conduct the test. The new system used the Fuzzy FCL (Fringe Center Locator) and the analytical close-form solution for fringe center detection. The old system used the modified FCL and the analytical close-form solution for fringe center detection. The modified FCL provides a graphical user interface (GUI). The user can click the mouse button to set the ROI and to obtain necessary parameters.

Table 5-18 shows that the new inspection system took 29.863 seconds to complete the entire process automatically. There were about 20 seconds spent on the fuzzy inference system, which also includes object recognition. In contrast, it took 36.349 seconds for the old system to do the same in Matlab. Although the new system was only 6.5 seconds faster than the old one, it could have been much faster if the Matlab scripts were converted into C/C++ source code. According to the information provided by

Matlab, the new system should only take about 0.453 seconds (about 70 times faster than the Matlab scripts). More importantly, the new inspection system is fully automated.

Table 5-18 The Comparison of Computing Time between the Old and New systems

Unit: second

	New system	Auto/Manual	Old system	Auto/Manual
Reading image	0.962	Auto	3.7550	Auto
Finding parameters		N.A.	180[+] (or 20[!])	Trial and Error
Setting a ROI		N.A.	10.0	Manual
Obj. Recognition	3.825	Auto		N.A.
Setting masks	0.851	Auto		N.A.
Fuzzy FCL	22.673 (or 0.4534*)	Auto		N.A.
S.D. FCL		N.A.	1.072	Auto
3-D reconstruction	1.552	Auto	1.522	Auto
Total time under DOS		N.A.	196.349[+]	Manual
Total time under Matlab (Windows)	29.863	Auto	36.349[!]	Manual
Total equivalent time under C/C++ (windows)	0.5972	Auto		N.A.

(1) A Pentium II 266Mhz PC was used.
(2) The ROI size is 215 pixel x 190 pixel (or equivalent to 12 fringes, 190 rows per fringe.

+: Under the DOS environment (Fortran).
!: Under Matlab 5.0 Windows environment (with Image processing toolbox 2.0).
*: It would have been only 0.4534 seconds if the Matlab scripts were

5.6 Discussions

(1) The results listed in Tables 5-1 to 5-5 show that the accuracy of the new FCL is as good as the old one, but the inspection process is fully automated.

(2) Averagewise, the difference of calculated fringe centers between the new and old systems is about 1 pixel. For a calibration factor of 83.3 pixel/inch as used in model 2 and the projection angle of 45°, the error in calculated height is translated into about 1/83.3" (= 0.012" or 0.306 mm). For a calibration factor of 260 pixel/inch as used for model 1, the error in height is about 1/260" (= 0.0038" or 0.097 mm).

(3) The designed rule base in the fuzzy inference system (FIS) is analogous to human's perceptions. Moreover, fewer rules reduce the computing time of the FIS. This rule base has proven correctly after testing three different models.

(4) The neural networks based object recognition is successful. From the results listed in Table 5-13, only 4 out of 32 objects that could not be recognized. With AOS signature algorithm, the accuracy of object recognition is much improved. The AOS signature recognition has a 100% success rate in the recognizing all of the 32 parts. When a part has the same shape, but different size, the AOS signature has the capability of recognizing them all.

(5) When a camera captures a circular calibration target with a non-zero angle, an ellipse will display on the screen. However, the ratio of R_{min} / R_{max} (i.e. minimum radius versus maximum radius) is always between 0.93 to 0.96. In other words, all of the calibration targets can be treated as circles. All 25 objects under tests were considered as circles, but with different areas.

(6) In terms of automatic camera angle detection, neural network did not work properly. The success rate was only 74%. This might be due to the input-to-output mapping that was not unique. This suggests that perhaps the number of output variables should be more than 1.

(7) The mathematical relationship or mapping between the circle's positions and the camera angle was established and employed in the new system. As soon as the hardware is set, the mapping can be formed immediately.

(8) A smaller ROI can reduce the calculation time without losing the accuracy.

(9) Although the old system (36.349 seconds of CPU time) has a similar performance as the new system (29.863 seconds of CPU time), it was not automated. It will take about 20 seconds for an experienced operator to input the necessary parameter values to begin the inspection process. For a beginner, it could take much longer. In contrast, the new inspection system is fully automated, and yields the same good measuring accuracy.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this work, some new algorithms have been developed and implemented in the new optical inspection system. Generally speaking, the test results are satisfactory. The conclusions of the research work are described below:

- The new inspection system is fully automated. When a part is under inspection, the inspection system can quickly recognize the object, check the shape, extract the 2-D features, and determine the 3-D geometry when necessary.
- The fuzzy-logic based fringe center locator (FCL) works very well. It can find fringe centers automatically, with the success rate of 100%.
- It is found that artificial neural network (ANN) could not recognize all kinds of objects. However, with help of the AOS (Angle of Sight) signature algorithm, the recognition rate became 100%.
- The new calibration procedure generates a more reliable calibration factor. Also, the camera angle can be monitored at any time. If the angle is off the exact value, but still within the allowable range, compensation can be made automatically. The capability of

detecting the camera angle gives an early warning to the system operator as to when the camera angle should be reset.

- The new system hardware is well arranged and compatible with the algorithms. The fringes from the new laser projector are sharp and focused, and their widths are suitable for 3-D geometry analysis.
- This system is able to complete a typical optical inspection in about 0.6 seconds if the Matlab scripts were converted into the C/C++ source code. This speed should be fast enough in most production lines.
- Matlab is a good piece of software for scientific programming. It is easy to write the script files, and works well in the Window's graphic user interface (GUI) environment. It has many toolboxes, such as neural networks, fuzzy logic and image processing, which helps in developing this new inspection system.

Future Work

Although this research is very successful, there are still some techniques that can further improve the system's functionality. Some future work is suggested as follows.

- Due to the use of the tiny laser projector, we can only project fringes onto a small area. A compact white light projector or a bigger laser projector may be used for future applications.
- The Matlab software cannot work directly with our image processing software (the Inspector), nor can it read an image directly from the frame grabber. It was necessary to save the captured image before using the Matlab software. A data communication module should be developed to capture images and automatically send them to the Matlab.

- The conversion of our Matlab scripts to the C/C++ source code is not easy to make, because the version number of these two languages has to be exactly right. Nevertheless, it is always desirable to make such a conversion if the Matlab scripts are to be used in a production line. It is not easy for the non-computer professional to do. The requirements for converting and compiling the Matlab script files are difficult to meet.

REFERENCES

Al-kindi, G. A. and Baul, R. M., "An Example of Automatic Two-dimensional Component Inpsection Using Computer Vision", Proc. Instn. Mech. Engr., vol. 205, pp71-83, 1991

Breuel, Thomas M., "Finding Lines under Bounded Error", Pattern recognition, vol. 29, No. 1, pp167-178, 1996

Bose, N. K. and Liang, P., "Neural Netowrk Fundamentals with Graphs, Algorithms, and Applications", p5-20, McGraw-Hill, Inc., New York, 1996

Brule, James F., "Fuzzy Systems – A Tutorial", http://www.austinlinks.com/Fuzzy/tutorial.html, Quadralay's Fuzzy Logic Archive, Quadralay Corporation. 1994-1996

Chen, B. T., Chen, Y. S. and Hsu, W. H., "Image Processing & Understanding based on Fuzzy Inference Approach", IEEE International Conference Proceedings on Fuzzy Systems, pp254-259, 1994

Cheng, F. H., Hsu, W. H. and Chen, C. A., "Fuzzy Approach to Solve the Recognition Problem of Handwritten Chinese Characters", Pattern Recognition, vol.22, #2, pp131-141, 1989

Craven, George M., "Object and Image: An Introduction to Photography", Prentice-Hall, INC., New Jersey, 1982

Firschein, Oscar, "The Image Understanding Program at ARPA", IEEE Expert, pp7-10, Oct. 1995

Hirota, K. and Nakagawa, Y., "Fundamentals of fuzzy knowledge base for image understanding", IEEE, pp1137-1142, 1995

Jain, R., Kasturi, R. and Schunck, B. G., "Machine Vision – Chapter 2, Geometric Properties", pp25-35, McGraw-Hill, INC., 1995

James, Mike, “Pattern Recognition”, A Wiley-Interscience publication, New York, Wiley, p19-34, 1988

Jang, Roger J., "ANFIS: Adaptive-Network-Based Fuzzy Inference System", Dept. of E.E and C. S, UC, Berkeley, CA 94720, 1993

Jang, J.S. Roger and Gulley, Ned, "Fuzzy Logic Toolbox for use with Matlab - Chapter 1: Introduction", pp1-14, The Mathworks Inc., 1995

Kasabov, N.K., "Hybrid fuzzy connectionist rule-based systems and the role of fuzzy rules extraction", Proceedings of FUZZ-IEEE/IFS '95, Fourth IEEE International Conference on Fuzzy Systems. Yokohama, Japan, IEEE Computer Society Press (1995) 49-56

Kawamura, A., Watanabe, N., Okada, H., and Asakawa, K., "A Prototype of Neuro-Fuzzy Cooperation System", pp1275-1282, IEEE, 1992

Kuo, Chan-teh, “Three-dimensional geometry measurement of a rotating object using an optical method”, Thesis (M.S. in M.E.)--Cleveland State University, 1993

Law, T., Ttoh, H. and Seki, H., "Image Occlusions as a Fuzzy Reasoning Problem", IEEE International Conference Proceedings on Fuzzy Systems, pp2045-2050, 1995

Lin, P. P., and Kuo, C. T., "A 3-D Surface Geometry Measurement Technique Using Fringe Projection", Lasers in Engineering, accepted for publication, 1998.

Lin, P.P. and Kuo, C. T., "An Efficient Optical Technique for Three-dimensional Geometry Determination", ASME Conference Proceeding in Manufacturing Science and Engineering, MED-Vol. 2-1/MH-Vol. 3-1, pp. 507-518, 1995.

Lin, P. P., Kuo, C. T. and Chawla, M. D., "Deformation Data Analysis for Dynamic Testing of F-16 Bias and radial Tires", Proceeding of 1st Int'l Symposium on Deformation and Fracture of Elastomer Composites, Paper No. 48, sponsored by American Chemistry Society, Rubber Division, 1997.

Lin, P. P., Parvin, F. and Schoenig, F.C. Jr., "Optical gaging of very short-term surface waviness", Machano-optics Laboratory, Advanced Manufacturing Center, Cleveland State University, 1992

Lin, P. P. and Parvin, F., "Edge Detection with Subpixel Resolution and its Application to Radius Measurement Via Fringe Projection Technique", SME 1990, MS 90-576

Moon, H. S. and Na, S. J., "A Neuro-Fuzzy Approach to Select Welding Conditions for Welding Quality Improvement in Horizontal Fillet Welding", Journal of Manufacturing Systems, Vol. 15, No. 6, 1996

Nakagawa, Y. and Hirota, K., "Fundamentals of fuzzy knowledge base for Image Understanding", IEEE, pp1137-1142, 1995

Nomura, H., Hayashi, I. and Wakami, N., "A Learning Method of Fuzzy Inference Rules by Decent Method", pp203-210, IEEE, 1992

Parker J. R., "Gray Level Thresholding in Badly Illuminated Images", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 8, August 1991

Pericles S. Theocaris, D. Sc., D. Appl. Sc, "Moire Fringes in Strain Analysis" pp1-10, Pergamon Press, 1969

Peng, T. M., Hubele, N. F. and Karady, G. G., "Advancement in the Application of Neural Networks for Short-term Load Forecasting", IEEE Transactions on power systems, vol. 7, No. 1, Feb. 1992

Pichumani, Ramani, " Construction of a Three-Dimensional Eometric Model for Segmentation and Visualization Of Cervical Spine Images", a dissertation submitted to the Program in Medical Informatics and to the Committee on graduate studies of Stanford University, 1996

Plassmann, P. and Jones, B. F., "An Aid for Reconstructing Damaged Sculptures by Estimating the Shape and Volume of a Missing Fragment", The journal of photographic science, vol,. 40, 1992

Pomerleau, D.A., "ALVINN: An Autonomous Land Vehicle in a Neural Network", Advances in Neural Information Processing System I, pp. 305-313. Morgan Kaufmann, San Mateo, CA, 1989.

Popovic, D. and Liang, N., "Fuzzy Apporach in Model-based Object Recognition", pp1801-1808, IEEE International Conference Proceedings on Fuzzy Systems, 1994

Rao, V. and Rao, H., "C++ Neural Networks and Fuzzy Logic", pp25-32, MIS Press, 1993

Russell D. Hoffman, "Internet Glossary of Statistical Terms: Histogram", http://www.animatedsoftware.com/statglos/sghistog.htm, 1998

Seitz, Peter, “Optical Superresolution Using Solid-State Cameras and Digital Signal Processing”, Optical Engineering, Vol. 27 No. 7, July 1988

Tissue, Brian M., "Lenses", http://www.scimedia.com/chem-ed/optics/lenses.htm Copyright 1996

Tsai, W. H. and Chou, C. L., "Detection of Generalized Principle axes in Rotationally symmetric Shapes", Pattern Recognition, vol.24 #2, pp95-104, 1991

Zadeh, L. A., "Fuzzy Sets", Information and control, vol. 8 pp338-353, 1965

Intelligent Robotics Laboratory, "Intelligent Soft Arm Control: 2-D Object Recognition", http://shogun.vuse.vanderbilt.edu/CIS/IRL/Projects/objrec.html, Vanderbilt University

"Machine Vision Courses over the Net", http://www.autovis.com/courses/course1/intro.html, Copyright © 1996 by Automated Vision Systems, Inc., Campbell, CA.

"Matlab v5.0 References", The Mathworks Inc., 1997

"Matlab compiler user's guide- chapter 3: getting started", pp3-4, The Mathworks Inc., September 1995

"Matlab Image Processing Toolbox User's Guide - Chapter 11: references", pp22-29, The MathWorks Inc., 1997

APPENDIX

APPENDIX A

(Data for Establishing AOS Objection Recognition Look-up Table)

Data collected from 3 standard parts for establishing of AOS signatures database

Part#1: circle (no corner)

Radius =>Average	Maximum	Minimum
46.0726	48.5164	43.5257
46.1464	48.6891	43.3234
46.1759	48.7978	43.1493
46.0964	48.8965	43.3146
46.0311	48.5500	43.1531
45.8044	48.5381	43.1731
46.2403	48.8626	43.4469
46.2424	48.9309	43.0489
46.0373	48.9310	43.4807

total = 9 samples

Part#2: rectangle (four corners)

Radius =>Average	Maximum	Minimum	|<=Angular distances between corners=>|			
65.7954	83.3363	48.2941	72.7407	106.9442	74.9774	105.3377
65.6742	82.9556	48.0788	72.0593	107.6127	74.7527	105.5752
65.8723	82.8277	48.3743	104.7330	72.1129	106.5901	76.5640
65.9842	84.0567	48.7679	75.1765	105.8583	70.6557	108.3094
65.6085	82.2145	48.2416	104.8051	72.3362	106.7772	76.0815
66.1025	84.1314	48.6168	75.2560	105.8353	70.4790	108.4298
65.7203	82.5219	47.9754	74.4963	106.6962	71.1781	107.6293
65.7104	82.9274	48.7431	105.0576	76.6382	104.1753	74.1289
65.7370	82.6580	48.1341	75.9565	106.2738	70.8949	106.8748

total = 9 samples

Part#3: triangle (three corners)

Radius =>Average	Maximum	Minimum	|<=Angular distances between corners=>|		
58.0982	96.0236	29.8407	208.5305	7.1350	144.3345
57.6442	94.8455	30.6935	210.5397	7.2610	142.1993
57.8152	94.7789	30.0198	210.3715	7.0777	142.5508
57.7529	94.6615	30.8156	6.3405	135.9577	217.7018
57.8072	94.5732	30.4072	209.0421	7.4112	143.5466
57.9138	95.2074	30.7801	5.6933	136.4579	217.8487
57.6195	94.6610	30.4422	210.0703	7.2518	142.6779
57.8606	95.4767	29.7759	215.3652	5.7665	138.8683
57.4872	94.1474	30.7056	210.2357	7.0801	142.6841

total = 9 samples

APPENDIX B

(Look-up Table Used in the Matlab Script)

Part #	**1**	**2**	**3**
Average radius	46	65	57
Normalized average radius	0.94 84	0.78 31	0.6
Maximum radius	48.5	83	95
Normalized maximum radius	1	1	1
Minimum radius	43.5	48.3	30.2
Normalized minimum radius	0.89 69	0.58 19	0.31 78
Number of corner	0	4	3
Inter-corner Distance (ICD) (degree)	Circle	71 74 104 106 Rectangle	7 142 210 Triangle

*Unit of average, maximum and minimum radii is pixel.
*Unit of Inter-corner Distances is degree

APPENDIX C

(Results of Finding Angular Distances for 32 Testing Objects)

Results of angular distances for 32 testing objects
Values are ascendingly sorted

10.4067	15.0382	34.7648	147.7929	151.9974
71.6932	75.3890	105.4741	107.4437	0
5.5785	137.7831	216.6384	0	0
0	0	0	0	0
5.6534	138.1138	216.2328	0	0
72.3261	75.3851	104.5773	107.7116	0
7.2418	143.9862	208.7721	0	0
0	0	360.0000	0	0
0	0	0	0	0
25.2660	26.3797	33.9003	91.3425	183.1115
0	0	0	166.0056	193.9944
26.0224	33.2331	37.4387	115.2650	148.0408
6.8315	20.0681	24.4219	114.4971	194.1814
8.7465	11.8943	66.2774	119.6664	153.4154
6.0147	146.0460	207.9392	0	0
0	0	0	0	0
70.3077	72.8710	107.6485	109.1729	0
0	0	0	0	0
5.4890	138.4924	216.0186	0	0
6.3057	136.0163	217.6780	0	0
73.5921	77.6005	103.3647	105.4427	0
12.4337	32.0896	34.7728	105.3484	175.3555
0	0	0	0	0
69.5986	73.5637	107.7966	109.0412	0
17.2889	42.2040	56.7122	91.1233	152.6716
6.8745	145.1478	207.9778	0	0
72.5137	75.3542	106.0131	106.1191	0
70.5474	75.4852	105.4343	108.5330	0
0	0	0	0	0
21.1367	21.1485	31.4197	142.3617	143.9334
72.8122	73.0149	106.6086	107.5643	0
13.3919	74.9055	86.9985	91.9386	92.7656
21.6409	21.8289	23.8580	137.6738	154.9983
17.8338	27.1575	28.3610	136.0684	150.5792
2.3443	140.9505	216.7052	0	0