ADAPTIVE MACHINE VISION FOR AUTOMOTIVE COMPONENT INSPECTION

ADAPTIVE MACHINE VISION FOR AUTOMOTIVE COMPONENT INSPECTION

By

KAI YANG, M. A. Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2008) McMaster University

(Mechanical Engineering) Hamilton, Ontario

TITLE: Adaptive Machine Vision for Automotive Component Inspection

AUTHOR: Kai Yang, M. A. Sc. (Xi'an Institute of Optics and Precision Mechanics of CAS, China)

SUPERVISOR: Dr. Gary M. Bone, Professor

NUMBER OF PAGES: xviii, 152

Abstract

Although the consistency, low cost, and high speed of machine vision systems make them suitable for many areas of manufacturing, there exist challenges for online inspection using machine vision systems due to the presence of variations in lighting, part position/orientation and part finish. In this thesis, a novel adaptive machine vision system based on pixel-by-pixel analysis and a neural network based vision system are presented to solve two challenging industrial inspection problems. Both of the vision systems stem from the idea of adaptive image processing to analyze an image with respect to its local properties.

In the first part of the thesis, a pixel-by-pixel analysis based adaptive vision system is designed for an automotive water pump housing surface inspection problem. This vision system is used to inspect the machined surface of a die-cast part. The defects on this part may include pores, dents and scratches. This problem is challenging for several reasons. First, non-uniform surface finish on the parts produces large brightness variations that must be reduced using a controlled lighting work cell and adaptive camera control. Second, the defects can be subtle and less than 1mm in diameter, while the surface is roughly 180mm by 110mm. Third, the surface often includes machining marks that appear similar to defects in the image, but are not considered defects. Finally, due to the manufacturing and fixturing variations, the size and location of the area to be inspected varies considerably, so that a simple fixed mask cannot be used to separate this area from the rest of the image. These challenges have been overcome by developing an adaptive machine vision system. This system includes custom-designed controlled lighting, and several software algorithms for adapting to the variations in surface quality and geometry. The system can detect defects as small as 0.15 mm. It has been tested with over 1,700 images that were collected at the factory. The majority of the defects were pores. These pores were correctly classified in 93% of the cases.

In the second part of the thesis, a neural network-based vision system is developed for an automotive beam clip present/absent inspection problem. In this

inspection problem, it is difficult to obtain the theoretical expression for the conditions of 'clip present', 'clip absent' and also for the clip orientation. Furthermore, there exist strong variations in this inspection problem, such as changing lighting conditions, environmental disturbances, clip locations and clip orientations. A CMAC neural network-based vision algorithm is developed to overcome these challenges. The CMAC neural network has the ability to learn fast and is suitable for real-time inspection applications. This vision system is demonstrated to correctly classify 100% of the cases, for the given automotive part images, after being trained for 151 seconds.

Acknowledgements

I would like to take this opportunity to thank the many individuals who have made the completion of this thesis possible.

First and foremost, I would like to express my sincere gratitude to my research supervisor, Dr. Gary M. Bone, for offering me the opportunity to pursue graduate studies, and for his valuable guidance, encouragement, patience and support. I would also like to thank the group members in AUTO 21 E202 project. I also thank my colleagues and friends for their advice and encouragement. Special thanks to Mr. Scott Spence for his aid in the field trip at Orlick Ltd. Finally, I would like to specially show appreciation to my family for their support.

Table of Contents

Abstract iii

Acknowledgements v

Table of Contents vi

List of Algorithms x

List of Figures xi

List of Tables xv

Abbreviations xvi

Nomenclature xvii

Chapter 1 Introduction 1

1.1 Preface 1

1.2 Objective and Organization of the Thesis 2

Chapter 2 Literature Review 3

2.1 Introduction 3

2.2 Machine Vision Based Inspection Techniques 3

2.3 Image Acquisition 6

2.3.1 Camera Calibration 6

2.3.2 Lighting Condition and Camera Control 7

2.4 Related Machine Vision Techniques 8

2.4.1 Adaptive Thresholding 9

2.4.2 Edge Detection 10

2.4.3 Region Growing 10

2.4.4 Image Registration 11

2.4.5 Image Classification 11

2.5 Summary 13

Chapter 3 Adaptive Machine Vision System for Surface Inspection: Image Acquisition 14

3.1 Introduction 14

3.2 Problem Description 14
3.3 Digital Camera Selection 20
3.4 Controlled Lighting Work Cell 21
3.5 Camera Calibration 26
3.6 Adaptive Camera Control 32
3.6.1 Algorithm Overview 32
3.6.2 Assumptions 34
3.6.3 Algorithm 35
3.7 Experimental Results 36
3.8 Summary 39
Chapter 4 Adaptive Machine Vision System for Surface Inspection: Region of Interest Masking 40
4.1 Introduction 40
4.2 Generating the ROI Mask using Seeded Region Growing 41
4.2.1 Algorithm Overview 41
4.2.2 Assumptions 42
4.2.3 Mask Generation by Seeded Region Growing Algorithm 42
4.3 Image Registration 46
4.3.1 Overview of Both Image Registration Algorithms 47
4.3.2 Assumptions 49
4.3.3 Feature Detection 50
4.3.4 Feature Matching 65
4.3.5 Motion Estimation 65
4.3.6 Mask Registration 69
4.3.7 Summary of the Accurate Image Registration Algorithm 71
4.3.8 Simplified Image Registration Algorithm 73
4.4 Summary 76
Chapter 5 Adaptive Machine Vision System for Surface Inspection: Part Classification 77

5.1 Introduction 77
5.2 Detection of Defective Pixels 78
5.2.1 Overview 78
5.2.2 Gradient Operator Followed by Global Thresholding 79
5.2.3 Adaptive Local Thresholding of the Original Image 82
5.2.4 Local Standard Deviation Followed by Global Thresholding 85
5.2.5 Local Standard Deviation Followed by Adaptive Global Thresholding 89
5.3 Removal of Falsely Classified Edge Pixels 91
5.3.1 Algorithm Overview 91
5.3.2 Assumptions 91
5.3.3 Algorithm 92
5.4 Defect Size Measurement 95
5.4.1 Algorithm Overview 95
5.4.2 Assumptions 95
5.4.3 Algorithms 96
5.5 Sealing Area Identification and Defect Classification 100
5.5.1 Algorithm Overview 100
5.5.2 Assumptions 102
5.5.3 Algorithm 102
5.6 Experimental Results 105
5.6.1 Timing Results 105
5.6.2 Inspection Results 107
5.7 Summary 117
Chapter 6 CMAC Neural Network Based Machine Vision System 118
6.1 Introduction 118
6.2 Problem Description 118
6.3 CMAC Neural Network 120
6.3.1 Basic CMAC Module 120

6.3.2 Complete CMAC Module 122
6.3.3 Adjustment of the Weights 124
6.4 CMAC Based Machine Vision System 126
6.4.1 Overview 126
6.4.2 Assumptions 127
6.4.3 Image Preprocessing 128
6.4.4 System Training 132
6.5 CMAC Software and Testing Procedure 136
6.6 Experimental Results 139
6.7 Summary 141
Chapter 7 Conclusions and Recommendations 142
7.1 Summary 142
7.2 Achievements 142
7.3 Recommendations for Future Work 143
Reference 145

List of Algorithms

Algorithm 3.1: Adaptive Camera Control.....35
Algorithm 4.1: Mask generation by seeded region growing.....42
Algorithm 4.2: Accurate Image Registration.....71
Algorithm 4.3: Simplified Image Registration.....75
Algorithm 5.1: Local Standard Deviation followed by Adaptive Global Thresholding.....90
Algorithm 5.2: Removal of Wrongly Classified Edge Pixels.....92
Algorithm 5.3: Clustering of Defective Pixels.....96
Algorithm 5.4: Defect Size Measurement.....98
Algorithm 5.5: Sealing Area Identification and Defect Classification.....102
Algorithm 6.1: CMAC Based Machine Vision Training Algorithm.....132

List of Figures

Figure 3.1 The automotive water pump housing part to be inspected.....15
Figure 3.2 An example of a good part.....16
Figure 3.3 A defective part with an obvious defect and machining marks.....17
Figure 3.4 A defective part with porosity and machining marks.....18
Figure 3.5 A defective part with pores and a subtle dent.....19
Figure 3.6 Picture of the selected PGR Scorpion SCOR-14SOM monochrome camera.[63].....20
Figure 3.7 Image taken under ambient lighting condition.....22
Figure 3.8 Controlled lighting work cell.....22
Figure 3.9 Controlled lighting work cell frame.....23
Figure 3.10 Diagram of the imaging geometry.....25
Figure 3.11 Image of the chessboard used for the camera calibration.....27
Figure 3.12 Result produced by the function cvFindChessboardCorners().....28
Figure 3.13 Diagram of the adaptive camera control.....33
Figure 3.14 Image of Part A taken using fixed camera parameters.....37
Figure 3.15 Image of Part B taken using fixed camera parameters.....37
Figure 3.16 Image of Part A taken using adaptive camera control.....38
Figure 3.17 Image of Part B taken using adaptive camera control.....38
Figure 4.1 Mask image generated by using the OpenCV library.....45
Figure 4.2 Mask image generated by using the seeded region growing algorithm.....45
Figure 4.3 Hole #1 - Hole #6 are features used in image registration.....48
Figure 4.4 Example result 1 using Hough transform.....51
Figure 4.5 Example result 2 using Hough transform.....51
Figure 4.6 Hole pattern used for image registration.....52
Figure 4.7 Binary images to demonstrate the searching procedure.....53
Figure 4.8 Several cases in pattern matching procedure when the hole in the pattern image is larger than the hole in the image to be processed.....54

Figure 4.9 Pattern matching results when the size of the hole in the pattern image is close to the actual holes in the image.....55
Figure 4.10 Pattern matching results when the size of the hole in the pattern image is larger than the actual holes in the image.....56
Figure 4.11 Pattern matching results when the size of the hole in the pattern image is smaller than the actual holes in the image.....56
Figure 4.12 Four-level pyramid of an example surface image.....58
Figure 4.13 Four-level pyramid of the pattern image.....59
Figure 4.14 Histograms of the sub-image.....63
Figure 4.15 Comparison of three methods to covert the sub-images to binary sub-images.....64
Figure 4.16 Spatial relationship between the registered pixel and the pixels on the current mask image.....70
Figure 4.17 Result with unregistered standard mask.....72
Figure 4.18 Result with registered current mask.....72
Figure 4.19 Simplified image registration.....74
Figure 4.20 Result using simplified image registration algorithm.....76
Figure 5.1 Example of an ROI to be inspected.....78
Figure 5.2 Result by applying Sobel gradient operator.....81
Figure 5.3 Result by applying Laplace differential operator.....81
Figure 5.4 Result by applying Canny edge detector.....82
Figure 5.5 Result by local thresholding using a threshold equal to the local mean.....84
Figure 5.6 Result by local thresholding using a threshold equal to a local weighted sum of the local minimum and maximum.....84
Figure 5.7 Result by local thresholding using a threshold equal to the local median.....85
Figure 5.8 Diagram of a 3 by 3 local window[74].....86
Figure 5.9 Standard deviation image.....87

Figure 5.10 Thresholded standard deviation image using a small global threshold....88
Figure 5.11 Thresholded standard deviation image with a large global threshold....88
Figure 5.12 Segmented ROI obtained by local standard deviation followed by adaptive global thresholding method....89
Figure 5.13 Mask contours shown in red, overlaid on a gray scale image of Surface #1....92
Figure 5.14 Segmented image with wrongly classified pixels....94
Figure 5.15 Segmented image after applying Algorithm 5.2....94
Figure 5.16 Clustering result obtained using Algorithm 5.3....97
Figure 5.17 Result with measured sizes of the potential defects....99
Figure 5.18 Sealing edge shown in red....101
Figure 5.19 Sealing area identification and defect classification result....104
Figure 5.20 Inspection results for defects close to the edge of Surface #1....110
Figure 5.21 Inspection results for the same automotive parts shown in Figure 5.20....110
Figure 5.22 Result where obvious machining marks are wrongly detected as defects....113
Figure 5.23 Result where obvious machining marks are ignored....114
Figure 5.24 Result where subtle dent is partially detected....115
Figure 5.25 Result where subtle dent and pores are missed....116
Figure 6.1 Automotive beam with clip present....119
Figure 6.2 Automotive beam with clip absent....120
Figure 6.3 Diagram of the basic CMAC mapping....121
Figure 6.4 Diagram of the complete CMAC mapping....124
Figure 6.5 Block diagram of the CMAC based machine vision system....127
Figure 6.6 The shape of Gaussian distribution used for filtering....130
Figure 6.7 Four level Gaussian pyramid....131

Figure 6.8 Comparison between the original and condensed images.....132
Figure 6.9 Training curve.....134
Figure 6.10 Testing curve.....135
Figure 6.11 Used memory curve.....140

List of Tables

Table 3.1: Lens parameters.....25
Table 3.2: Typical calibration errors.....31
Table 4.1: Example results for motion estimation.....69
Table 5.1: Timing Results.....106
Table 5.2: Definition and source of measured quantities listed in Table 5.3.....108
Table 5.3: Inspection results.....109
Table 6.1: Experimental results.....139

Abbreviations

ANN	artificial neural network
CMAC	cerebella model articulation controller
EKF	extended Kalman filter
ESN	echo state network
GPD	Gaussian pyramid decomposition
K-L transformation	Karhunen-Loeve transformation
LMS	least mean square
MLP	multiple layer perceptron
NN	neural network
PTM	polynomial texture map
RBF	radial basis function
ROI	region of interest
SVD	singular value decomposition

Nomenclature

f	Focal length of the lens
d_{object}	Distance from the lens to the object
v	Distance from the image to the lens
l_1, l_2	Object size and image size
(x_i, y_i)	Coordinates of the chessboard corners in the real world coordinate system and in the image coordinate system
(x_i', y_i')	Coordinates of the chessboard corners in the image coordinate system
$\mathbf{H}$	Camera extrinsic matrix
$\mathbf{H}_1$	Transformation matrix from the real world coordinate system to the image coordinate system
$\mathbf{H}_2$	Transformation matrix from the image coordinate system to the real world coordinate system
d_{rowmin}	Distance between the neighbour pixels in the same row
d_{colmin}	Distance between the neighbour pixels in the same column
$\overline{p}$	Average intensity
N_{mask}	Total number of pixels with binary value of 1 in the mask image
p_k	Gray scale value of a masked pixel in part image
c_{i_0, j_0}	Difference between the pattern and the sub-image
h, I	Pixels from the grayscale image of the hole pattern image and the image to be processed
d_{hole}	Diameter of the hole in pixels
$\overline{x}$, $\overline{y}$	Calculated coordinates of the centre of mass in pixels
$\mathbf{A}$, $\mathbf{B}$	Matrices used to estimate the affine motion model
$\mathbf{X}$	Motion vector

$\mathbf{A}^+$	Pseudo-inverse matrix of $\mathbf{A}$
Δx, Δy, $\Delta\theta$	Parameters for the simplified motion estimation model
$dst(x, y)$	Gray scale value of the pixel in the result image
$src(x, y)$	Gray scale value of the pixel in the source image
$\mathbf{M}_h$	Horizontal Sobel kernel
$\mathbf{M}_v$	Vertical Sobel kernel
$\mathbf{M}_l$	Laplace kernel
mean	Mean value
var	Variance value
std_dev	Standard deviation value
d_E	Euclidean distance
D_{image}	Diameter in pixels for sealing area identification
$\mathbf{s}$	State vector in the input state space
C	Number of memory addresses corresponding to $\mathbf{s}$ in CMAC
A_i	The *i*th memory address in CMAC
$W(A_i)$	Weight stored at address A_i
$\|\bullet\|$	L^2-norm of the vector
$f(\mathbf{s})$, $f_d(\mathbf{s})$	Measured and desired output of CMAC
β, β_1, β_2	Parameters to adjust the weights in CMAC
$g(\bullet)$	Function of 2-D Gaussian probability distribution

Chapter 1
Introduction

1.1 Preface

Machine vision systems are applied in many manufacturing operations. They have the advantages of consistency, low cost, high speed, safety, and ease of maintenance. However, the online inspection of manufactured parts by machine vision remains a difficult and largely unsolved problem.

Strong variations exist in industrial inspection problems, such as changing lighting conditions, part position/orientation variations and part finish variations. With automotive parts another difficulty is finding small and sometimes subtle defects on large surfaces. The presence of these challenges requires that machine vision systems have the ability to adapt in order to perform well.

Research on adaptive machine vision systems has been ongoing for decades. A typical adaptive machine vision system includes an image acquisition unit and related adaptive image processing software algorithms and hardware. High quality cameras and lens are now available; and many adaptive image-processing algorithms have been developed. The recent availability of low-cost high performance computers enables the implementation of increasingly complicated image processing algorithms for real-time applications. It is an ideal time to make advances on challenging industrial inspection problems.

There are two main adaptive machine vision systems: pixel-by-pixel analysis based adaptive machine vision systems and neural network based machine vision systems, which are suitable for different vision problems. Both vision systems analyze the local properties of the current processed image. The pixel-by-pixel analysis based vision system is fit for vision problems that can be defined by parametric models while the neural network based technique works well for problems that are nonparametric. Both types of systems will be investigated in this thesis.

1.2 Objective and Organization of the Thesis

The objective of this thesis is to develop two different adaptive machine vision systems (including the required algorithms) for different automotive component inspection problems. First, a pixel-by-pixel analysis based adaptive machine vision system will be developed in detail, including the hardware design and the software design. Next, a neural network based machine vision system will be developed. Both adaptive machine vision systems will be tested experimentally.

The organization of the thesis is as follows. In Chapter 2 the literature related to adaptive machine vision is reviewed. The pixel-by-pixel analysis based adaptive machine vision system for inspecting a machined surface is described in Chapters 3 - 5. In Chapter 3, the hardware and software for the image acquisition procedure to obtain images with high quality is discussed, including the design of a controlled lighting work cell, the selection of a digital camera, and a novel adaptive camera control algorithm. In Chapter 4, an image masking technique is developed to segment the region of interest (ROI) from the remainder of the image to prevent unimportant areas of the surface from being inspected. The algorithms to size, locate and classify the defects are presented in Chapter 5. In Chapter 6, a neural network based machine vision system is developed for an automotive clip present/absent inspection problem. The algorithm details and experimental results are included in this chapter. The achievements of this research are summarized in Chapter 7. Recommendations for future work are also presented in this final chapter.

Chapter 2

Literature Review

2.1 Introduction

In this chapter the research literature related to the adaptive machine vision systems for inspection problems will be reviewed. This field has been developing for decades and the literature is quite rich. The following aspects will be reviewed: machine vision based surface inspection techniques for manufactured parts, image acquisition, and related machine vision techniques.

2.2 Machine Vision Based Inspection Techniques

Product inspection is very important in industry and manufacturing. There are a lot of techniques used for inspection problems, such as ultrasonic [1], laser beam [3], eddy current [4], x-ray [13], machine vision, and hybrid techniques [8][12][14]. Within the applied techniques, machine vision based inspection systems have the advantages of being fast, accurate, non-contact, safe and low cost. With the development of improved computer hardware and software, vision based inspection systems are becoming more and more popular in industry and manufacturing.

In 1984, Okawa systematically described the design of a vision based inspection system for cast pulleys [2]. In [2], several important problems that should be considered in the vision system design were summarized, including: multiple views from different angles, software execution time, noise removal, and defect classification rules. The proposed inspection process was analogous to that of human judgment. To detect the pores on the surface, the local image parameters were calculated to generate a parameter for separating defective and non-defective areas. The five candidate parameters were the standard deviation, the average of the difference between the ideal surface and the actual surface, the average of the smoothed difference, the ratio of the averages of the peaks and valleys in a gray level profile, and the ratio of the maximum and minimum of the

smoothed gray level. These parameters were used to obtain the optimized set-up of the system hardware and make the defects and non-defective areas more distinguishable. In his proposed vision system, the method to adaptively adjust the hardware was not mentioned. The images used in this inspection system were also very small (192 by 192 pixels^2). So the computational burden was not a significant problem for his system.

In [5], Fernandez *et al.* described the hardware design for an automated vision inspection system. The vision system was used for on-line detection and analysis of surface defects in a continuous flat metallic production manufacturing process. They modeled the roughness of the surface as a zero mean normal distribution. With this model, different lighting patterns were analyzed and the rules to choose the light source were proposed. Furthermore, they also proposed a criterion to choose the image acquiring equipment. The algorithms to process the acquired images were only introduced briefly. The detected defects were classified by a rule-based approach.

In [6], the defects were detected by analyzing the textures on the surface. Instead of a single camera, a camera matrix was applied to acquire high-resolution images over a large area in the vision system proposed by Zeichen, Hufnagl, and Berger. To reduce the inspection time, two techniques were applied. One was to preprocess the obtained images. The feature vectors were extracted during the image preprocessing procedure. The trainable textures were described by the extracted feature vectors, which were used to train the classifier. The second technique was to apply a transputer system with a high performance transfer bus to analyze the data in parallel. They showed example inspection results for a cast aluminium valve lid.

In [7], Platero *et al.* described the artificial intelligence based vision inspection techniques used with the hardware presented in [5]. In this paper, more complicated and accurate algorithms were applied to extract the feature vectors of the acquired images. The feature vectors were classified by means of rule-based systems along with data-based classifiers. At the end of this paper, a hybrid system was recommended to achieve better inspection results. They claimed to achieve 99.7% success rate.

In [9] and [11], Tsai *et al.* applied Fourier transforms to analyze the obtained images for defective textures. In [9], a two-dimensional Fourier transform first was computed for the input image. Then the features derived in the spatial-frequency domain were input to a classifier for further processing. Two kinds of classifiers were used in the proposed vision system: Bayes classifier and neural network based classifier. The reported results demonstrated high performance (100%) in classification accuracy rates. In [11], Tsai and Huang applied the Fourier transform and inverse Fourier transform to extract the statistical features of the inspected surface. This allowed them to simplify the vision based inspection problem into a simple thresholding problem. They included example results for various textured surfaces with and without defects.

Recently, Mery *et al.* proposed a novel inspection algorithm for aluminium castings in [10] and [17] that was named automated multiple view inspection. With comparison to the traditional methods, the proposed inspection algorithm utilized images of the same area from multiple views for inspection. This imitates the natural human inspection procedure. When a human being inspects objects, he always changes the views to detect the region of interests. In these papers, images from different views were analyzed to detect the potential defects. After processing the individual images, all the images were registered. Only the defects that were detected in all views were reported as real defects. The redundant images from different views improved the inspection accuracy. However, the accuracy of the image registration is critical for the final inspection results and the computational cost appears to be very high.

In **[15]**, Steiner and Katz proposed a 2.5D model of porosity flaws on machined surfaces. This 2.5D model was based on the contour lines from binary images obtained using different thresholds. They assumed that the contour obtained with a threshold close to black measures the pore at a greater depth than a threshold close to white. They described two methods to detect pores close to the edges of the surface. The first method used curve analysis to locate the pores. The second method applied basic morphological operations to determine the relationship between the pores and the edges. The images of the pores were taken with an overhead camera. A second camera was used to distinguish

the pores and the stains or scratches on the surface. Steiner and Katz claimed their algorithms improved the accuracy of porosity size measurement, but no quantitative evaluation was provided.

In [16], Gamage and Xie proposed a real-time vision system for defect inspection. The vision system was applied to inspect the surface produced by a cast extrusion manufacturing process. The difficulty of the inspection problem is the small size of the defects (diameter as small as 480μm); and the large size and fast motion of the inspected surface (2m in width moving at the speed of 50m/s). A custom designed light source was applied to highlight the possible defects. The utilized light source was referred to as Mie light scattering. With a series of image processing instructions such as image smoothing, histogram based image thresholding, noise removal, object labelling, image classification, the image was categorized as non-defective or defective. Furthermore, the area inspection method and the line inspection method were compared in this paper. They concluded that a vision system with a stationary area inspection camera has a better inspection accuracy than the one with a traversing line camera (80% vs. 76%).

2.3 Image Acquisition

A machine vision system is based on the analysis of images. So obtaining images with high quality is a key problem for a machine vision system. Two key factors for image acquisition are the digital camera (including the lens) and lighting.

2.3.1 Camera Calibration

Camera calibration is an essential component of most machine vision systems. Many calibration algorithms have been developed in machine vision recently [18] - [24]. All these techniques can be classified roughly into two categories: photogrammetric calibration [18][20][24] and self-calibration [19][21][22][23]. The photogrammetric calibration category is the traditional one. The camera is calibrated by observing a calibration object with precise 3D geometry. The approaches using the photogrammetric calibration can be very effective, but can also require an expensive calibration apparatus and a complicated setup.

No calibration object with precise 3D geometry is needed with the self-calibration methods. When using self-calibration methods to calibrate the camera, the camera moves in a static scene. Two constraints are utilized: the fixed internal parameters of the camera and the rigidity of the scene. Both the internal and external parameters of the camera parameters can be recovered by applying the detected correspondence between images. While this approach is very flexible, it is not robust. The obtained results are often unreliable with many parameters to be estimated and the calibration is very sensitive to the image noise.

2.3.2 Lighting Condition and Camera Control

Besides the digital camera, the lighting condition is a fundamental factor in image acquisition. With image acquisition the sensor in the digital camera senses the light reflected by the objects. Different images are obtained under different lighting conditions. In the other words, an image is a description for a certain lighting condition. Based on this idea, a novel technique named computational photography was developed recently. The term computational photography was first used by Mann in [27]. In contrast with digital imaging, computational imaging integrates sensing and data processing to output an image with greatly improved quality. This technique has great potential in image processing, computer graphics, applied optics and machine vision.

Computational photography covers the subject areas: computational illumination [25][26][28], computational optics [29][30][31], computational processing, and computational sensors. Since only the first two topics are related to this research, only the literature about them will be reviewed.

When using computational illumination, the photographic illumination is controlled in a custom structure. A number of images are obtained under different lighting conditions. By processing the captured images, mappings between the illuminations and the images can be acquired. Finally, an image with improved quality will be generated from the images and the mappings. The techniques can be applied in image-based relighting, image enhancement, geometry/material recovery and so on.

There were a number of mappings developed. In 1978, Blinn proposed a mapping named bump mapping (BM) [25][26]. This mapping tries to describe the relationship between the illumination and the surface normals of the local reflecting surface when the light is moved around the object. It is easy to use with a low computational cost. But it is very difficult to generate BMs for manufactured parts since their surfaces contain both dull and shiny patches. This approach is more popular in computer graphics.

Malbender proposed the polynomial texture maps (PTM) in 2000 [28]. In contrast with the BM method, the light was fixed in the PTM method. The light was composed by a number of bulbs that are fixed in a specific structure fashion. The bulbs were turned on and off one by one to change the illumination. A group of images were obtained under the different lighting conditions. The images were analyzed to obtain the PTMs. Finally, an enhanced image was generated from the images and the acquired PTMs. The disadvantage of this method is an elaborate experimental set-up is needed.

It is well known that a different aperture size changes the quantity of light sensed by the digital camera. If the coded aperture is used in the image acquisition procedure, the images obtained will have improved quality. In [29], the shape of the aperture was changed by inserting a encoded mask between the lens and the aperture. By applying the relative decoding and image processing algorithms, the quality of the image was greatly improved. In [30], the aperture was designed in the specific encoded shape in contrast with the conventional aperture.

Similarly, different exposure or shutter time will generate different effects in the obtained image. There also are many approaches about improving image quality by changing the shutter time. These approaches are widely applied in image deblurring. In [31], a technology based on the idea mentioned above was developed. By controlling the shutter time, the motion blur caused by the camera motion or object motion was reduced and the image was improved significantly.

2.4 Related Machine Vision Techniques

Machine vision is the application of computer vision in industry. Besides processing the digital images, machine vision involves interfacing with other industrial or

manufacturing equipment such as robots. There is extensive literature about the topic of machine vision. In this thesis, only the literature related to the surface inspection problems is reviewed, including edge detection, image segmentation, image registration, and image classification.

2.4.1 Adaptive Thresholding

Image segmentation is an important topic in image processing. There are four main ways to segment the image into objects and background: threshold techniques, boundary-based techniques, region-based methods and hybrid techniques. Thresholding is the simplest and most fundamental method in image segmentation. Conventionally, a fixed threshold is applied to segment the whole image. In practice, a global threshold is often not suitable for most images. As a result, many adaptive thresholding methods have been proposed. The main approaches employed in adaptive thresholding techniques are: Chow and Kaneko's approach [32][33]; and local thresholding methods [34][35][36][37][39].

In 1972, Chow and Kaneko proposed their famous adaptive thresholding algorithm [32]. When applying this technique, the image is divided into an array of overlapped sub-images. For each sub-image, the histogram is calculated. By fitting the histograms to bimodal ones the thresholds can be determined. This method can choose the thresholds dynamically and acquire good segmentation results for bimodal histograms. But the computational burden of this approach is heavy since it requires a histogram fitting procedure. The generation of the sub-image array also makes this method inflexible.

In 1979, Nakagawa and Rosenfeld further developed Chow and Kaneko's method [33]. They extended the algorithm to allow the histogram to be trimodal. This makes it possible to develop algorithms about segmenting the image in multiple scales. But their improved algorithm was also sensitive to the shadows in an image.

Local thresholding is a method to choose the threshold pixel by pixel with respect to its neighbours. References [34][35][36][37][39] described several different ways to choose the local thresholds. In [34], the average of the gray scale values in some

neighbourhood around the checked pixel is considered as the threshold. The validation of this method was demonstrated by Venkateswarluh and Boyle [37]. Bernsen used another method to choose the threshold pixel by pixel [36]. The mean of the minimum and the maximum gray scale values was applied as the threshold. In [35], the methods in described in [34] and [36] were combined together to improve the results by introducing an ad-hoc parameter. The adjustment of the ad-hoc parameter made this threshold choice flexible to meet different problems. Methods based on the median, combination of mean and standard deviation value were introduced in [39]. A survey of adaptive thresholding algorithms was presented in [38], where many approaches were discussed and evaluated.

2.4.2 Edge Detection

Segmenting objects via their edges is a widely used technique in machine vision since edges always exist as features of the objects and the data to describe the edges are always less than the original image data. For edge detection, the Sobel operator is a well-known edge detector [41]. It is a discrete differential operator. The Prewitt operator, which is also a differential operator, is also commonly applied as an edge detector [40]. These operators are very sensitive to the noise. Canny proposed a multi-stage algorithm to detect the edges in an image, the Canny edge detector [42]. In the Canny edge detector, the idea of optimization was introduced to satisfy the defined edge detection criterions, which is very important in image processing.

2.4.3 Region Growing

Region growing is another important method to segment images into objects and background. Actually, region growing is an extension of image thresholding. Not only the brightness of the pixels but also the connectivity between pixels are considered in the segmentation procedure. In [43], Adams and Bischof modeled region growing mathematically. In their algorithm, a set of points, or "seeds", were needed. They also proposed methods to choose the required seeds and threshold. They applied both semi-automated and automated methods to choose the seeds and the thresholds, and obtained satisfactory results.

In [44], a hybrid method by combining threshold technique and region growing method was proposed to detect the abnormalities in images of the sun. The image thresholding was applied as the seed detector in this approach. The image first was roughly segmented. Then the seeded region growing technique was used to cluster the segmented objects.

2.4.4 Image Registration

Image registration is another important topic in image processing. It is widely used in medical, remote sensing, industrial, and manufacturing applications. There are four main steps in image registration: feature detection, feature matching, mapping functions, resampling. The integrated literature was introduced in [45]. Many approaches about these four areas were discussed and evaluated. Sinha and Wu proposed a fast image registration algorithm in 2007 [46]. In their algorithms, the images were registered by translating and rotating first. Next they considered the edges in the images as the features. Then the features were applied to refine an affine model to estimate the motion between images. When solving the optimization problem, the gradient descent algorithm was used for fast convergence.

2.4.5 Image Classification

There are two main kinds of classification methods applied in machine vision inspection systems: rule-based classification and texture-based classification. If the precise definition of the classification rules is easy to obtain, a rule-based classifier is suitable for the inspection system, e.g. [5][7][10][11][16][17]. Otherwise, the texture-based classifier is a good choice, e.g. [2][6][7][9][56][57][59][61].

In machine vision, neural networks are often used as texture-based classifiers. Neural networks (NN), also known as artificial neural networks (ANN) are widely applied in control, signal processing, image processing, and computer vision. They are based on the biological neural networks and are used to analyze complicated nonlinear problems. A comprehensive introduction to NN is presented in [58]. There are many types of NN, such as Multiple Layer Perceptron (MLP) [48], Cerebella Model

Articulation Controller (CMAC) [49], Radial Basis Function (RBF) [53], Extended Kalman Filter (EKF)[55], and Echo State Network (ESN)[60].

Most of the NNs have complicated structures and the computational burden is very heavy. In the 1970s', Albus proposed a simple and fast NN structure he termed CMAC [49]. The CMAC neural network is based on the table look-up technique. It has the abilities to learn fast and to generalize. By building a hashing table [50], the space to store the weights is reduced to an acceptable and practical number. Miller further developed Albus's approach and applied it to robotic control [52]. Miller summarized the detailed theory of CMAC in [54].

CMAC neural networks have also been used for computer vision applications [56][57][59][61]. They were applied as classifiers in these four papers. In [59], a practical way to design CMAC based classifiers was described. Because of the limitation of the input dimension, images needed to be preprocessed before being input into the CMAC. Different image preprocessing methods were used to extract the features of the images. Of these methods, converting a high-resolution image to a low-resolution one, also known as image down-sampling, is an effective and computationally efficient one. In an image, the neighbouring pixels have high correlation. That means some information of the image is redundant. This makes it possible to keep most of the features in a low-resolution version. Gaussian pyramid decomposition (GPD) [51] is often used to down-sample images. In [57], edge detector and GPD are used to extract the features. In [59], a Karhunen-Loeve (K-L) transformation [47] worked as a feature extractor. A feature vector was generated as the input to the CMAC. In [61], images whose size was within the CMAC input limitation were input to the network. In this paper, the experimental results exhibited an error rate of 8.5%.

2.5 Summary

More and more adaptive machine vision techniques are being developed to improve the inspection results in industry and manufacturing. The idea of adjusting parameters adaptively with respect to the image properties enters a lot of areas in computer and machine vision, including image acquisition, image processing and image classification. The development of improved computer hardware and software algorithms makes it possible to design fast, robust, reliable and low cost machine vision systems.

Chapter 3

Adaptive Machine Vision System for Surface Inspection: Image Acquisition

3.1 Introduction

This chapter and the following two chapters describe the design of an adaptive machine vision system for the surface inspection of machined cast parts. The chosen part is an automotive water pump housing manufactured by Orlick Industries. This chapter focuses on the image acquisition procedure for the surface inspection system. The algorithms to process the acquired images will be discussed in Chapters 4 and 5.

This chapter begins with an overview of the automotive water pump housing part surface inspection problem. This is followed by the design of the controlled lighting work cell, the selection of the digital camera, and an explanation of the camera calibration algorithm. Next, an adaptive camera control algorithm is presented. Finally, the chapter closes with some experimental results and conclusions.

3.2 Problem Description

Surface inspection is very important in automotive manufacturing. Even a very small defect on an automotive surface may cause a car to fail or a customer to be unhappy. Therefore, for each car part provider the surface inspection is critical. Traditionally, automotive parts are inspected manually. However, it is very difficult for human beings to inspect parts consistently. Many factors may exert influence on the manual inspection, for example: fatigue, mood, and illness. With a computer vision system the consistency of the inspection process is greatly improved. Furthermore, thanks to the availability of inexpensive and powerful computers, a computer vision system can inspect an automotive part in several seconds, similar to the speed of a human being.

In this thesis we specifically study the surface inspection problem for the machined surfaces of cast parts. Figure 3.1 illustrates the particular automotive water

porosity on Surface #1. Finally, in Figure 3.5, we can observe pores and a subtle dent on the surface.

As can be seen from the figures, the defects are irregularly shaped. So we cannot match their shapes with a database and can only detect the defects by the changes brightness of the pixels of the part images. However, besides the defects, there are some other areas on Surface #1 where the brightness changes obviously. These areas are caused by machining marks, which are also apparent in Figures 3.2 to 3.5. These obvious machining marks exist in most of the parts. Another factor that may affect the detection of the defects is the view of the part. While a person can check the part from different views of the surface, our system will be limited to acquiring the image of the surface with a fixed view. All of these factors make the discrimination of the defects from the non-defective regions a challenging problem.

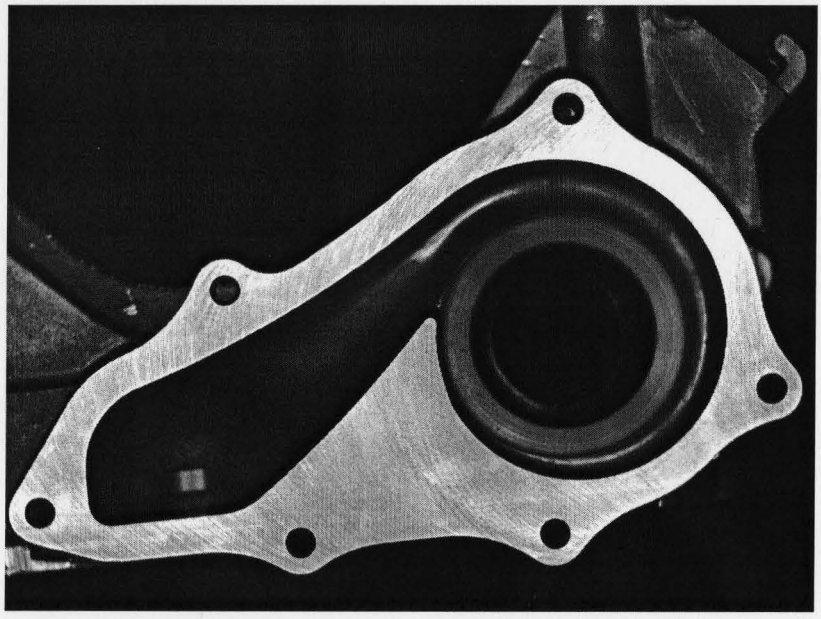


Figure 3.2 An example of a good part.

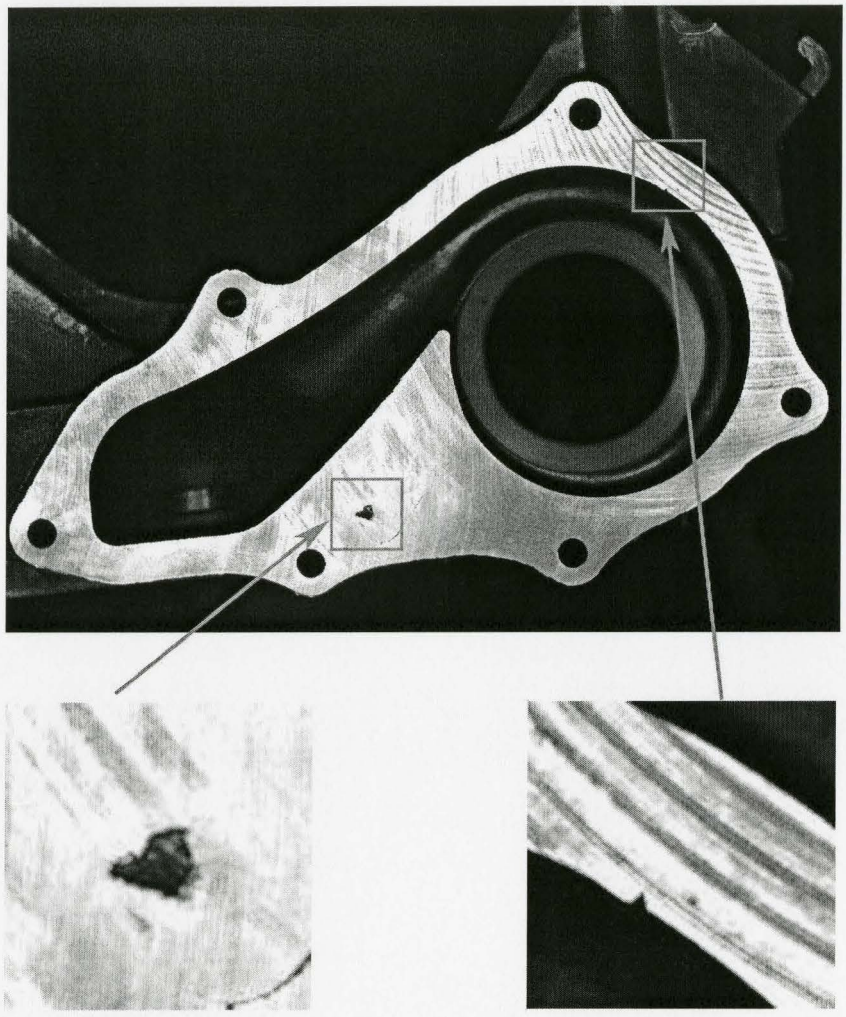


Figure 3.3 A defective part with an obvious defect and machining marks.

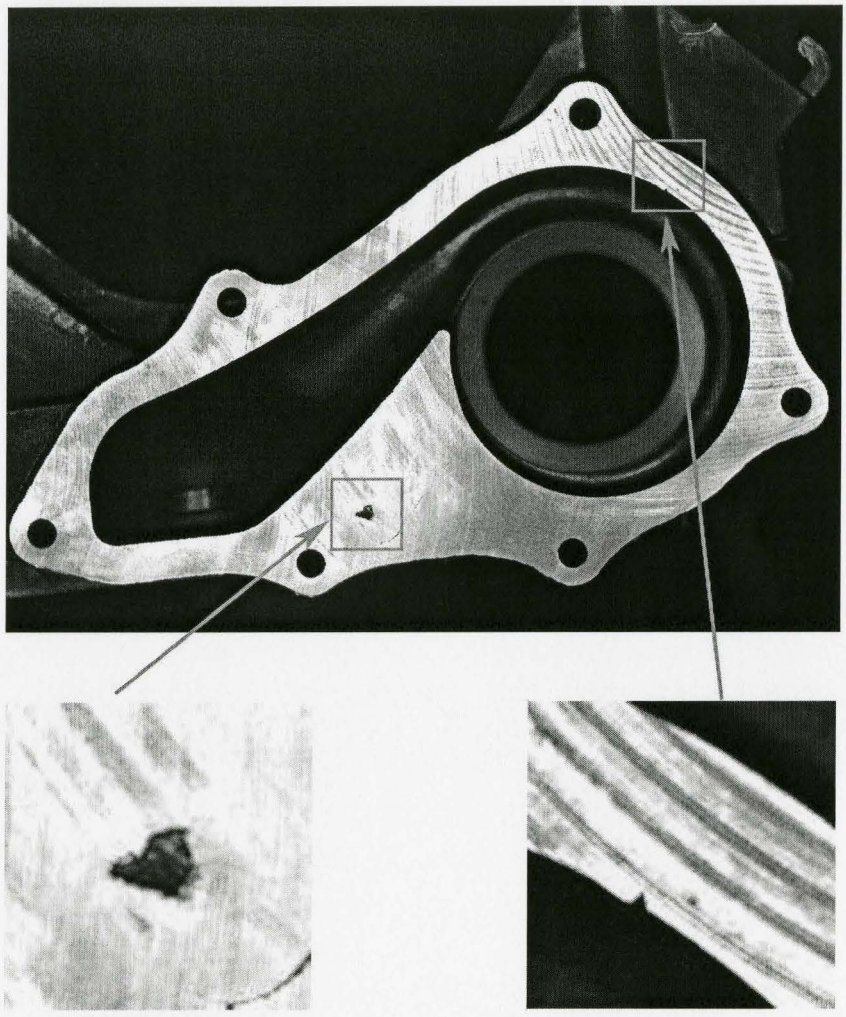


Figure 3.3 A defective part with an obvious defect and machining marks.

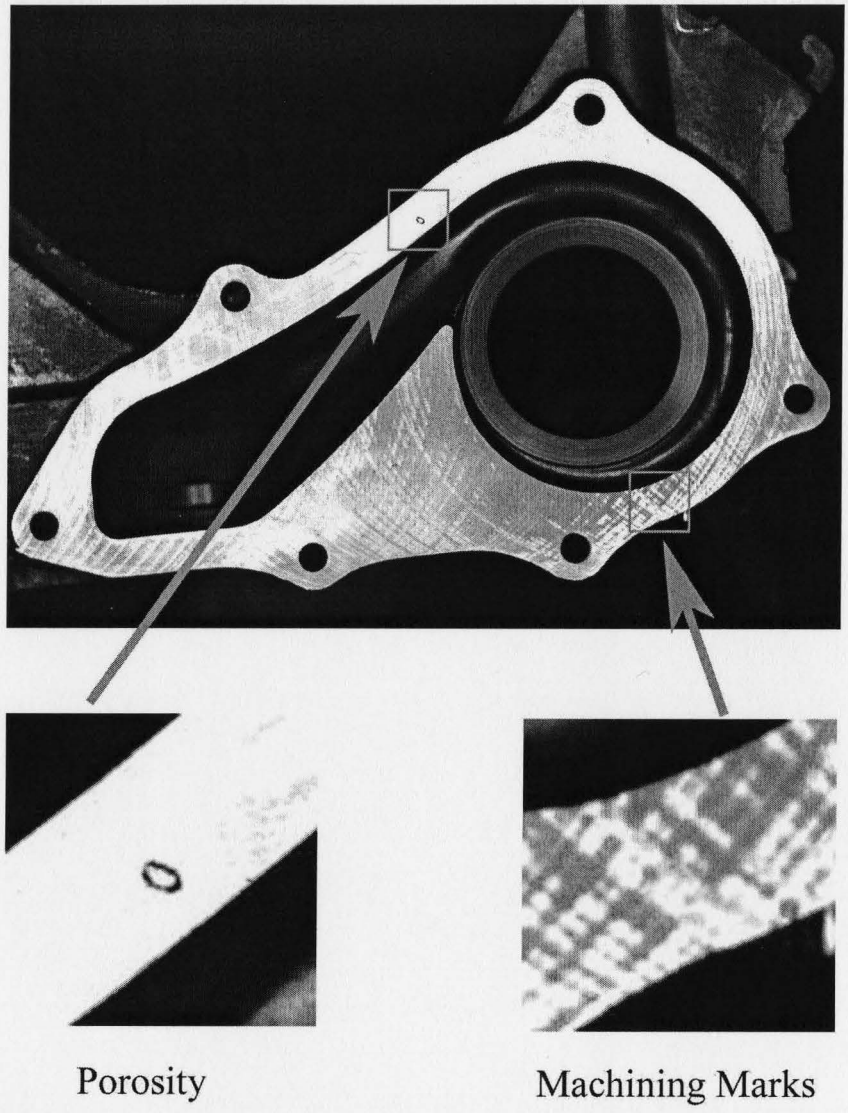


Figure 3.4 A defective part with porosity and machining marks.

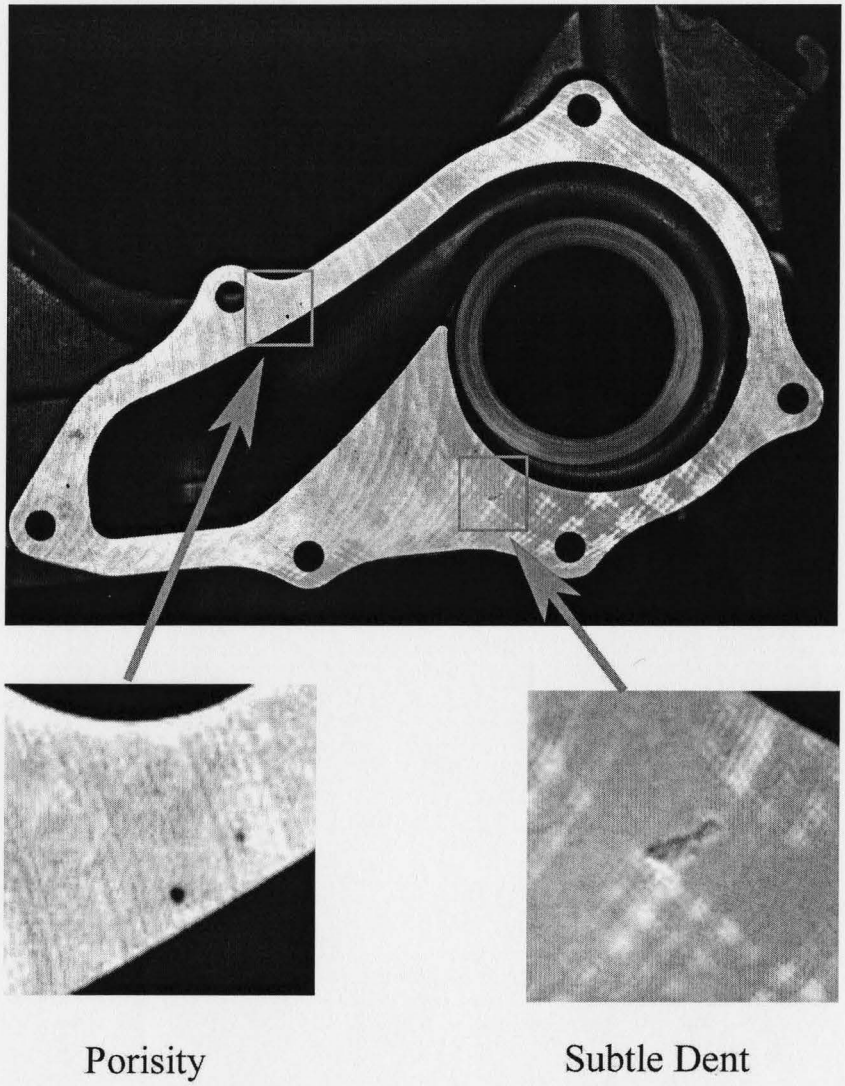


Figure 3.5 A defective part with pores and a subtle dent.

3.3 Digital Camera Selection

For a computer vision system a key problem is to obtain high quality images. Better images make the processing steps easier and the final results more reliable. First of all, we need to choose a camera as the imaging device. Since the automotive water pump housing parts are made of aluminium, colour images are not better than gray scale ones in the image processing procedure. Therefore, a monochrome camera is well suited for this surface inspection problem. We choose a PGR Scorpion SCOR-14SOM IEEE-1394 monochrome digital camera as the imaging device, which is shown in Figure 3.6.

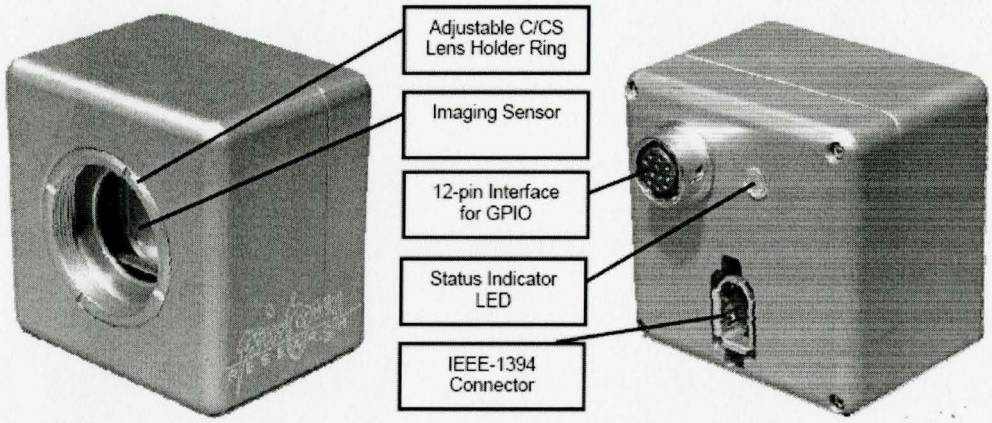


Figure 3.6 Picture of the selected PGR Scorpion SCOR-14SOM monochrome camera. [63]

The Scorpion SCOR-14SOM digital camera uses a monochrome, SONY 1/2" CCD sensor based imaging device, and the IEEE-1394 bus for rapid data transfer. The CCD sensor is a SONY ICX267AL. It is a progressive scan image sensor with a square pixel array and 1.45 million effective pixels. The ICX267AL has the following desirable features: [64]

- Images with 1280×960 pixel^2 can be read at a frame rate of 30 Hz;
- Progressive scan allows individual readout of the image signals from all pixels;

- High sensitivity, and low dark current;
- Low smear, and excellent anti-blooming characteristics; and
- Continuously variable speed shutter.

The size of the Surface #1 is approximately 180mm (horizontal) × 110mm (vertical). If we use the Scorpion SCOR-14SOM IEEE-1394 mono digital camera to take images with the resolution of 1280 (horizontal) × 960 (vertical), we can estimate the corresponding resolution on Surface #1 as:

Horizontal: $180\text{mm} / 1280 = 0.14\text{mm}$

Vertical: $110\text{mm} / 960 = 0.11\text{mm}$

One of the objectives of our computer vision system is that it should have the ability to detect defects with a diameter of 0.5mm. Since $0.14 < 0.5$, the Scorpion SCOR-14SOM digital camera has sufficient resolution to be applied as the imaging device for this surface inspection problem. The resolution of the camera will be analyzed in section 3.5.

3.4 Controlled Lighting Work Cell

The goal is to inspect the machined Surface #1 of an automotive water pump housing part. Compared with a coarse surface, a machined surface reflects more light into the camera. For a gray scale image, more light reflection produces a brighter area. It was expected that the machined surface would be brighter than other areas in the acquired image. Unfortunately, under the ambient lighting condition, it was very difficult to distinguish the machined surface to be inspected from the irrelevant areas in the acquired image. As an example, Figure 3.7 shows an image taken under the ambient lighting condition. To enhance the contrast of the image, we built a work cell with controlled lighting. The work cell is shown in Figures 3.8 and 3.9.

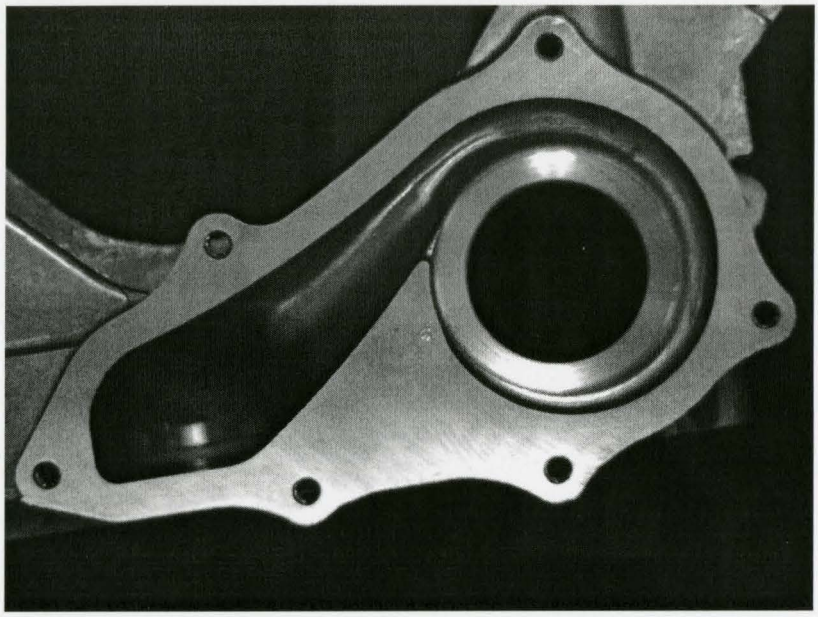


Figure 3.7 Image taken under ambient lighting condition.

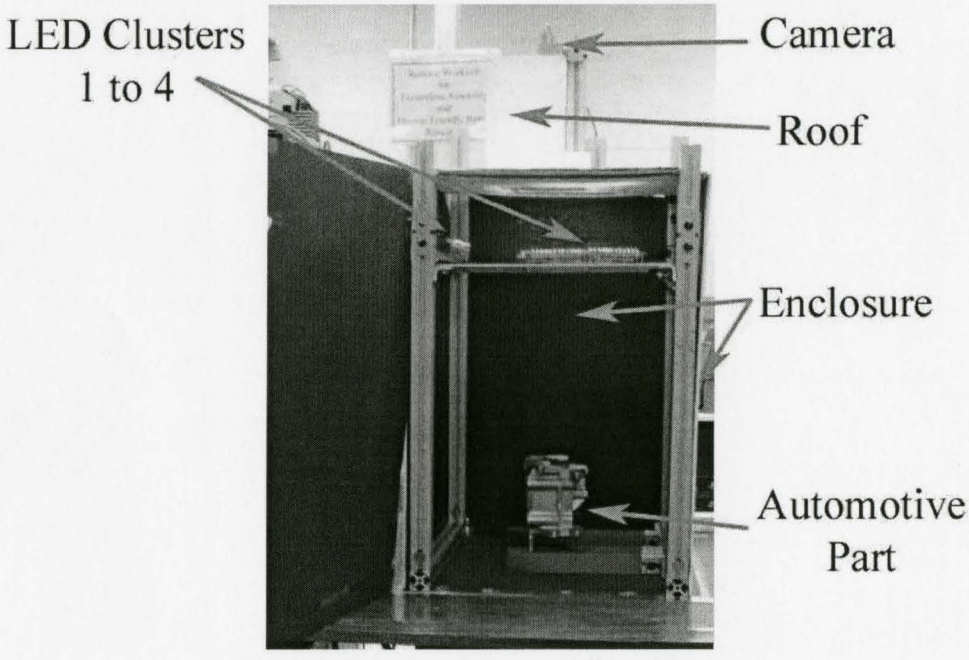


Figure 3.8 Controlled lighting work cell.

As indicated in Figure 3.8, the work cell is covered by foam board "walls" and a special designed roof. The inner side of the wall is black, which reduces light reflection

onto the machined surface to avoid camera saturation. The roof is designed to prevent light out of the work cell from entering the camera and to work as a reflecting area in the work cell. The light source of the work cell is composed of four LED clusters. Each cluster consists of 24 SSL-LX5093SRC/E red LEDs. The LED clusters can be slid along the tracks to change their positions. Also, they can be turned to adjust the light reflecting angles.

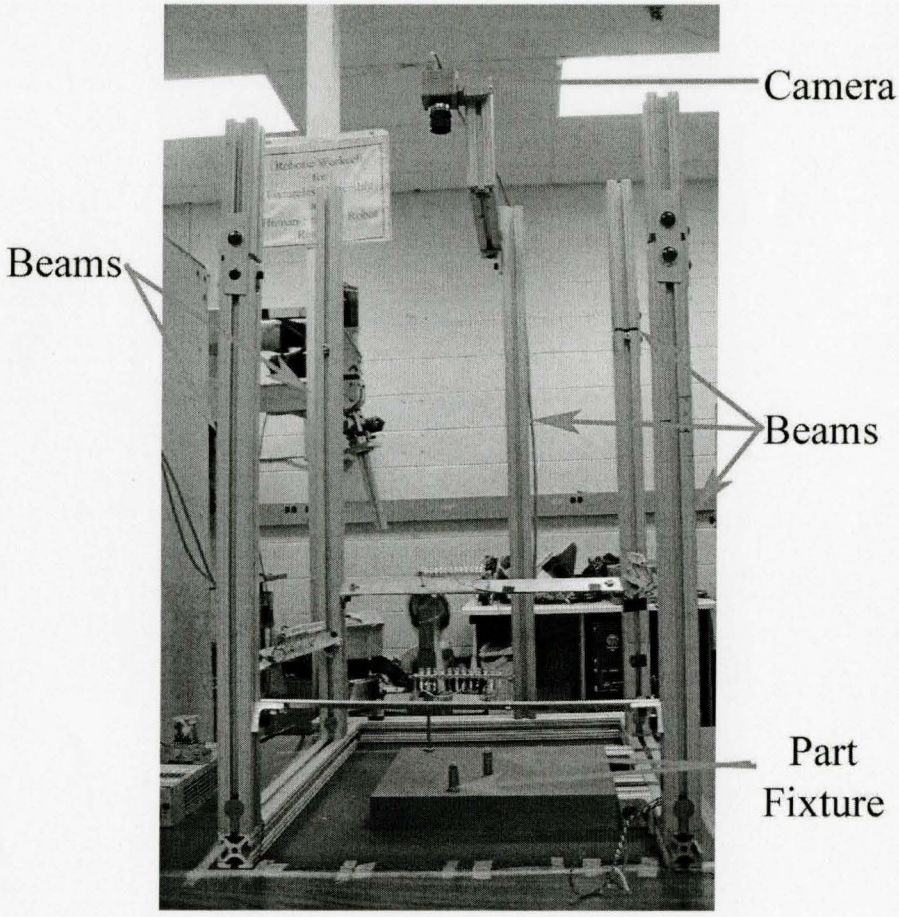


Figure 3.9 Controlled lighting work cell frame.

As illustrated in Figure 3.9, there are five beams framing the work cell: one is used to fix the camera; and the other four are used to support the walls and as tracks for the LED clusters. At the bottom of the work cell there is a part fixture to guarantee the parts stay at a fixed location when taking pictures. There are two main reasons to install

this part fixture. One is that the machined surface should be as large as possible in the acquired image in order to get better results. This part fixture can guarantee that the entire surface appears in the acquired images. Another reason is that the measured size of the detected defects is related to the distance between the imaging plane and the camera. Since it is not practical to calibrate the camera for each part, this part fixture is installed to keep the machined surfaces of the different parts in the same plane when taking pictures.

We chose the PENTAX 25mm 1:1.4 C-mount lens for the digital camera. The parameters of the lens are listed in Table 3.1 [65]. Based on the parameters of the lens and the digital camera, we can estimate the distance between the lens and the Surface #1 when setting up the work cell. Assume the lens is a thin lens, we use the following thin lens formula [62]:

$$\frac{1}{f} = \frac{1}{d_{object}} + \frac{1}{v} \tag{3.1}$$

where f is the focal length of the lens, d_{object} is the distance from the lens to the object, and v is the distance from the image to the lens.

From Table 3.1, the minimum object distance is 0.3m, which is larger than ten times of the focal length of the lens: 25mm. By applying (3.1), the maximum image distance will be 27.3mm. So we can simply consider the image distance the same as the focal length of the lens. The diagram of the imaging geometry is shown in Figure 3.10.

Table 3.1 Lens parameters.

Format Size		1
Focal Length		25mm
Max. Aperture Ratio		1: 1.4
Iris Range		F1.4 22
Horizontal Angle of View (Degrees)	1/4	8.23
	1/3	10.97
	1/2	14.62
Min. Object Distance		0.3m
Back Focal Length		14.98mm
Filter Size		27mm, P=0.5mm
Mount (Flange Back)		C (17.526mm)
Weight		76 *g*
Remarks		Lock Screw Extra

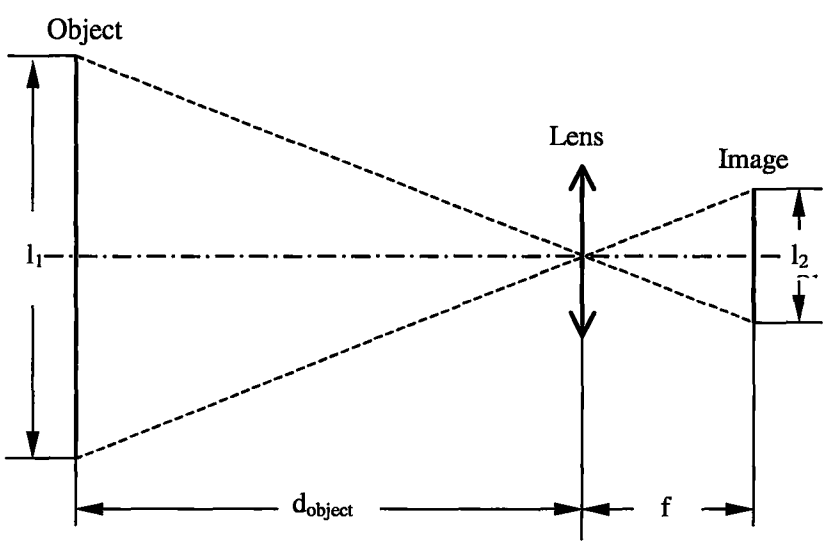


Figure 3.10 Diagram of the imaging geometry.

From Figure 3.10, we can obtain the equation to estimate the distance between the object and the lens as

$$d_{object} = \frac{l_1}{l_2} \times f \tag{3.2}$$

where d_{object} is the distance between the object and the lens; f is the focal length of the lens; and l_1 and l_2 are the sizes of the object and the image respectively. In this surface inspection problem,

$$\begin{cases} f = 25\text{mm} \\ l_1 = 175\text{mm} \\ l_2 = 1280 \times 4.65\mu\text{m} = 5.925\text{mm} \end{cases} \tag{3.3}$$

Based on these parameters, d_{object} can be estimated as 738mm. In practice, we manually adjusted this distance. After careful adjustment, the distance between Surface #1 and the lens that gave the desired field of view was set as 760mm, close to the estimated value.

When taking pictures, the light comes from the LED clusters and is reflected by the ceiling of the work cell. By adjusting the location and the orientation of the LED clusters, we could obtain different images. After carefully adjusting the configuration of the work cell, high quality images were obtained. The same work cell configuration was used to obtain all of the images of Surface #1.

3.5 Camera Calibration

Camera calibration must be done after setting up the work cell because of the following two reasons. First, we need to know the resolution of the camera. The camera resolution determines the minimum displacement that can be measured by the camera. The second reason is the conversion of measurement units. The object size is measured in mm in industrial applications while the size is measured in pixels in image processing. By calibrating the camera we can acquire the mapping that describes the relationship between the real world coordinates and the image coordinates.

We apply a standard least mean square (LMS) based optimization method to calibrate the camera. The main idea of the algorithm is to obtain the transformation matrices (also termed extrinsic matrices) of the camera calibration through the coordinates of the feature points measured in both the real world and the image space.

A standard chessboard is used as the reference object in the calibration, which is illustrated in Figure 3.11. Before calibrating the camera, we measure the size of the black/white blocks in the chessboard. The unit of the block size is mm. Then we assign an origin of the chessboard and set up a coordinate system in mm for the chessboard. With this coordinate system, we can obtain the coordinates of the corners in the chessboard in the real world coordinate system.

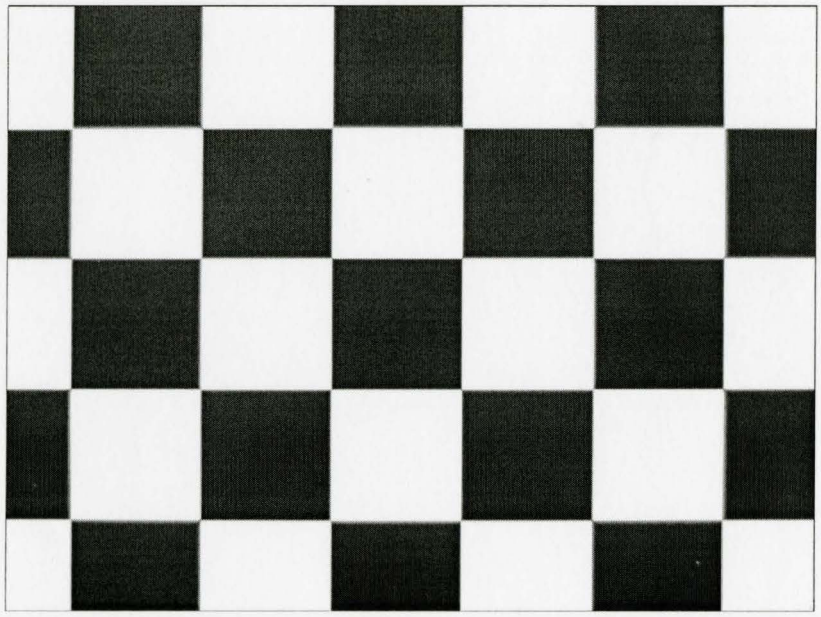


Figure 3.11 Image of the chessboard used for the camera calibration.

Now we need to find out the coordinates of the corners in the image plane. By applying the function of cvFindChessboardCorners() in the OpenCV library [66], we can detect the corners of the chessboard in the image. The result is shown as Figure 3.12. With aid of function cvFindCornerSubPix(), we can find out the coordinates of the corners measured in pixels. The accuracy of this function is sub-pixel. With the obtained

coordinates of the chessboard corners both in real world and in image coordinate systems, we can calculate the transformation matrix (extrinsic matrix) between the two coordinate systems.

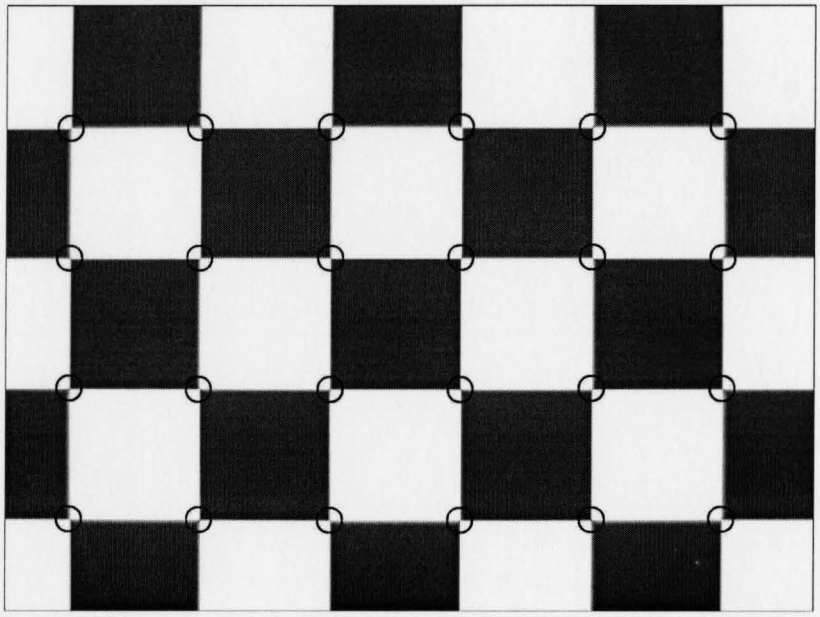


Figure 3.12 Result produced by the function cvFindChessboardCorners().

Assuming we detect n chessboard corners by using OpenCV function cvFindChessboardCorners(), we define (x_i, y_i) and (x_i', y_i') as the coordinates of the chessboard corners in the real world coordinate system and in the image coordinate system, respectively. Same as in [66], we define the camera extrinsic matrix as

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.4}$$

In the LMS based algorithm, the optimal extrinsic matrix **H** is the one which minimize the back-projection error. The back-projection error is defined as

$$\sum_i \left[\left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \right] \tag{3.5}$$

Appying function cvFindHomography() of the OpenCV library, we can obtain the extrinsic matrix $\mathbf{H}$. With our work cell, the extrinsic matrices of the camera are:

$$\mathbf{H}_1 = \begin{bmatrix} 7.23 & 0.01 & 101.99 \\ -0.01 & 7.26 & 194.33 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{3.6}$$

and

$$\mathbf{H}_2 = \begin{bmatrix} 0.14 & -0.0 & -14.08 \\ 0.0 & 0.14 & -26.77 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{3.7}$$

where $\mathbf{H}_1$ is transformation matrix from the real world coordinate system to the image coordinate system, and $\mathbf{H}_2$ is the transformation matrix from the image coordinate system to the real world coordinate system.

After obtaining the extrinsic matrix of the camera, we can test its real world resolution. We need to calculate the minimum displacement that can be measured by the camera. The minimum displacement is the minimum of the distance between the neighbour pixels in the same row and in the same column. The distance between the neighbour pixels in the same row is:

$$d_{rowmin} = \left\| \mathbf{H}_2 \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\| = 0.13 \text{ mm} \tag{3.8}$$

The distance between the neighbour pixels in the same column is:

$$d_{colmin} = \left\| \mathbf{H}_2 \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\| = 0.14 \text{ mm} \tag{3.9}$$

So the resolution of the camera is 0.14mm. Since the minimum defective size of this project is 0.5mm, this camera has sufficient resolution for our application.

The calibration errors in mm were also calculated. First we numbered the 24 detected chessboard corners from left to right and from top to bottom. Next, the transformation matrix $\mathbf{H}_2$ in (3.6) was used to convert the coordinates of these corners in

the image space to the coordinates in the real world. The differences between these calculated coordinates and the corresponding real coordinates were calculated as the calibration errors. These errors are listed in Table 3.2. It should be noted that the local errors, for example when determining the size of a defect, will be much smaller than the maximum value listed in this table.

Table 3.2 Typical calibration errors

Corner No.	Real Coordinates (mm) (x_0, y_0)	Calculated Coordinates (mm) (x_1, y_1)	Error (mm) $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$
1	(0.0, 0.0)	(0.06, 0.08)	0.1
2	(28.5, 0.0)	(28.48, 0.04)	0.05
3	(57.0, 0.0)	(57.05, -0.02)	0.05
4	(85.5, 0.0)	(85.57, -0.02)	0.07
5	(114.0, 0.0)	(114.15, 0.02)	0.15
6	(142.5, 0.0)	(142.56, -0.03)	0.07
7	(0.0, 28.5)	(0.02, 28.50)	0.02
8	(28.5, 28.5)	(28.38, 28.44)	0.13
9	(57.0, 28.5)	(57.07, 28.46)	0.08
10	(85.5, 28.5)	(85.51, 28.48)	0.03
11	(114.0, 28.5)	(114.23, 28.46)	0.23
12	(142.5, 28.5)	(142.58, 28.50)	0.08
13	(0.0, 57.0)	(-0.04, 56.92)	0.09
14	(28.5, 57.0)	(28.34, 57.05)	0.17
15	(57.0, 57.0)	(57.01, 57.08)	0.09
16	(85.5, 57.0)	(85.50, 57.11)	0.11
17	(114.0, 57.0)	(114.23, 57.13)	0.26
18	(142.5, 57.0)	(142.559, 57.019)	0.06
19	(0.0, 85.5)	(0.09, 85.31)	0.21
20	(28.5, 85.5)	(28.45, 85.43)	0.09
21	(57.0, 85.5)	(57.02, 85.56)	0.06
22	(85.5, 85.5)	(85.51, 85.58)	0.08
23	(114.0, 85.5)	(114.14, 85.53)	0.14
24	(142.5, 85.5)	(142.45, 85.49)	0.05

3.6 Adaptive Camera Control

It is very difficult to obtain high quality images for different parts if fixed camera parameters are used during the image acquisition. In general, there are two main methods to improve the image quality for different parts: one is to adjust the parameters of the camera for each part; and the second way is to change the lighting condition of the work cell adaptively. Since the controlled lighting work cell we built cannot be adjusted automatically at present, adjusting the camera parameters via software is a good choice to acquire images of high quality.

In this section, an adaptive camera control algorithm is discussed. The main idea of this algorithm is to adjust the camera parameters according to the acquired images. Compared with the images under the fixed camera parameters, images taken by using the adaptive algorithm should have better contrast and be easier to analyze.

3.6.1 Algorithm Overview

For a given digital camera, the exposure time is an important parameter. The exposure time is defined as the interval when the shutter is open to allow a certain amount of light (also related with the opening of the lens diaphragm) to pass through and expose the CCD sensor. Different values of exposure time will yield different kinds of visual effects in an image. Generally, longer exposure times will allow more light to reach the sensor and make the images brighter. On the other hand, longer exposure times will blur the image while shorter exposure time can freeze the action and enhance the contrast of the image. Therefore, by adjusting the exposure time, we can adjust the brightness, saturation, and contrast of the image.

At present, we are only interested in Surface #1, which is defined as the region of interest (ROI). Our objective is to obtain high quality for the ROI images. Therefore, we can evaluate the quality of the ROI for each image and adjust the exposure time adaptively. However, the evaluation of the image quality of ROI is influenced by the pixels outside the ROI. To solve this problem, we first segment the ROI from the whole image using a ROI mask. The introduction of this mask provides the benefits of lower

computational cost and eliminates the pixels outside the ROI from the camera control algorithm. The method for generating this mask will be discussed in Chapter 4.

In order to apply the ROI mask to different images, we need to keep the different parts in the same location of the images. Although we designed the part fixture to try to keep the location of the parts unchanged, there still exists slight motion among the images of the different parts and even the different images of the same part. As a result, there exists a slight deviation between the generated mask and every automotive part image. To compensate for this deviation, the mask has to be registered before being used. The image registration algorithms will be discussed in the next chapter.

After registration of the ROI mask, we segment the ROI from each image and calculate the statistical information of the ROI. The exposure time can be adjusted adaptively according to the acquired statistical information to achieve the desired quality. The diagram of this adaptive camera control algorithm is shown in Figure 3.13.

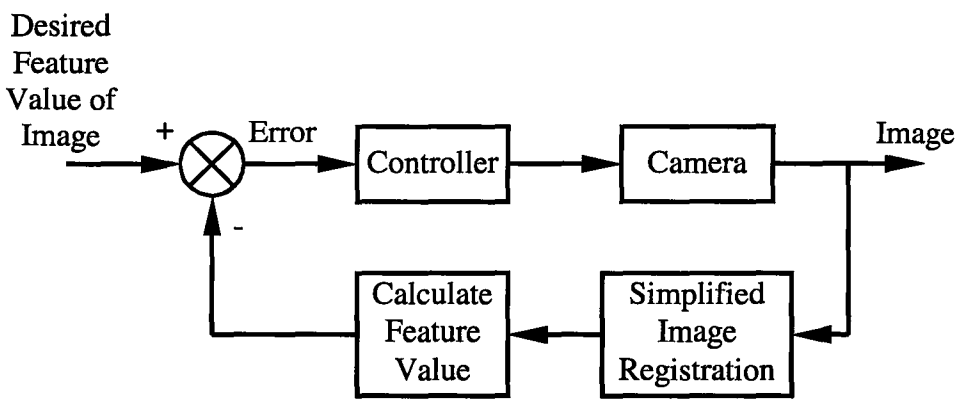


Figure 3.13 Diagram of the adaptive camera control.

3.6.2 Assumptions

We assume the following:

1. Surface #1 of the inspected part is well machined;
2. Holes used for the mask registration are drilled;
3. The initial values of the camera parameters are suitable for finding the features easily and accurately enough;
4. Surface #1 is approximately perpendicular to the axis of lens;
5. The image fully covers Surface #1;
6. The mask can be matched to the part without altering its scale;
7. The coordinates of the hole centres in the mask image are known;
8. The desired feature value of the image is given.

The first two assumptions are satisfied by most of the automotive water pump housing parts. Initializing the camera with proper exposure time satisfies the third assumption. This makes the image brightness and contrast sufficient.

The simplified registration algorithm (described in Chapter 4, Section 4.3.8) satisfies the sixth assumption. Under this assumption, we can model the motion between the mask image and the part image by translation and rotation only. Therefore, only two hole centres are needed for the image registration.

The remaining four assumptions were satisfied by initial manual setting up of the work cell. Satisfying assumption four and seven was straightforward. We adjusted the locations of the part fixtures and the focal length of the digital camera to make the part image contain all of Surface #1 (assumption five). The last assumption involves the inputs to the algorithm. To acquire this feature value, we first adjust the camera manually and take images under different exposure times. The manual adjustment procedure continues until a high quality image is acquired. Finally, we calculate the feature value of the image and use it as the desired value of the adaptive camera control algorithm. Note that this task is performed only once.

3.6.3 Algorithm

Algorithm 3.1 Adaptive Camera Control

An image is taken by the following procedure:

Step 1: Initialize the camera with the parameters needed to take a bright image;

Step 2: Acquire an image;

Step 3: If this is the first image, use the simplified registration algorithm to match the mask image up with the acquired image. Else, continue.

Step 4: Calculate the feature value of the acquired image;

Step 5: Compare the feature value of the acquired image with the desired feature value;

Step 6: If the magnitude of the error is larger than a predefined threshold, calculate the new value camera parameter based on the error. Change the camera parameter and go to step 2. Else, output the image and stop.

First, the camera should be initialized with the parameters to acquire bright images. We chose the exposure time as the adjustable camera parameter. The initial value of the exposure time needed to obtain bright images of the parts must be manually determined once. The bright image should satisfy two conditions: the area of Surface #1 is bright enough to be identified easily and the contrasts of the areas close to the six holes are strong enough to be segmented. This makes it possible to locate the holes and obtain the centres of the holes more accurately and more reliably for use in the mask registration.

After the first image is acquired, we apply the simplified image registration algorithm (described in detail in Chapter 4) to match the mask image up with the current image approximately. To reduce the computing cost, we move the mask image only once. Since the part does not move significantly during the whole image acquisition procedure, it is unnecessary to re-register the mask image for every shot.

The feature value is calculated for the registered image. In this thesis, we use the average intensity of the pixels in the mask area as the feature value. The average intensity is calculated as:

$$\bar{p} = \frac{1}{N_{mask}} \sum_{k=1}^{N_{mask}} p_k \tag{3.10}$$

where N_{mask} is the total number of pixels with binary value of 1 in the mask image, and p_k is the gray scale value of a masked pixel in part image. The average intensity is compared with the desired value. If the error is smaller than a pre-set threshold, the exposure time is considered acceptable and the taken image is used to inspect the part. Otherwise, we adjust the exposure time with respected to the error and re-take the image for checking.

The control strategy to adjust the camera exposure time is simple. If the error is positive then the exposure time is increased by a fixed increment. Similarly, if the error is negative the exposure time is decreased by the same amount. This adjustment procedure continues until the error magnitude is smaller than the threshold. It should be noted that this iterative approach can be slow. It can take several seconds to acquire an acceptable image of the part.

3.7 Experimental Results

We used the controlled lighting work cell and the adaptive camera control algorithm to acquire more than 1,700 images of different automotive water pump housing parts. A few examples that demonstrate the camera control algorithm will be discussed in this section. A comparison between Figure 3.14 and Figure 3.15 shows that the brightness distribution is different for different parts under fixed camera parameters. Comparing Figure 3.14 with Figure 3.16, and Figure 3.15 with Figure 3.17, we can observe that by using the adaptive camera control algorithm, the contrast of the image is enhanced, which is very important for the further image processing.

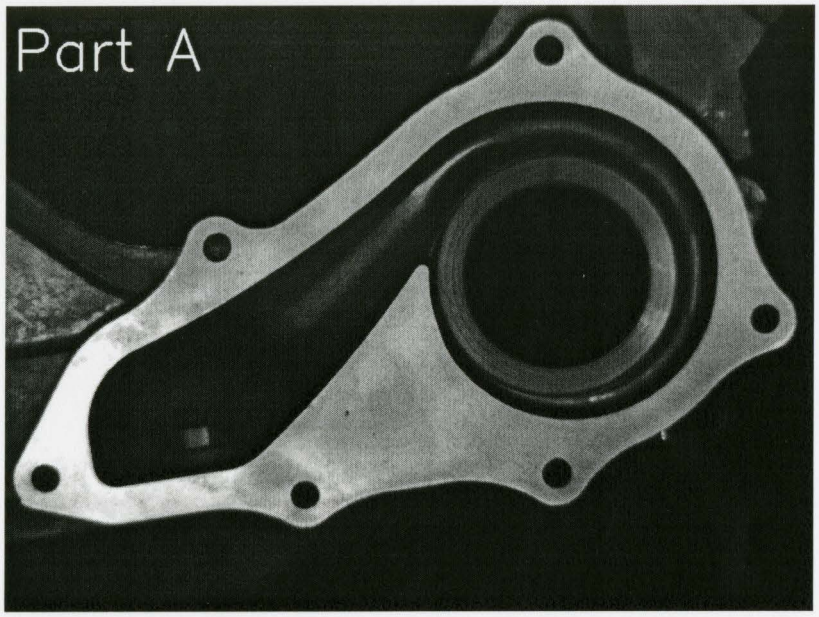


Figure 3.14 Image of Part A taken using fixed camera parameters.

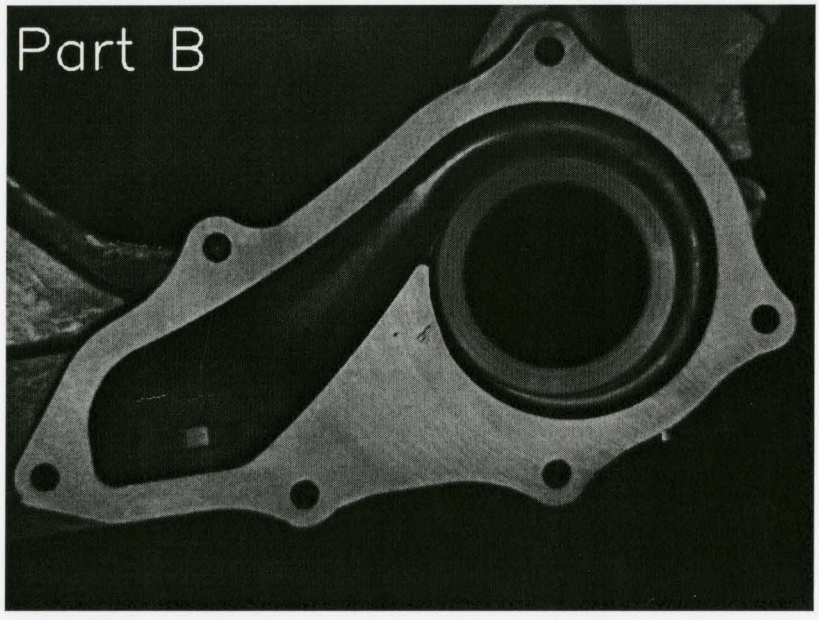


Figure 3.15 Image of Part B taken using fixed camera parameters.

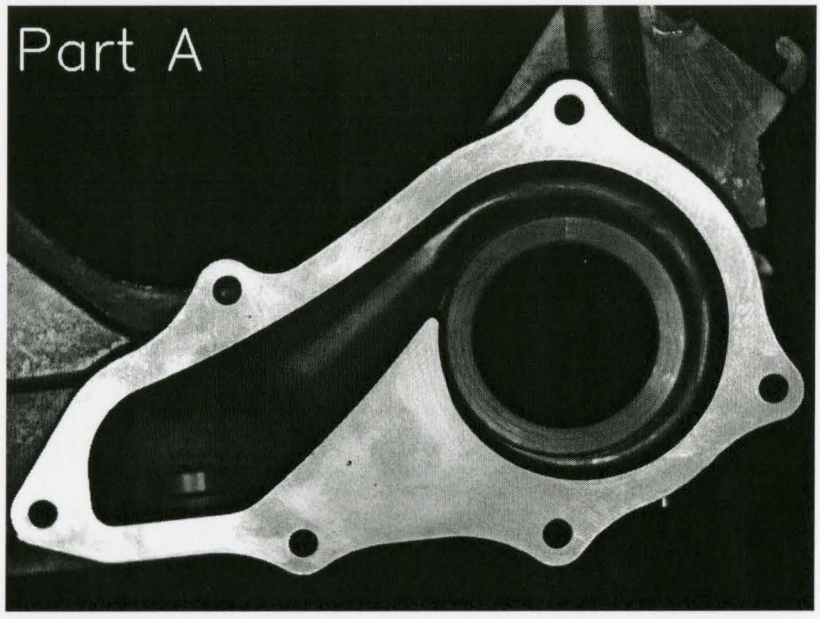


Figure 3.16 Image of Part A taken using adaptive camera control.

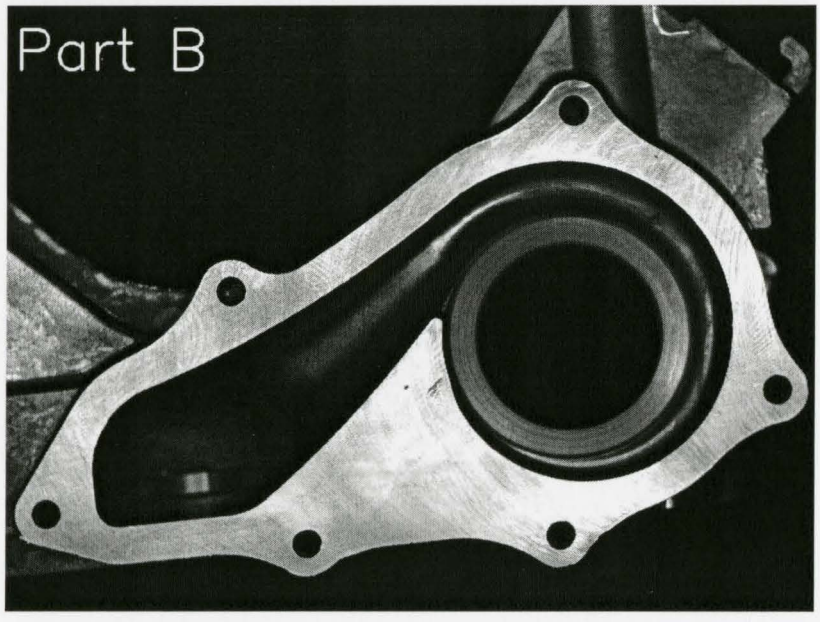


Figure 3.17 Image of Part B taken using adaptive camera control.

3.8 Summary

In this chapter, we presented the image acquisition procedure for the surface inspection problem. A controlled lighting work cell was designed to acquire part images. A camera calibration algorithm was used to obtain the transformation matrices between the image coordinate system and the real world coordinate system. In order to acquire high quality images, we developed a camera control algorithm to adaptively adjust the brightness and contrast of the acquired images. Experimental results showed that high quality images were acquired with the custom designed control lighting work cell and the proposed adaptive camera control algorithm.

Chapter 4

Adaptive Machine Vision System for Surface Inspection: Region of Interest Masking

4.1 Introduction

By employing the work cell and the adaptive camera control algorithm described in the last chapter, high quality images are available for further processing. As introduced briefly in the last chapter, we apply an image masking technique to segment the ROI before detecting and measuring the defects on Surface #1. This masking technique will be described in detail in this chapter.

This chapter is organized as follows. First, a seeded region growing based algorithm is developed to generate the ROI mask from one acquired image. Second, an accurate image registration algorithm is introduced to compensate the global motion between the ROI mask and the surface to be inspected. Third, a simplified image registration algorithm developed for matching the ROI mask up with the surface approximately during image acquisition is presented.

4.2 Generating the ROI Mask using Seeded Region Growing

Many machine vision algorithms process a ROI instead of the whole image. For the current inspection problem, Surface #1 is the ROI. Therefore, we must segment it from the whole image using an ROI mask. By applying this masking technique, we can avoid the possible negative affect of the background in both the image acquisition procedure and the inspection process. Furthermore, since the ROI area is always smaller than the entire image, we can reduce the computational cost greatly by only processing the pixels inside the ROI.

In this section, a seeded region growing based algorithm is proposed to generate the ROI mask. The proposed method is compared with the method available in the OpenCV library to evaluate its performance. Experimental results are shown in the end of this section.

4.2.1 Algorithm Overview

A mask image is defined as a binary image with pixel values of 0 or 1. The pixels with a value of 1 are inside the ROI. The remainder are outside of the ROI. Generally, there are two ways to generate an ROI mask: thresholding and region growing. The thresholding technique has the advantages of simplicity and small computational cost. However, it has two main drawbacks. First, the performance of the thresholding technique largely depends on the choice of the threshold. If there exists no prior knowledge of the ROI and the image, a good threshold is very difficult to obtain. Second, due to the noise existing in almost every natural image, the ROI mask generated by the thresholding technique cannot be guaranteed to be a connected area. The region growing technique overcomes the above two drawbacks. However, compared to the thresholding technique, it is much more complicated and the computational cost is higher.

For the automotive water pump housing part surface inspection problem, the parts to be inspected are standard parts and they all have similar shape and size. Also, the part fixture designed in the work cell can fix the location of the parts while taking images. Therefore, it's not necessary to generate the ROI mask for each acquired image. Instead,

we only generate one ROI mask and then match the mask up to every surface image. Since only one ROI mask needs to be generated, we care more about the quality of the mask than the computational cost. Based on this analysis, we developed the seeded region growing technique to generate the ROI mask.

4.2.2 Assumptions

We generate the mask under the following assumptions:

1. Surface #1 (ROI) is a connected area. That is, there exists at least one path between any two pixels inside the area;
2. There is a distinctive closed border between the ROI and other areas of the image;

Satisfying these assumptions will guarantee that the algorithm converges and generates a mask image covering the whole area of Surface #1.

The first assumption is true since the Surface #1 is an un-broken surface. The second assumption will be true when the lightning condition is well adjusted so that the contrast of the acquired image is sufficient.

4.2.3 Mask Generation by Seeded Region Growing Algorithm

Algorithm 4.1 Mask generation by seeded region growing

A mask image is generated by the following procedure:

Step 1: Initialize the data buffer used to store the categorized pixels of the part image and set all the pixels as UNCHECKED_PIXEL;

Step 2: Set the original seed as NEW_SEED and push it onto the stack;

Step 3: Pop the next seed from the stack, and check if it is 8-connected. If any of its neighbouring pixels is classified as UNCHECKED_PIXEL, categorize it according to the following rules. If the absolute value of the difference between the value of the UNCHECKED_PIXEL neighbour pixel and the value of the original seed is less than a threshold of 120, set the pixel as NEW_SEED and push it onto the stack; Otherwise, set the

checked pixel as CHECKED_PIXEL;

Step 4: Repeat Step 3 until no more pixels are pushed onto the stack;

Step 5: Check the data buffer storing the categorized pixels. If the pixel is set as NEW_SEED, set the value of the corresponding pixel (in the same location) in the mask image equal to 1. Otherwise, set the value of the corresponding pixel in the mask image equal to 0.

Step 6: Stop.

This algorithm is based on the work of Adams and Bischof [43]. At the very beginning, we allocate a data buffer to store the categorized pixels of the image. The size of the data buffer is the same as the image: 1280 elements by 960 elements. Every element in the data buffer is composed of two parts: one part is used to store the index of the responding pixel in the original image; another part is used to store its classification value. All the classification values in the data buffer are initialized as UNCHECKED_PIXEL.

The region growing procedure starts with the original seed we predefined manually. The original seed is classified as NEW_SEED and pushed onto a stack. The stack is used to store the seeds generated while the program is running. Then the seed is popped out of the stack and its 8-connected neighbour pixels are checked. For these neighbouring pixels, only those classified as UNCHECKED_PIXEL are processed to reduce the program running time and guarantee that the code converges. We compare the value of the checked pixel with the value of the original seed to classify whether the pixel is a new seed. Once the pixel is classified, the classification value will not be changed. If the neighbour pixel has been checked it is unnecessary to check it again. We only push the newly generated seeds onto the stack to avoid storing all the seeds in one operation cycle. Generally, the original seed can be chosen from some feature points extracted from the image. To make the program run faster, we manually chose a pixel in the central area of Surface #1 as the original seed.

According to the above growing procedure, it is expected that the number of the new seeds will increase rapidly. For example, after the first iteration, there may be eight new seeds being pushed onto the stack. The number of new seeds increases to 16 after the second iteration, 24 after the third iteration, and 32 after the fourth iteration, etc. Fortunately, the number of new seeds will decrease when the region growing procedure reaches the edge of Surface #1. The region growing procedure will end when no new seed has been pushed onto the stack.

After the region growing finishes, we generate the binary mask image corresponding to the seeds stored in the stack. For every stored seed, we set the value of the corresponding pixel in the mask image as 1. Note that all the pixels in the mask image must be initialized with a value of zero.

To evaluate the performance of the proposed seeded region-growing algorithm, we compared it with the method provided in the OpenCV library [66]. The masks generated by our proposed method and by calling the function cvFloodFill() are shown in Figure 4.1 and Figure 4.2 respectively. The reverse images are displayed for convenience. That is, the dark area is the ROI. From the figures, we can observe the invalid pixels existing in the mask generated by the OpenCV function. These pixels will introduce errors into the subsequent image processing. The mask generated by the proposed seeded region-growing method completely covers the Surface #1 without introducing any invalid pixels.

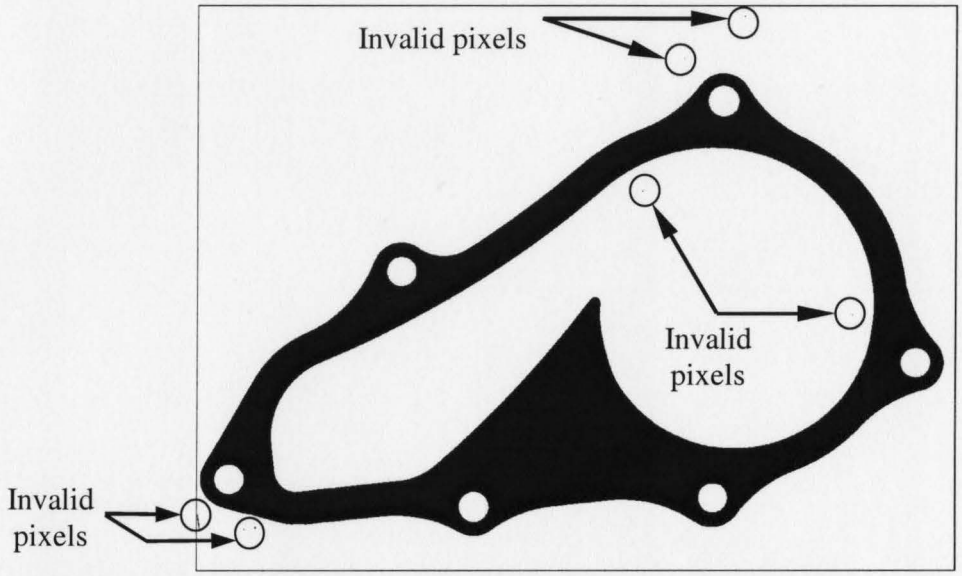


Figure 4.1 Mask image generated by using the OpenCV library.

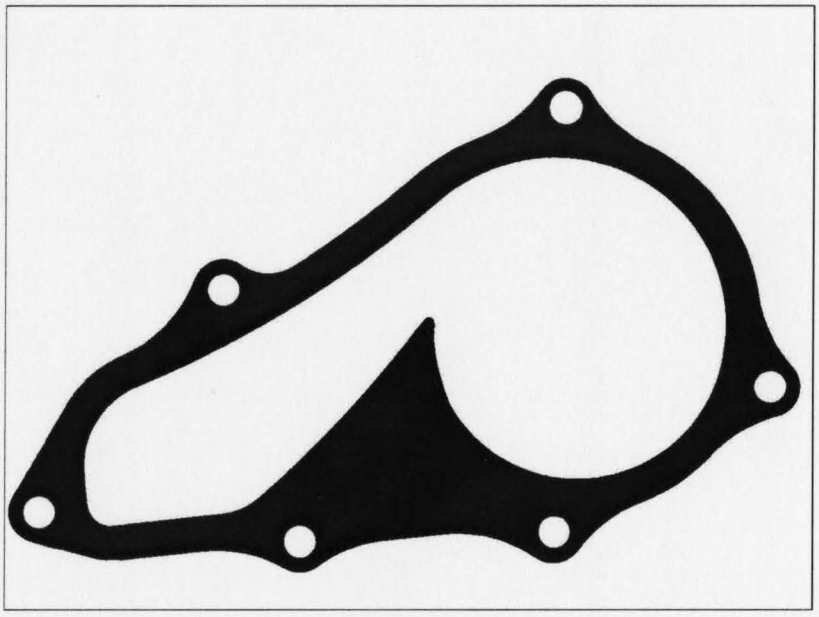


Figure 4.2 Mask image generated by using the proposed seeded region growing algorithm.

4.3 Image Registration

By applying Algorithm 4.1, we obtained a mask for use in the upcoming image processing procedures. When applying the mask to the images of the different automotive parts, there is an important problem to be solved: matching the mask up with these images. Consequently, an image registration algorithm is required. In order to detect the defects on Surface #1 and reduce the negative affects of the background as much as possible, it is required that the mask should match up with each image accurately. In this thesis, we will use two registration algorithms: an accurate algorithm, and a simplified algorithm.

This section is organized as the follows. First, we present an overview of both algorithms. Second, we describe the accurate image registration algorithm. The algorithm includes a pattern-matching algorithm to detect the coarse features, an adaptive seeded region growing algorithm to refine the feature points, and a least mean square (LMS) based method to find an accurate global motion model. Finally, we will present the simplified image registration algorithm used during the image acquisition.

4.3.1 Overview of Both Image Registration Algorithms

As discussed in the previous section, in our system an ROI mask is generated from one image and applied to every other image in both the image acquisition procedure and the inspection procedure. Although we apply the part fixture in the work cell to fix the automotive parts, there still exists slight global motion among the images of different parts and among different images of the same part. Therefore, we need to register the mask with the images to be processed.

Our image registration process includes the following four main steps:

- Feature detection: detecting and locating the features on the image to be processed;
- Feature matching: matching the detected features between the mask image and the image to be processed;
- Motion estimation: estimating the motion model parameters according to the detected and matched features; and
- Mask registration: transforming and resampling the mask image to register it with the image to be processed.

The accuracy of the estimated motion largely depends on the accuracy of the feature detection. In general, features are always chosen as some distinctive and easily detectable objects, such as edges, corners, line intersections, etc. By analyzing the automotive water pump housing parts we can observe that there are six cleanly drilled holes on Surface #1, which are labelled in Figure 4.3. In every image, these six holes appear as six circle circular regions with good contrast. We chose these six holes as the features for our image registration algorithms.

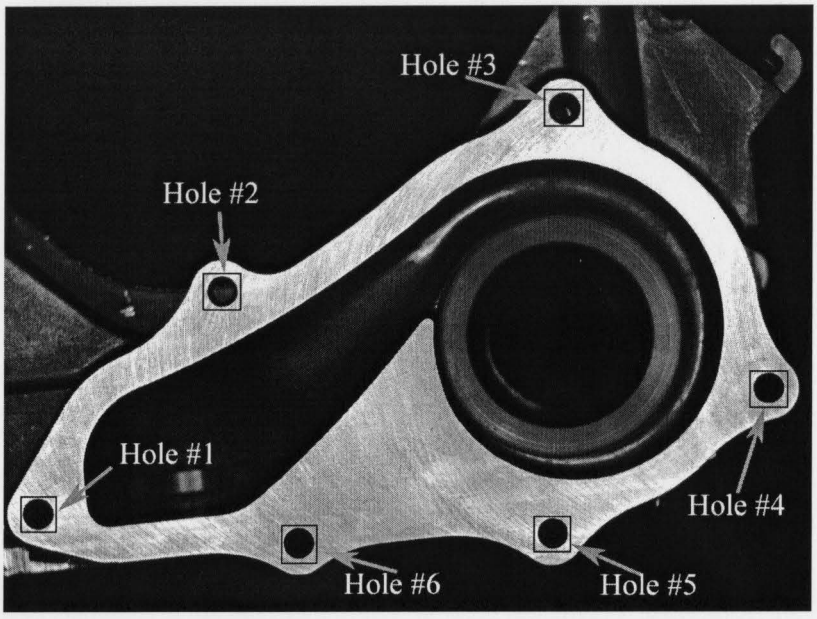


Figure 4.3 Hole #1 - Hole #6 are features used in image registration.

In the feature detection step, these six holes are first segmented from the whole image. An adaptive seeded region-growing algorithm is applied to the six sub-images to locate the centres of these holes. These six holes are then matched with the six holes in the mask image in the feature-matching step.

After locating the feature points, the motion estimation problem with the accurate registration algorithm converts to an optimization problem to find the optimal parameters of the motion model, which turns out to be a typical LMS based optimization problem. Several motion models can be applied in image registration, including translation model, affine model, perspective model, etc [69]. Since we use part fixtures to locate the parts, the affine model is satisfactory to describe the small global motion between the standard mask and the images.

With the estimated motion model parameters, the mask can be registered with the image to be processed by transforming and resampling. Since the estimated motion has sub-pixel precision, image interpolation may be involved in the resampling process.

The affine motion model can describe translation, rotation and scaling motion. However, in the image acquisition procedure, since we only require the statistical information of the ROI for the camera control algorithm, the mask does not have to match Surface #1 very accurately. Furthermore, a simplified image registration algorithm will help to speed up the image acquisition. As a result, we use a simpler algorithm that ignores the possible scaling motion between the standard mask and the images to be taken. Since now only translation and rotation motion is considered, we only use two instead of six feature points to estimate the motion. The centres of the two holes with the best contrast were chosen as these two feature points. By observing many acquired images, Hole #1 and Hole #4 were the two holes that satisfied this requirement in most cases, and were selected.

4.3.2 Assumptions

We assume that:

1. Surface #1 of the inspected part is well machined;
2. Holes used as the features in Surface #1 are drilled;
3. The image fully covers Surface #1;
4. The histograms of the sub-images of the holes are bimodal;
5. The sub-images of the holes are connective.

The first three assumptions are same as for the adaptive camera control algorithm discussed in Chapter 3. The fourth assumption holds when suitable lighting is used, and is true for the controlled lighting work cell. The last assumption holds when these two conditions hold:

1. The dimensions of the sub-image are larger than the maximum diameter of the six holes;
2. The holes are not at the edges of the sub-images.

These two conditions will be made true by applying the proposed feature detection algorithm.

4.3.3 Feature Detection

The objective of the feature detection part is to locate the six holes in the image of an automotive water pump housing part and segment these six sub-images from the whole image. The contours of these six holes appear as circles in the images. A popular technique to detect circles in images is using the Hough transform[67]. We conducted experiments to detect circles in the part images with the software implemented by Papamarkos [71]. This software detects edges and generates a binary image from the input image, and the circles within the given diameter range are detected in the binary image using Hough transform.

Figures 4.4 and 4.5 show two examples of the results using this software. The detected circles are illustrated in red. The maximum diameter of the circles to be detected was set equal as 70 pixels and the minimum diameter was set as 40 pixels. Totally 16 circles and 14 circles were detected in Figure 4.4 and Figure 4.5, respectively. This is clearly a problem since there are only six holes in each image. The software assigned multiple distinct circles to each hole. Furthermore, there exists one circle falsely detected in both of these results. The false detection may be caused by several reasons. First, the Hough transform is applied on the binary image generated by edge detection and thresholding, the edge detection algorithms and the choice of the threshold will affect the final detection results. Second, both the edge detection and the Hough transform are sensitive to the noise existing in the part images, such as the machining marks. Third, the choice of the diameter range is also critical. Another fact that prevents the using Hough transform in our surface inspection problem is its high computation complexity. The running time of the software for each part image is around 20 seconds that is too long for real-time inspection. For these reasons we developed an alternate approach for feature detection.

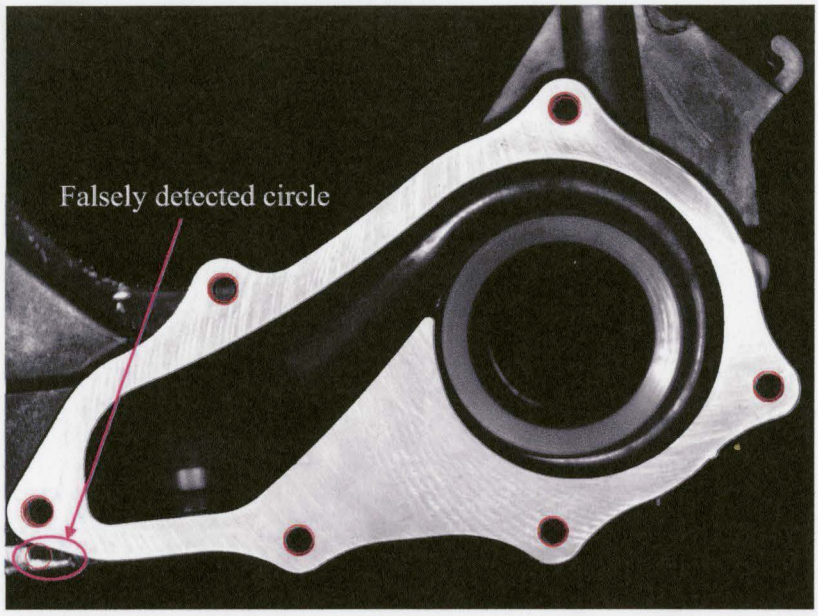


Figure 4.4 Example result 1 using Hough transform

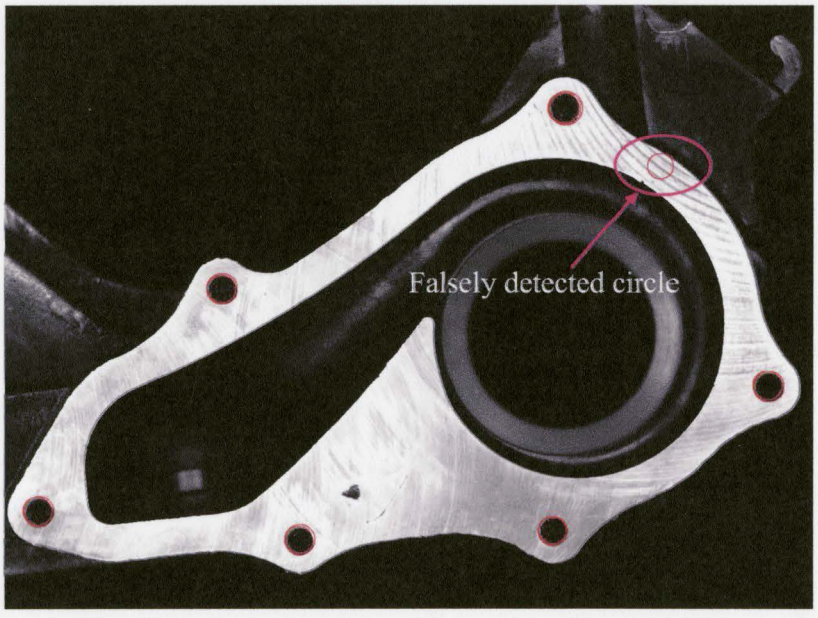


Figure 4.5 Example result 2 using Hough transform

Since the holes to be segmented all have similar shape and all have very high contrast, we developed a feature detection algorithm based on a pattern matching

technique. The pattern is a pre-generated binary image illustrating the shape of the hole, and is presented in Figure 4.6.

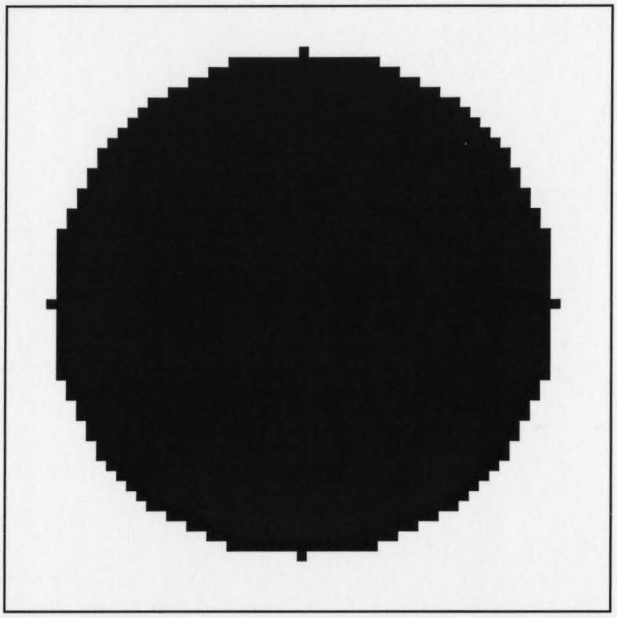


Figure 4.6 Hole pattern used for image registration.

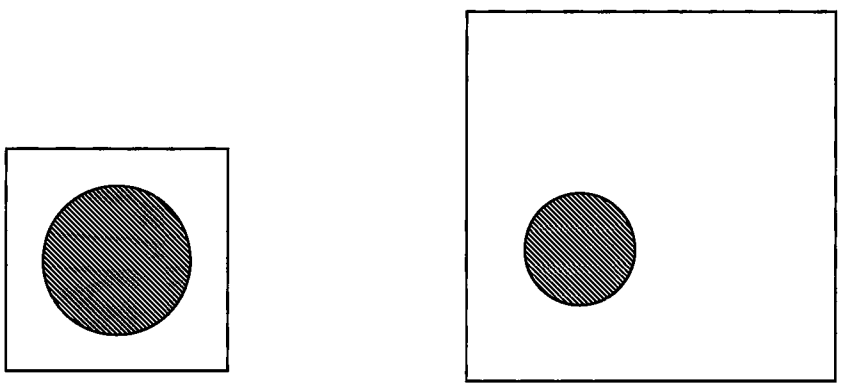
For any sub-image of pixels with top-left corner location (i_0, j_0) in the image, the cost function is defined as the difference between the pattern and the sub-image:

$$c_{i_0,j_0} = \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (h_{i,j} - I_{i_0+i,j_0+j})^2 \tag{4.1}$$

where N_{row}, N_{col} are the numbers of the rows and the columns of the hole pattern image and also the size of the sub-image; h and I are pixels from the grayscale image of the hole pattern image and the image to be processed respectively; and the subscripts of h and I indicate the row index and column index of a pixel. Our objective is to find the sub-image that minimizes c_{i_0,j_0}.

The remaining question is how to generate the pattern image, that is, how to determine the size of the hole to use in the pattern image. Since we need to find the centres of the six holes afterward, we require that the holes not be at the edges of the sub-images. Therefore, the size of the hole in the pattern image should have the same size as

the real holes in the images to be processed. Otherwise, the cost function in (4.1) may have multiple minimum values. This fact is illustrated in Figures 4.7 and 4.8. Figure 4.7 (A) shows the pattern and Figure 4.7(B) shows an image to be processed. Here we assume both the pattern and the image are binary images.



(A) (B)

Figure 4.7 Binary images to demonstrate the searching procedure. (A) Hole pattern; (B) An image to be processed.

Figure 4.8 illustrates several cases in the pattern matching procedure. In case (A), since the two holes have no intersection, the cost function achieves its maximum value. The cost function decreases in cases (B) and (C), and reaches the minimum in case (D). However, since the sizes of the holes are different in the pattern and the image, the values of the cost function are same for case (D) and (E). Actually, the cost function remains the same if the hole in the image is inside the hole in the pattern, no matter the relationship between the two holes. Therefore, if the size of the hole in the pattern is not the same as the size in the image, the pattern matching cannot guarantee that the detected hole is located in the centre of the sub-image.

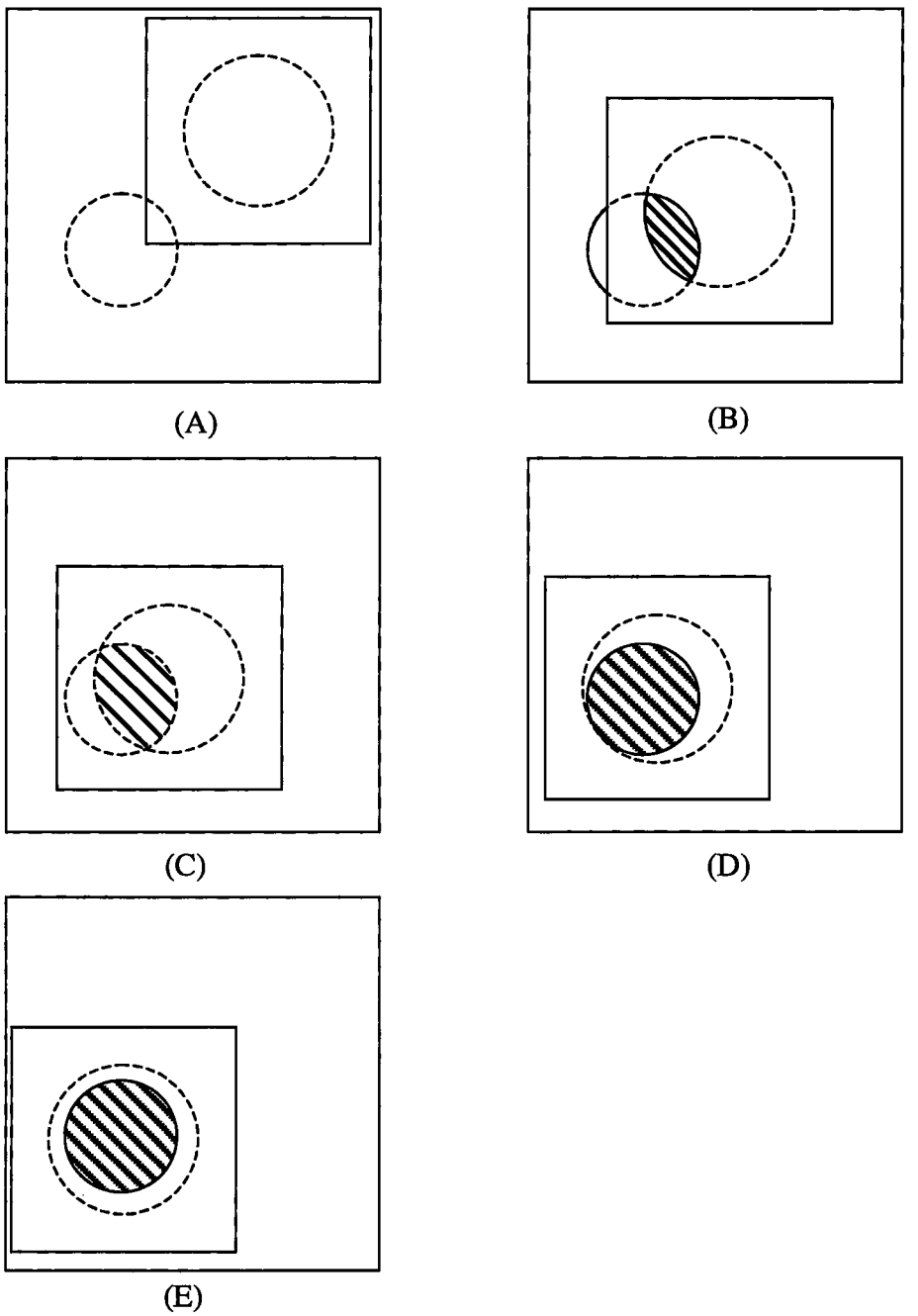


Figure 4.8 Several cases in pattern matching procedure when the hole in the pattern image is larger than the hole in the image to be processed. The cost function reaches its minimum in both case (D) and case (E).

Experiments on real acquired images also revealed this fact. Figures 4.9 to 4.11 depict the pattern matching results with different sizes of the hole in the pattern image. When the size of the hole in the pattern image is very close to the actual size of the holes in the image, the detected hole is located in the centre of the sub-image, which is illustrated in Figure 4.9. Figures 4.10 and 4.11 show the results where the hole in the pattern image is larger and smaller than the actual holes, respectively. We can see that not all the holes in Figures 4.10 and 4.11 locate in the centre of the segmented sub-images. This offset will make the holes in the sub-images not connective, and will introduce significant error in the subsequent calculation of the centres of the holes.

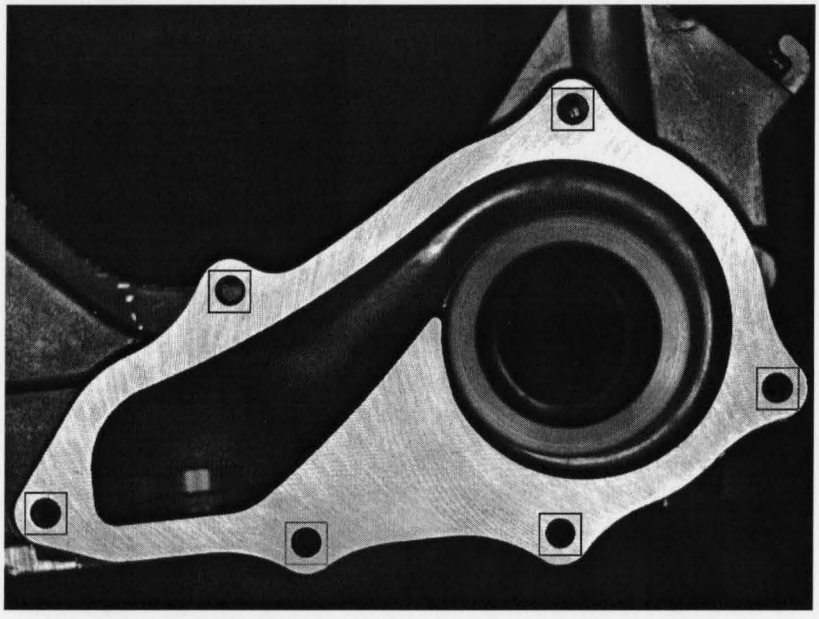


Figure 4.9 Pattern matching results when the size of the hole in the pattern image is close to the actual holes in the image.

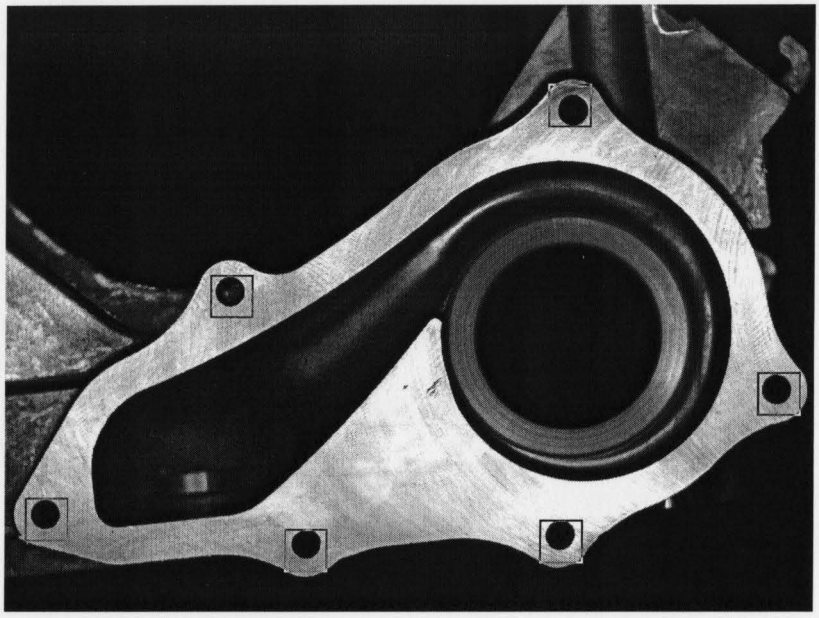


Figure 4.10 Pattern matching results when the size of the hole in the pattern image is larger than the actual holes in the image.

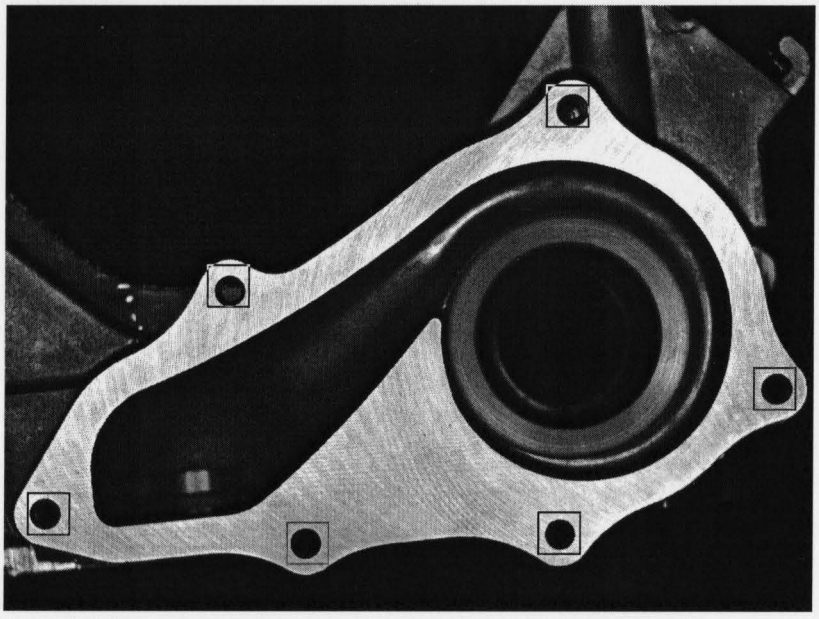


Figure 4.11 Pattern matching results when the size of the hole in the pattern image is smaller than the actual holes in the image.

According to the above analysis, we generate the pattern image as follows: First, we determine the diameter of the hole in the pattern image. The diameter of the six holes on Surface #1 is 7.5mm. After calibrating the camera, we obtain the transformation matrix from the real world coordinate system to the image coordinate system depicted in Chapter 3. So we can calculate the diameter of the hole in pixels as:

$$d_{hole} = \left\| \mathbf{H}_1 \times \begin{bmatrix} 7.5 \\ 0 \\ 0 \end{bmatrix} \right\| \approx 53 \ pixels \tag{4.2}$$

Now we generate the pattern image as a binary image with the size of 64 by 64 pixels^2. We initialize the pattern image with all zeros. Then for each pixel in the image, which is larger than 26 pixels away from the centre of the pattern image, we set its binary value as 1. The generated pattern image is shown in Figure 4.6.

In order to segment the six sub-images, we need to find the six locations where the cost function achieves its minimum. The computation burden will be tremendously heavy if we search for these minima inside the whole image. In order to save computation time, we locate the holes approximately first, and refine the locations afterward.

To locate the holes approximately and quickly, we apply the Gaussian pyramid decomposition [51]. The details of the decomposition procedure will be explained in Chapter 6. By using the Gaussian pyramid decomposition, we down-sample the image of the automotive part surface and the pattern image in the same scale. The 4-level pyramids of the surface image and the pattern image are shown in Figure 4.12 and Figure 4.13 respectively.

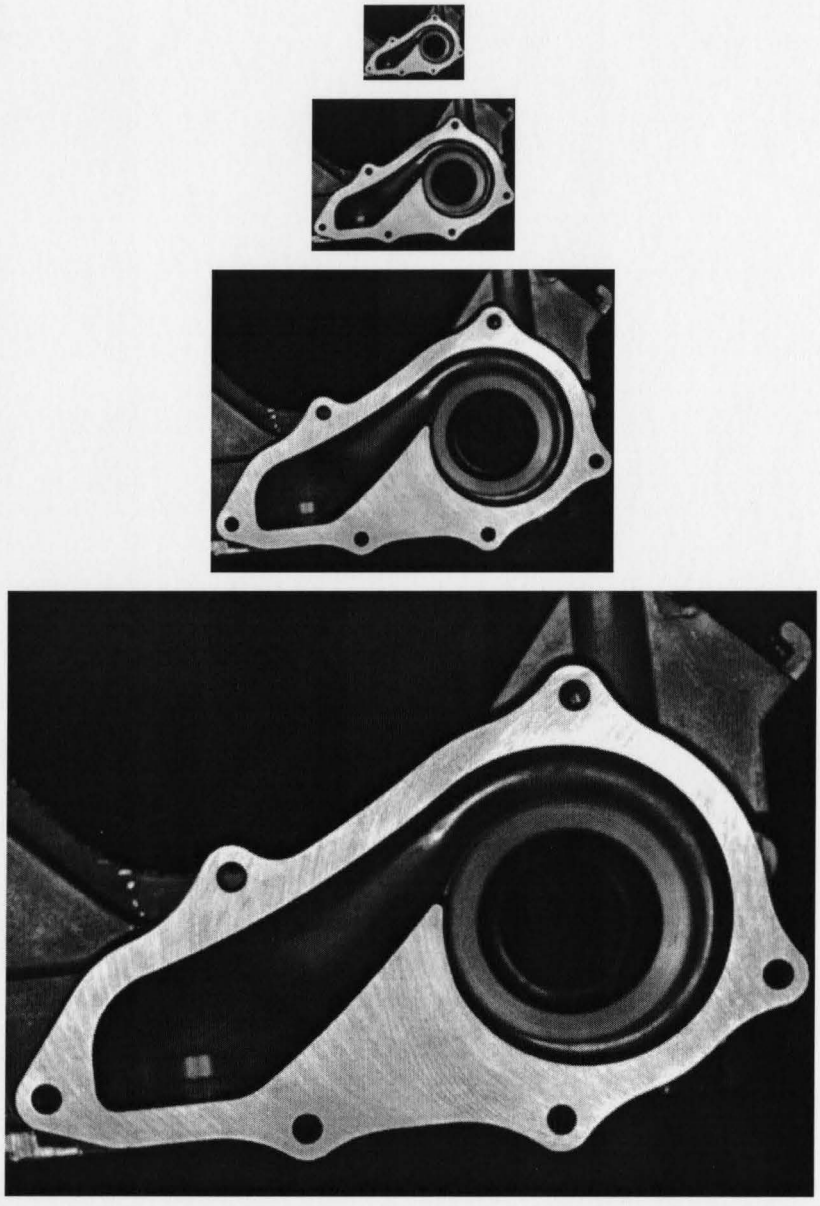


Figure 4.12 Four-level pyramid of an example surface image

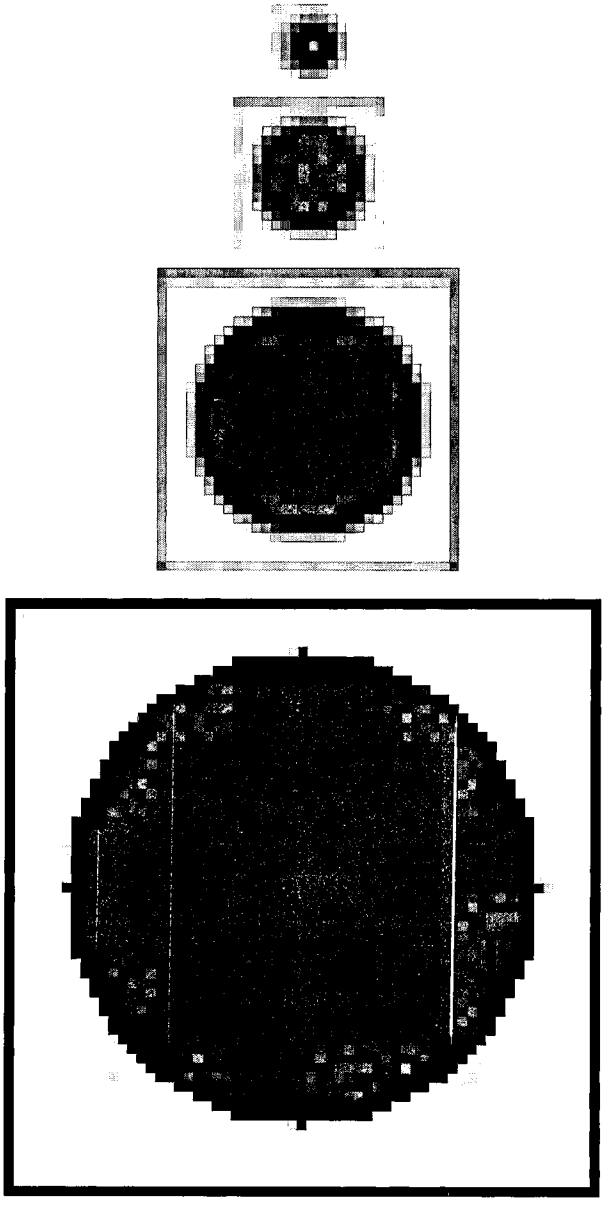


Figure 4.13 Four-level pyramid of the pattern image.

If the down-sampling order is N_d, the running time of the program will be reduced by a factor of 2^{-2N_d}. But the downsampling order cannot be too large. Otherwise, the down-sampled image will be too small to retain the feature of the original image. In our experiments we set N_d as 3 and applied the pattern matching to the top level of the pyramid. This reduced the pattern to 8 by 8 pixels^2, the image to 160 by 120 pixels^2 and the processing time by a factor of 2^{-6}.

By applying the pattern matching to the down-sampled surface image with the down-sampled pattern image, we can find the holes approximately. Then another pattern matching procedure is applied inside six small search areas to locate the holes accurately. The size of each search area is 100 by 100 pixels^2 in our experiments, centred at the approximately detected location.

To further reduce the computation complexity of the program, we apply an alternative cost function for the searching procedure as:

$$c_{i_0,j_0} = \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (h_{i,j} \bigcup I_{i_0+i,j_0+j}) \tag{4.3}$$

where the operator $\bigcup$ is an "or" operator which is defined as:

$$h_{i,j} \bigcup I_{i_0+i,j_0+j} = \begin{cases} 255, & \text{if } h_{i,j} = 1 \\ I_{i_0+i,j_0+j}, & \text{if } h_{i,j} = 0 \end{cases} \tag{4.4}$$

The application of the alternative cost function reduces the operations of multiplication and subtraction, which largely reduces the running time. However, it is worthy to mention that the alternative cost function can only be used in the search area where there is no dark area with size larger than the size of the pattern image (64 by 64). Otherwise, the cost function in (4.3) fails. No failures occurred with the water pump images.

After segmenting the six sub-images from the whole surface image, we need to determine the centre points of the holes because these centre points are the feature points we are looking for. The main idea to determine the centre points of the holes is to convert the sub-images to binary images so that the pixels in the area of the hole all have zero value. The centres of the holes will be calculated from these binary sub-images.

One simple way to convert the gray scale sub-images to binary images is using the thresholding technique. That is, all the pixels that have value larger than the threshold are assigned the value 1; otherwise, the value is set as 0. However, there are always bright spots existing in the hole due to the shape of the part and sensor noise. These bright spots will severely affect the accuracy of the calculated coordinates of the centre points of the holes. Therefore, we apply a seeded region-growing algorithm similar to Algorithm 4.1 to convert the gray scale sub-images to binary images. An original seed is chosen inside the area surrounding the hole in a sub-image. Since the hole is an area with a closed contour inside the sub-image, after region growing, the hole is segmented from the background. This method requires that the area outside of the hole is a connected area. Therefore, we require that the area of the hole be completely inside the sub-image. As previously shown, this requirement can be satisfied if the size of the hole in the pattern image is close to the size of the actual holes in the pattern matching procedure.

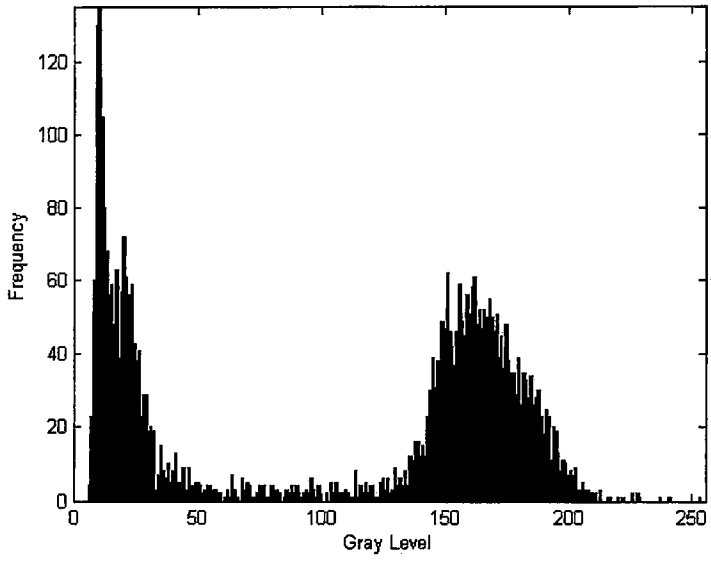
The performance of Algorithm 4.1 depends on the choice of the original seed and the threshold between the pixel value and the original seed. Since the brightness varies among the images of the different water pump housing parts, it is very difficult to choose an original seed and a threshold suitable for all of the images. To solve this problem, we developed an adaptive seeded region growing method, which adaptively chooses the original seed and the threshold based on the histograms of the sub-images.

The histogram of the sub-image is filtered first to avoid the effect of the noise. We apply a triangle filter to smooth the histogram [33], which is described as

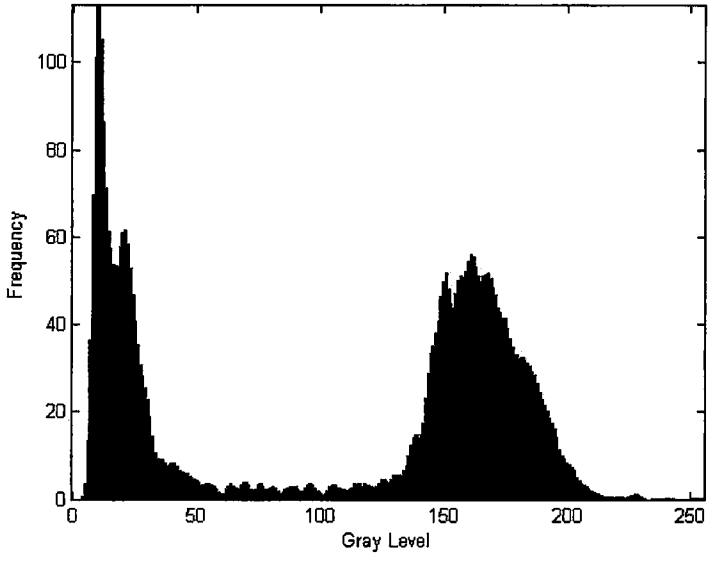
$$f(k)=\frac{1}{n^2}\left\{\sum_{i=1}^{n-1}[(n-i)\times f_0(k-i)+(n-i)\times f_0(k+i)]+nf_0(k)\right\} \tag{4.5}$$

where k is the index of the bins of the histogram , $f(k)$ is the smoothed number of pixels falling in the kth bin, $f_0(k)$ is the original number of pixels falling in the kth bin, and n is the order of the triangle filter. The choice of n depends on the noise level. Large n will smooth the histogram more than small n, but small n can keep the histogram in the original shape. In our experiments, we found $n = 3$ was an effective choice.

Figure 4.14 presents an un-smoothed and a smoothed histogram of a sub-image. As shown in the figure, the smoothed histogram of the image is bimodal, which means that there are two distinct peaks in the histogram. This bimodal shape corresponds to a sub-image including a hole with good contrast, and the two peaks correspond to the area inside and outside the hole respectively. Therefore, a value from the valley between these two peaks is a good choice of the threshold. Specifically the threshold was empirically set equal to 0.7 times the mean of the left mode plus 0.3 times the mean of right model. A pixel that has a value close to the threshold and is located outside of the hole is chosen as the original seed. The pattern image illustrated in Figure 4.6 is used as the mask to indicate the area outside of the hole.



(A)



(B)

Figure 4.14 Histograms of the sub-image: (A) Un-smoothed histogram; and (B) Smoothed histogram.

The adaptive seeded region growing method was tested on more than 100 images with different water pump housing parts. To evaluate the performance, the proposed method was compared with an adaptive thresholding method and region growing method with fixed original seed and threshold. Figure 4.15 shows the comparison on two examples. In every case, the proposed adaptive seeded region growing performed best.

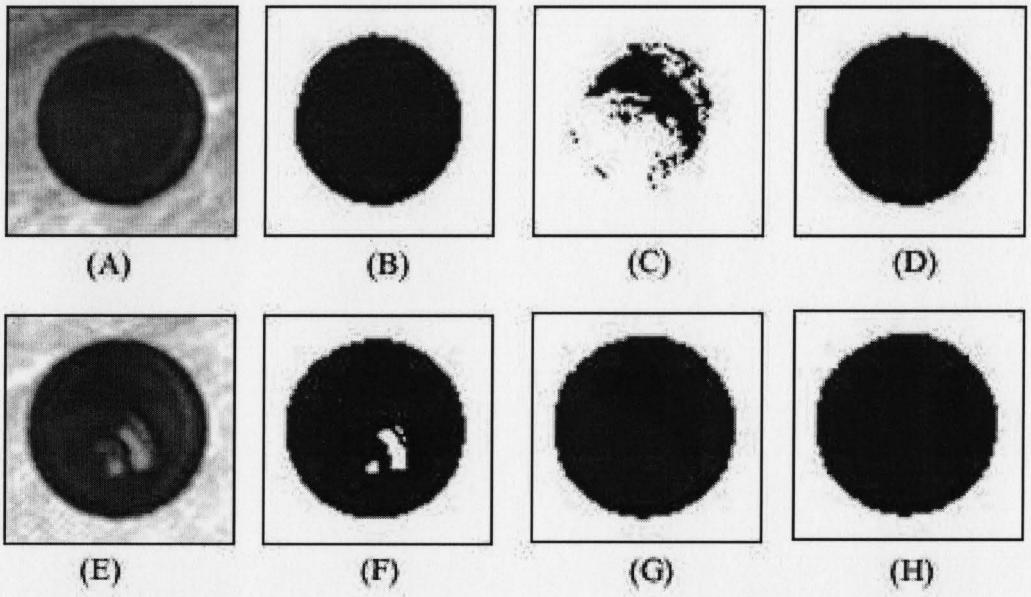


Figure 4.15 Comparison of three methods to convert the sub-images to binary sub-images: (A), (E) Example gray scale sub-images; (B), (F) Results from adaptive thresholding method; (C), (G) Results from seeded region growing algorithm with fixed seed and threshold; (D), (H) Results from proposed adaptive seeded region growing algorithm.

After the binary sub-images are obtained, the centre pixel of the hole can be located accurately by locating the centres of mass in pixels. Assuming the object pixels all have the unit mass, we have [70]:

$$\bar{x} = \frac{1}{N_{row} \times N_{col}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} jI_{i,j}$$
$$\bar{y} = \frac{1}{N_{row} \times N_{col}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} iI_{i,j} \tag{4.6}$$

where $\bar{x}$ and $\bar{y}$ are the calculated coordinates of the centre of mass in pixels, N_{row} and N_{col} are the numbers of the rows and the columns of the sub-image respectively, and $I_{i,j}$ is the value of the pixel at the ith row and jth column in the sub-image. The detected feature point is $I_{\bar{x},\bar{y}}$ located at the coordinate of $(\bar{x},\bar{y})$.

4.3.4 Feature Matching

After locating and segmenting the six holes, we need to match the detected holes with the holes on Surface #1 of the automotive part. To distinguish these six holes, we defined them as Hole #1 to Hole #6 as shown in Figure 4.3. Therefore, the objective of the feature matching process is to number the six detected holes from 1 to 6. The process is described as follows.

First, we sort based on the horizontal coordinates (i.e. columns) of the centres of the six holes. The most left hole is numbered as #1, while the most right hole is numbered as #4.

Next, we will number the remaining four holes. According to the obtained coordinates of the centres of the holes, we number the upper left one as #2, the upper right one as #3, the bottom left one as #6, and the bottom right one as #5.

4.3.5 Motion Estimation

With the detected and matched feature points (i.e. centre points of the six holes), we can estimate the motion between the standard mask and the surface image. In this thesis, we choose the affine model as the motion model, which can describe the translation, rotation, and scaling motion. The affine model is defined as [69]:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \times \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix} \tag{4.7}$$

where (x, y), (x', y') are the coordinates of the pixel in the reference image and the target image respectively. $a_i (i = 1, \cdots, 6)$ are the parameters of the affine model.

When applying the affine model, the objective of the motion estimation process is to estimate the six model parameters. We use the six pairs of matched feature points to

estimate the six model parameters. It theoretically doesn't matter whether the standard mask image is the reference image or target image in this step, however, it will make the mask registration easier if choosing the standard mask image as the target image. This will be discussed in the following section.

We denote the coordinates of these six feature points in the reference image and target image as (x_i, y_i), and (x_i', y_i'), $(i = 1, \cdots, 6)$, respectively. By applying the motion model in (4.7), we have the following equations:

$$\begin{cases} x_1 = a_1 x_1' + a_2 y_1' + a_5 \\ y_1 = a_3 x_1' + a_4 y_1' + a_6 \\ x_2 = a_1 x_2' + a_2 y_2' + a_5 \\ y_2 = a_3 x_2' + a_4 y_2' + a_6 \\ x_3 = a_1 x_3' + a_2 y_3' + a_5 \\ y_3 = a_3 x_3' + a_4 y_3' + a_6 \\ x_4 = a_1 x_4' + a_2 y_4' + a_5 \\ y_4 = a_3 x_4' + a_4 y_4' + a_6 \\ x_5 = a_1 x_5' + a_2 y_5' + a_5 \\ y_5 = a_3 x_5' + a_4 y_5' + a_6 \\ x_6 = a_1 x_6' + a_2 y_6' + a_5 \\ y_6 = a_3 x_6' + a_4 y_6' + a_6 \end{cases} \tag{4.8}$$

In matrix form, (4.8) may be rewritten as,

$$\begin{bmatrix} x_1' & y_1' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1' & y_1' & 0 & 1 \\ x_2' & y_2' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2' & y_2' & 0 & 1 \\ x_3' & y_3' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3' & y_3' & 0 & 1 \\ x_4' & y_4' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_4' & y_4' & 0 & 1 \\ x_5' & y_5' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_5' & y_5' & 0 & 1 \\ x_6' & y_6' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_6' & y_6' & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \\ x_5 \\ y_5 \\ x_6 \\ y_6 \end{bmatrix} \tag{4.9}$$

If we define

$$\mathbf{A} = \begin{bmatrix} x_1' & y_1' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1' & y_1' & 0 & 1 \\ x_2' & y_2' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2' & y_2' & 0 & 1 \\ x_3' & y_3' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3' & y_3' & 0 & 1 \\ x_4' & y_4' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_4' & y_4' & 0 & 1 \\ x_5' & y_5' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_5' & y_5' & 0 & 1 \\ x_6' & y_6' & 0 & 0 & 1 & 0 \\ 0 & 0 & x_6' & y_6' & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$\mathbf{X} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}^T \tag{4.11}$$

and

$$\mathbf{B} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & x_3 & y_3 & x_4 & y_4 & x_5 & y_5 & x_6 & y_6 \end{bmatrix}^T \tag{4.12}$$

then (4.9) may be rewritten as

$$\mathbf{AX} = \mathbf{B} \tag{4.13}$$

Now the problem is changed to a typical problem to find the solution of a group of linear equations. Since the number of the linear equations is larger than the number of the variables, the solution based on least mean square criteria is [68]

$$\mathbf{X} = \mathbf{A}^{+}\mathbf{B} \tag{4.14}$$

where $\mathbf{A}^{+}$ is the pseudo inverse matrix of matrix $\mathbf{A}$.

The best way to calculate $\mathbf{A}^{+}$ is to use singular value decomposition (SVD) [68]. The matrix $\mathbf{A}$ can be decomposed as

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} \tag{4.15}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{S}$ is a diagonal matrix with real non-negative singular values of $\mathbf{A}$. The pseudo inverse matrix $\mathbf{A}^{+}$ is calculated as following:

$$\mathbf{A}^{+} = \mathbf{V}\left(\mathbf{S}^{T}\mathbf{S}\right)^{-1}\mathbf{S}^{T}\mathbf{U}^{T} \tag{4.16}$$

Example motion estimation results are listed in Table 4.1. We can see that the error of the motion estimation is less than half a pixel. This precision was considered sufficient for use in the next processing procedure.

Table 4.1 Example results for motion estimation

Feature Points	Coordinates in Reference Image (pixels) (x, y)	Coordinates in Target Image (pixels) (x', y')	Registered Coordinates in Reference Image (pixels) (x_R, y_R)	Error (pixels) $\sqrt{(x_R - x)^2 + (y_R - y)^2}$
1	(64.41, 805.14)	(54.15, 804.15)	(64.26, 805.21)	0.17
2	(355.34, 452.22)	(345.13, 450.96)	(355.37, 452.22)	0.03
3	(896.86, 164.26)	(886.53, 162.69)	(896.90, 164.19)	0.08
4	(1221.02, 606.66)	(1210.63, 605.40)	(1220.89, 606.85)	0.23
5	(880.32, 836.88)	(870.21, 835.39)	(880.38, 836.67)	0.22
6	(477.97, 851.62)	(468.00, 850.47)	(478.13, 851.64)	0.16

4.3.6 Mask Registration

After estimating the motion between the standard mask image and the surface image to be processed, we need to register the mask image with the surface image to create a "current mask image" for further processing. As mentioned in last section, for easier processing, we define the standard mask image as the target image and the current mask image as the reference image. To reduce the computational cost, we preset the current mask image as all zeros and only process the pixels whose value equals 1 on the standard mask image. Let $p'_{i,j}$ represent such a pixel located in the *i*th row and *j*th column on the standard mask image. Applying the affine motion model, we have:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \times \begin{bmatrix} j \\ i \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix} \tag{4.17}$$

where (x, y) is the registered coordinates in the current mask image. Unlike with regular images, the coordinates are floating point numbers. Assume $p_{m,n}$ is a pixel of the moved mask image in *m*th row and *n*th column and satisfies:

$$\begin{cases} m \le y < m+1 \\ n \le x < n+1 \end{cases} \tag{4.18}$$

then the pixel (x, y) is located in the area shown as Figure 4.16.

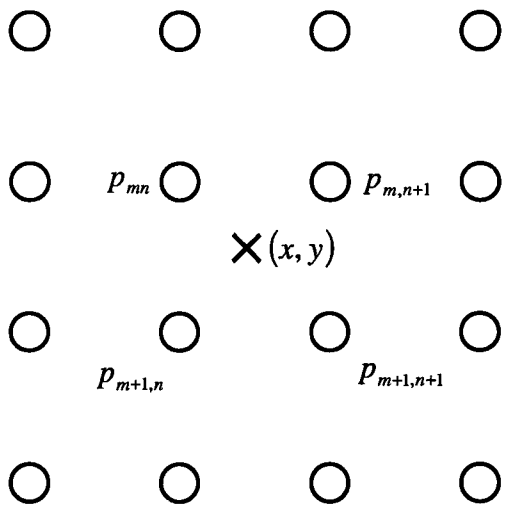


Figure 4.16 Spatial relationship between the registered pixel and the pixels on the current mask image.

Since (x, y) does not lie on the integer grid, its gray scale value is obtained by traditionally some interpolation method. However, in our application, the interpolation is unnecessary since the mask image is defined as a binary image. Instead, we simply set the surrounding pixels p_{mn}, $p_{m,n+1}$, $p_{m+1,n}$, and $p_{m+1,n+1}$ equal to 1 on the current mask image. One problem is that this method will cause the current mask image to be larger than the original mask, which may result in a wrong decision during the latter defect inspection. Fortunately, this problem can be solved by a technique that will be described in the next chapter.

4.3.7 Summary of the Accurate Image Registration Algorithm

The accurate image registration algorithm is summarized below.

Algorithm 4.2 Accurate Image Registration

Step 1: Apply Gaussian pyramid decomposition to down-sample the pattern image and the surface image with $N_D = 3$;

Step 2: Apply the pattern matching method to the down-sampled pattern image and the down-sampled surface image to obtain the approximate positions of the holes in the surface image;

Step 3: Generate the search areas with respect to the approximate hole locations in the surface image;

Step 4: Accurately search inside the search areas to locate the holes;

Step 5: Segment the sub-images of the holes;

Step 6: Calculate the histograms of the segmented sub-images;

Step 7: Smooth the histograms;

Step 8: Determine the threshold used for the seeded region growing algorithm according to the smoothed histograms;

Step 9: Find the original seed used for the seeded region growing algorithm according to the threshold;

Step 10: Apply Algorithm 4.1 to convert the sub-images to binary images;

Step 11: Calculate the coordinates of the centres of the six holes on the surface image;

Step 12: Number the detected holes matching with those in the standard mask image;

Step 13: Estimate the parameters of the affine model with the obtained coordinates of the centres of the six holes;

Step 14: Apply the affine model to every pixel of the mask image to generate the current mask image.

Example experimental results are shown in Figures 4.17 and 4.18, where the mask image is shown in the red channel of the colour image, and the surface image is shown in the green channel. As a result, the overlapped area appears as a yellow colour.

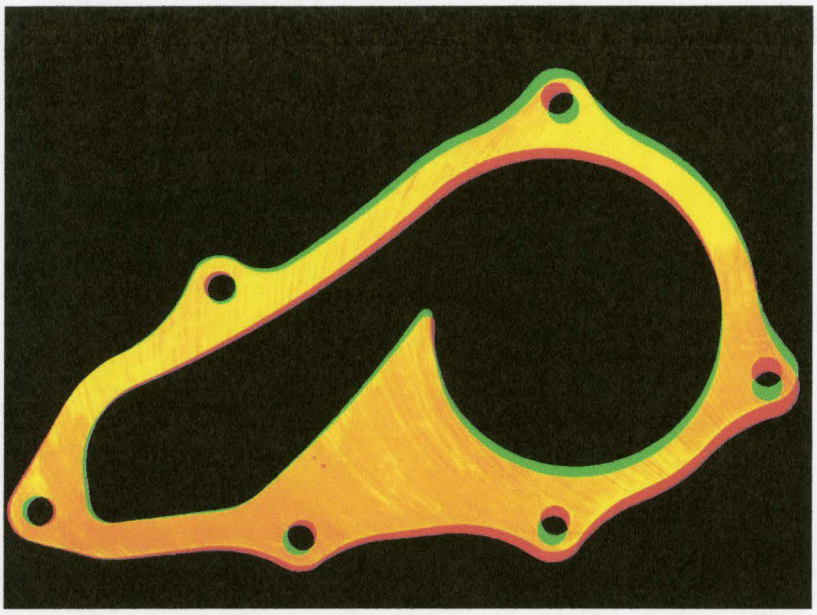


Figure 4.17 Result with unregistered standard mask.

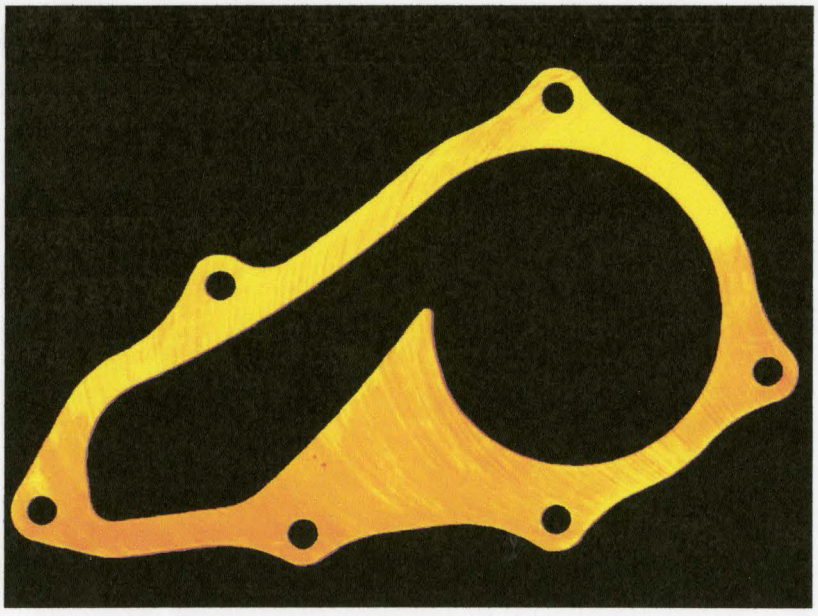


Figure 4.18 Result with registered current mask. .

Figure 4.17 shows the result by using the unregistered standard mask. From the figure, we can see that the mask and the surface image are mismatched along the edges of Surface #1 in the image. Therefore, some pixels outside of Surface #1 will be included and some information within the ROI will be missed if using this unregistered mask directly. This mismatch will introduce very large error in the subsequent inspection procedure. As can be seen from Figure 4.18, by using the current registered mask the mismatch is removed.

4.3.8 Simplified Image Registration Algorithm

As mentioned above, during the image acquisition procedure we only require the statistical information of the ROI in the surface image as the feature value for the adaptive camera control algorithm. It is not necessary to very accurately match the mask up with the surface image. Also, the computation of the accurate image registration algorithm described in the previous section is too heavy for the image acquisition procedure. Therefore, we developed a faster simplified image registration algorithm for use in the image acquisition procedure.

The main simplification involves the motion model. As previously discussed, it is assumed that there only exists translation and rotation between different images. Therefore, only two feature points need to be detected to estimate the motion. The key idea of the simplified algorithm is shown in Figure 4.19 in which two images are registered by translation and rotation.

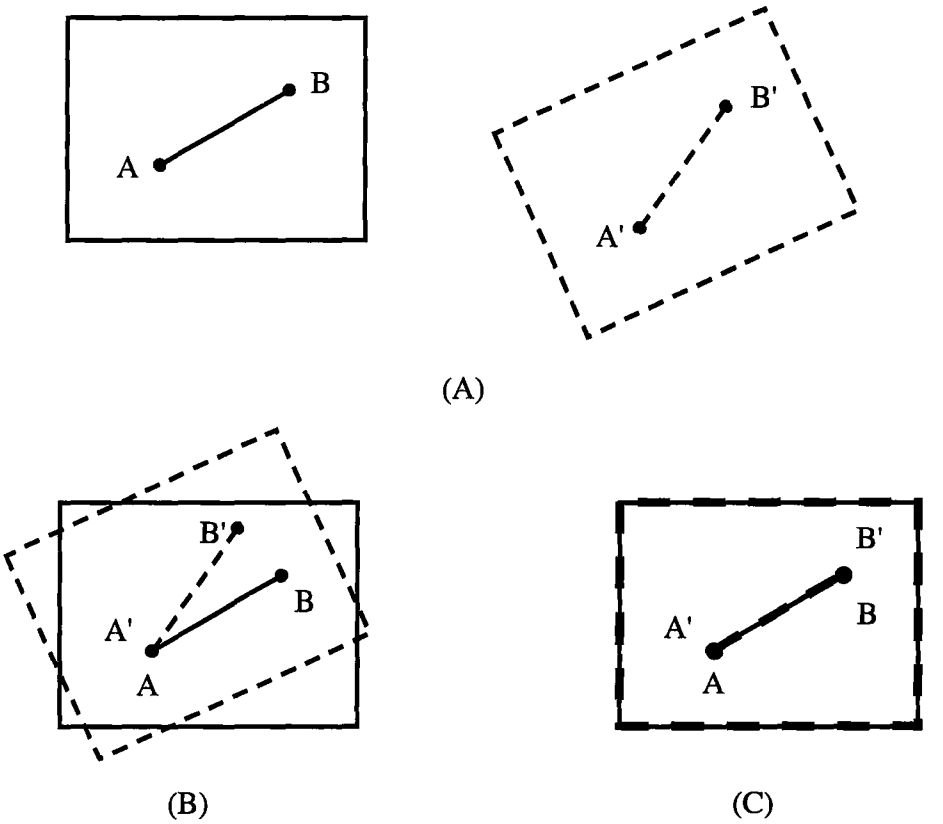


Figure 4.19 Simplified image registration. (A) Original images. (B) Images after translation. (C) Images after rotation.

Let A, A' and B, B' be the two pairs of feature points; and (x_A, y_A), $(x_{A'}, y_{A'})$, (x_B, y_B) and $(x_{B'}, y_{B'})$ are coordinates of A, A', B and B', respectively. Then we have:

$$\begin{cases} \Delta x = x_A - x_{A'} \\ \Delta y = y_A - y_{A'} \\ \\ \Delta\theta = \arctan\left(\dfrac{y_A - y_B}{x_A - x_B}\right) - \arctan\left(\dfrac{y_{A'} - y_{B'}}{x_{A'} - x_{B'}}\right) \end{cases} \tag{4.19}$$

The registration with such a motion model is very simple and fast. Specifically, with a pixel (x_0, y_0) in the standard mask image, we first translate the coordinates to

$$\begin{cases} x_1 = x_0 + \Delta x \\ y_1 = y_0 + \Delta y \end{cases} \tag{4.20}$$

Then we rotate the image and get the registered coordinates as

$$\begin{cases} x = x_1 \cos(\Delta\theta) - y_1 \sin(\Delta\theta) \\ y = y_1 \cos(\Delta\theta) + x_1 \sin(\Delta\theta) \end{cases} \tag{4.21}$$

Comparing to the accurate image registration algorithm developed in the previous section, the motion estimation process is faster since we does not need to calculate a pseudo inverse of a matrix. Instead, we obtain the parameters of the motion model through (4.19).

The simplified image algorithm can be summarized as follows.

Algorithm 4.3 Simplified Image Registration

Step 1: Search for two holes (Hole #1 and Hole #4) as the features by applying the same method described in Algorithm 4.2;

Step 2: Calculate the histograms of the two segmented sub-images;

Step 3: Smooth the histograms;

Step 4: Convert the sub-images to binary images according to the smoothed histograms;

Step 5: Find the coordinates of the centres of the holes;

Step 6: Estimate the parameters of the motion model using (4.19);

Step 7: Register the standard mask image and generate the current mask image in the same manner as in Section 4.3.6.

An example result obtained using Algorithm 4.3 is shown in Figure 4.20. Comparing this with Figures 4.17 and 4.18, by using the simplified image registration algorithm the mask is approximately registered with the surface image. Although the performance is inferior to the accurate image registration algorithm, the processing speed is almost 3 times faster than the accurate image registration algorithm, and the matching precision meets the need of the image acquisition procedure. Detailed timing results will be presented in the next chapter.

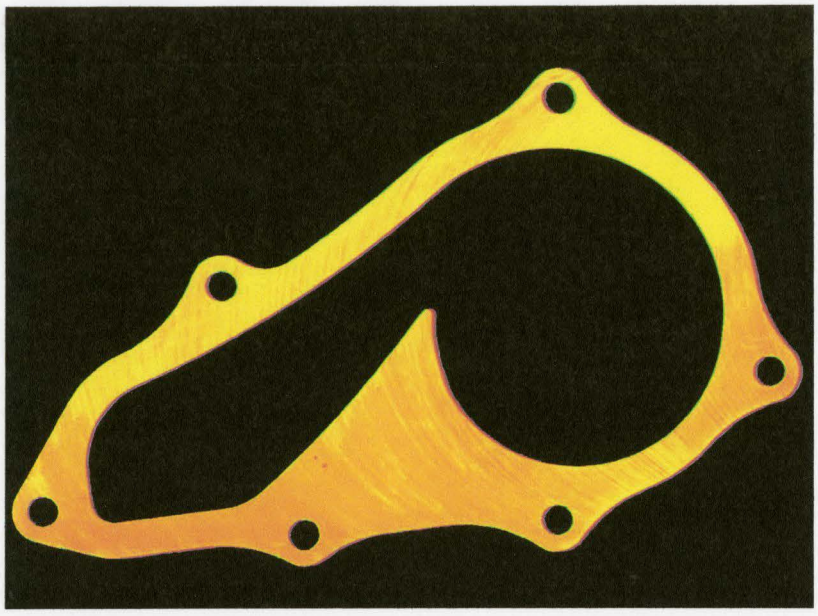


Figure 4.20 Result using simplified image registration algorithm.

4.4 Summary

In this chapter, we presented a seeded region growing based algorithm to generate the ROI mask from one acquired image. An accurate image registration algorithm was proposed to compensate the global motion between the ROI mask and the surface to be inspected. We also developed a simplified image registration algorithm for use during the image acquisition. Experiments showed that the accurate image registration algorithm had higher accuracy but longer program running time, while the accuracy of the simplified one satisfied the requirements of the image acquisition.

Chapter 5

Adaptive Machine Vision System for Surface Inspection: Part Classification

5.1 Introduction

By applying the registered mask, Surface #1 has been segmented from the automotive part image. Next, we will inspect the segmented Surface #1 to classify the automotive parts into acceptable or unacceptable categories according to the size and location of the defect(s). This chapter discusses the algorithms for inspecting Surface #1.

This chapter is organized as follows. First, the algorithm to detect the defective pixels on Surface #1 is described. Second, we describe an algorithm for removing falsely classified pixels outside the edges of Surface #1. Third, the algorithm to measure the size of the defect is discussed. Next, the sealing area identification and part classification algorithm is presented. Finally, the chapter closes with extensive experimental results and conclusions.

5.2 Detection of Defective Pixels

5.2.1 Overview

By applying the accurate image registration algorithm, we can match the mask up with Surface #1 of the automotive part image accurately. Then Surface #1, the ROI in this inspection problem, is segmented from the image to prevent unimportant areas of the surface from being inspected. An example of a segmented ROI is shown in Figure 5.1.

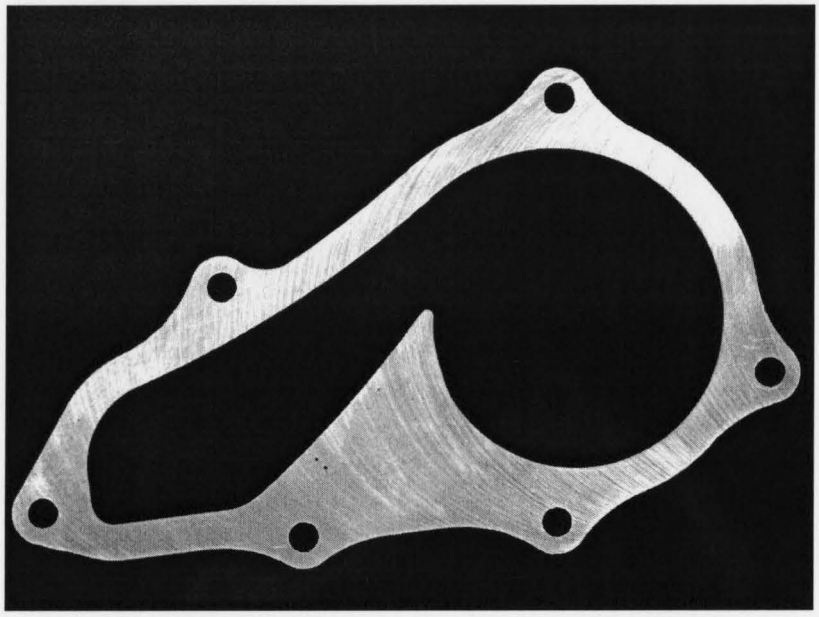


Figure 5.1 Example of an ROI to be inspected.

Next we need to detect the defective pixels in the segmented ROI. First of all, we need to find a way to describe the defects on Surface #1. By manually analyzing the automotive water pump housing parts, we found that the defects on the Surface #1 do not have regular shapes. The irregular shapes of the defects make it impossible to detect them by applying methods like pattern matching. Also, the images of different automotive parts change over a very large range. So it is difficult to detect the defects by comparing with a reference version of Surface #1. From the part images, we found that there are always sudden changes in the brightness between the defective areas and the smooth areas of

ROI. So we will analyze the variance of the pixel gray scale values to find the potential defects.

In this inspection problem, there always are machining marks on the Surface #1s for most of the images. The machining marks display obvious brightness change in the acquired images that should be ignored during the inspection procedure. Distinguishing the brightness variation for the defects from the variation for the machining marks makes this inspection problem a challenging one.

The following approaches were evaluated:

- Gradient operator followed by global thresholding;
- Adaptive local thresholding of the original image;
- Local standard deviation followed by global thresholding;
- Local standard deviation followed by adaptive global thresholding.

Details of the algorithms and sample experimental results will be provided in the next four sections. Note that these algorithms assume that the mask matches up with Surface #1 perfectly. The correction for mismatches at the edges will be addressed in Section 5.3.

5.2.2 Gradient Operator Followed by Global Thresholding

In this surface inspection problem, we have observed that the variation of the gray level in the automotive part image can describe the defects on Surface #1. So the most obvious methods to try first are the gradient operators and edge detectors. We tried using the Sobel gradient operator, Laplace differential operator and the Canny edge detection algorithm followed by global thresholding to detect the defective pixels.

The Sobel gradient operator is applied to calculate the first, second, third or mixed image derivatives by convolving the image with the appropriate kernels [66]:

$$dst(x,y) = \frac{d^{Nx+Ny} src(x,y)}{dx^{Nx} \cdot dy^{Ny}} \tag{5.1}$$

where (x,y) is the location of the pixel in the images, $dst(x,y)$ is the gray scale value of the pixel in the result image, $src(x,y)$ is the gray scale value of the pixel in the source

image, *Nx*, *Ny* are the orders of the differential operation in horizontal direction and vertical direction, respectively. Most often, the Sobel gradient operator is used to calculate the first order horizontal or vertical image derivative. The horizontal and vertical kernels are [70]:

$$\mathbf{M}_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.2}$$

and

$$\mathbf{M}_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{5.3}$$

The Laplace differential operator calculates the Laplacian of the image. The Laplacian of an image is defined as: [66]

$$L(x,y) = \frac{d^2 src(x,y)}{dx^2} + \frac{d^2 src(x,y)}{dy^2} \tag{5.4}$$

where (x, y) is the location of the pixel in the images and $src(x, y)$ is the gray scale value of the pixel in the source image.

The operation of the Laplace differential operator is to convolve the image with the following kernel:

$$\mathbf{M}_l = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.5}$$

The Canny edge detection algorithm was developed in 1986 [42]. It uses a multi-stage algorithm to detect a wide range of edges in images.

The experimental results by applying the Sobel gradient operator, Laplace differential operator and the Canny edge detection algorithm to the ROI from Figure 5.1 are shown in Figures 5.2, 5.3, and 5.4, respectively. For convenience, these figures show the negative images.

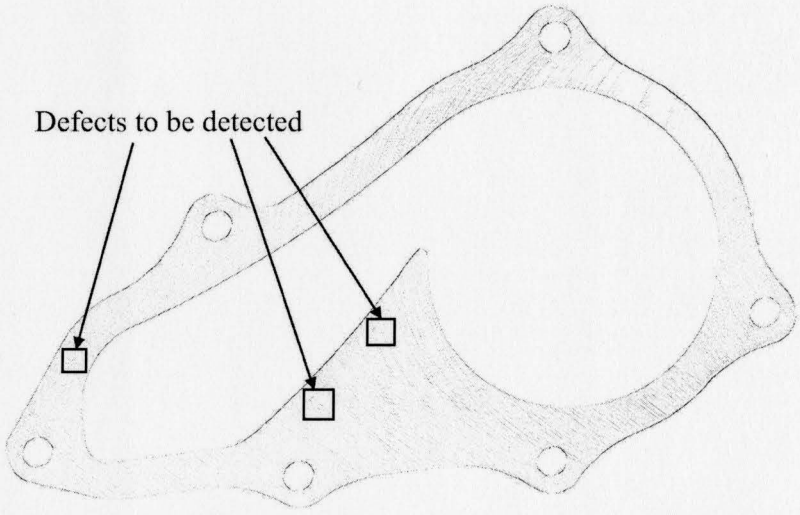


Figure 5.2 Result by applying Sobel gradient operator.

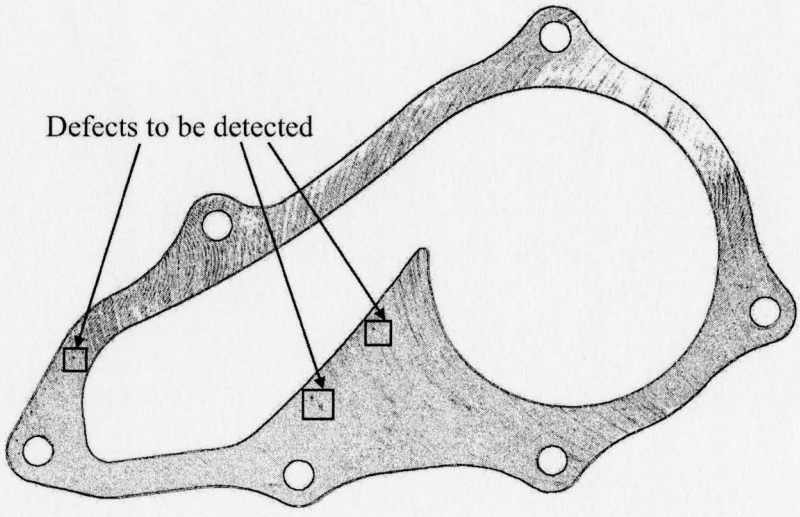


Figure 5.3 Result by applying Laplace differential operator.

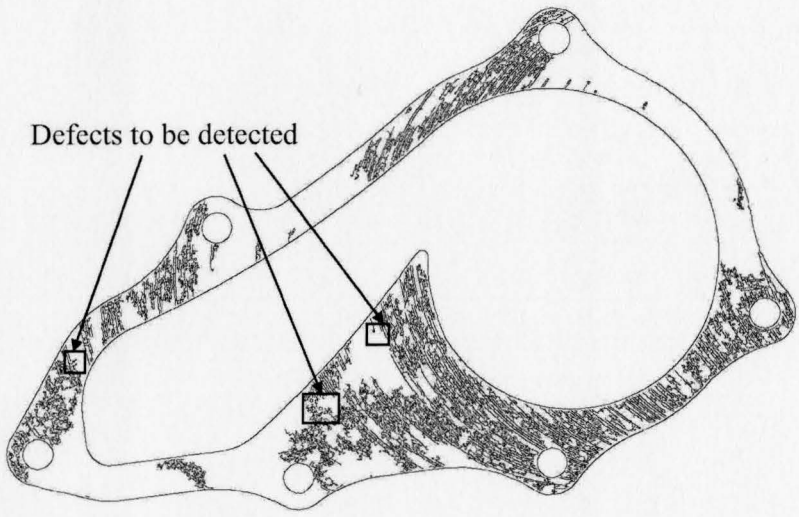


Figure 5.4 Result by applying Canny edge detector.

From Figures 5.2 and 5.3, we can observe that after applying the Sobel gradient operator or Laplace differential operator, the defects and the machining marks have similar gray levels. We tried to distinguish the defects and machining marks using global thresholding with a manually chosen threshold. Unfortunately, a threshold could not be found that separated the defects and machining marks for all of the part images.

By using the Canny edge detector, both strong and fine edges are detected, which can be seen in Figure 5.4. However, the machining marks and the defects are still not separated. Therefore, no gradient operators or the edge detectors could both detect the defective pixels on the Surface #1 and ignore the machining marks.

5.2.3 Adaptive Local Thresholding of the Original Image

Thresholding is a technique that is commonly applied to segment an image by setting all pixels whose intensity values are larger (or in some cases smaller) than the given threshold as an object value and all the remaining pixels as a background value. Conventionally, the thresholding operator uses a global threshold for all pixels in an image. However, the distributions of the intensity values of the pixels in different images

are different. Furthermore, for each image, the distributions of the intensity values vary in different areas. Therefore, it is impractical to use a fixed global threshold for every image. We need to adaptively select the threshold for our defect detection problem. Note that for our problem the defects appear darker than the rest of the surface.

There are two main approaches to adaptively determining the threshold: (i) Chow and Kaneko's approach [32] and (ii) local thresholding. Both methods are based on the following assumptions:

1. Smaller image regions are more likely to have approximately uniform illumination;
2. There are no significant brightness changes in the smooth areas of the image under even illumination.

With Chow and Kaneko's adaptive thresholding approach, an image is divided into an array of overlapping sub-images and the optimum threshold for each sub-image is determined by investigating its histogram. This approach is expensive in computation unless the ROI is small. Since Surface #1 contains about 257,558 pixels it is inappropriate to use Chow and Kaneko's approach for our defect detection problem.

The local thresholding method determines a local threshold for every pixel in the image. We applied this method to detect defective pixels on Surface #1. The local threshold for each pixel is chosen as a statistical value from its neighbourhood in a local window. In our experiments, we tried the mean, the sum of 0.7 times the minimum plus 0.3 times the maximum, and the median of the pixels as the local threshold.[39] The local window was 3×3 pixels^2. These results are shown in Figures 5.5 to 5.7. From these figures, we can see that although the defects are detected and even a little enhanced, they are still indistinguishable from the machining marks.

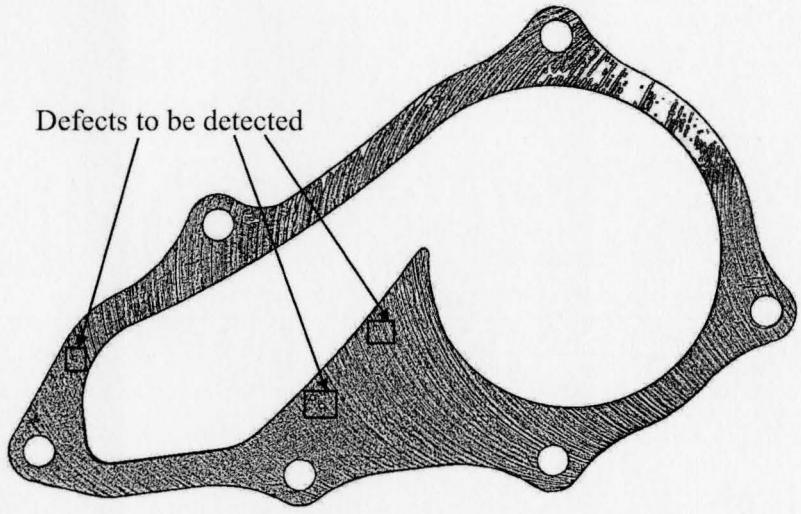


Figure 5.5 Result by local thresholding using a threshold equal to the local mean.

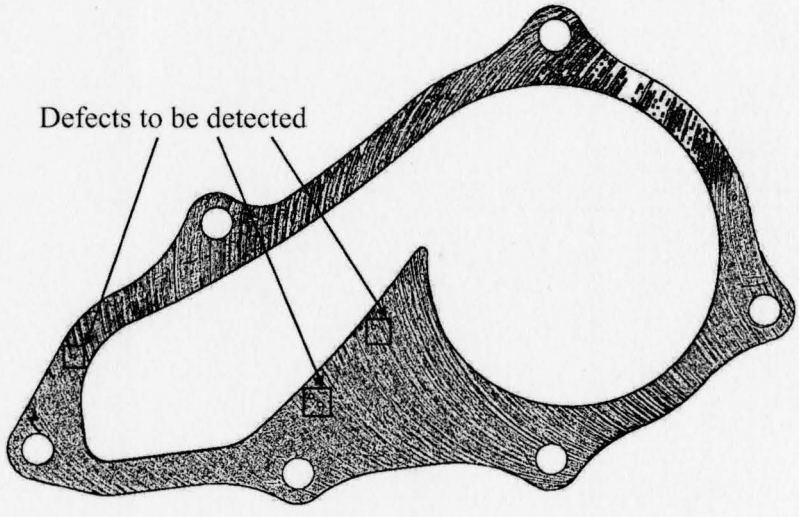


Figure 5.6 Result by local thresholding using a threshold equal to a weighted sum of the local minimum and maximum.

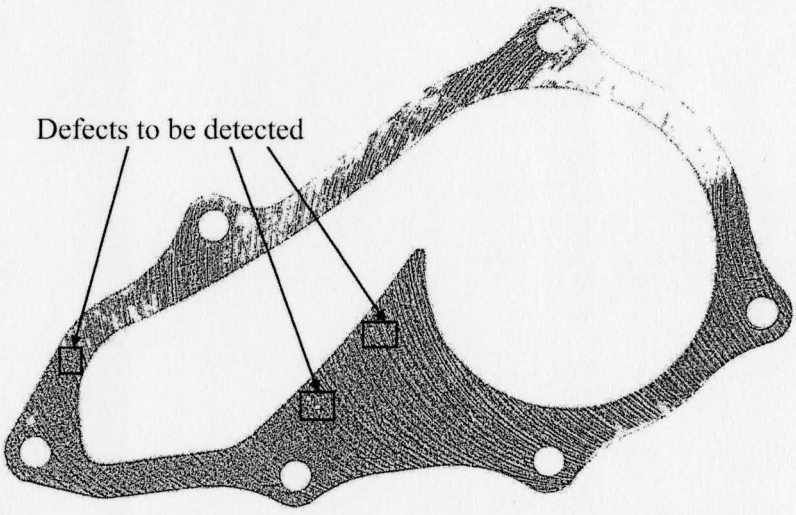


Figure 5.7 Result by local thresholding using a threshold equal to the local median.

5.2.4 Local Standard Deviation Followed by Global Thresholding

The previous two approaches failed to distinguish the defects from the machining marks because the brightness changes for both of them. However, we observed that most machining marks display as regular textures and have a relatively large area. Based on this observation, we tried generating a local standard deviation image from each part image to enlarge the gap between the defects and the machining marks.

The standard deviation of a group of data describes their variation. We use the standard deviation of the neighbouring pixels in a local window of a pixel to depict the local variation of the brightness. We have observed that the brightness variation of the machining marks for most automotive part images is not as sharp as that of the defects. So we should be able to use the standard deviation to detect the potential defects and ignore the machining marks on Surface #1.

We will generate a standard deviation image for the automotive part image using a method similar to [75]. The gray scale value of every pixel in the standard deviation image is equal to the standard deviation value of a small window whose centre is the

relative pixel in the original image. The generated standard deviation image varies with the different size of the window used. We experimentally determined that a 3×3 pixel^2 window produces the best inspection results.

For a pixel in the acquired image, the pixel and its 8 neighbours compose the 3 by 3 window. This is illustrated in Figure 5.8. For every 3 by 3 window, we first calculate the mean value as [72]:

$$mean = \frac{1}{9}\left(\sum_{i=1}^{8} x_i + y\right) \tag{5.6}$$

where y is the gray scale value of the current pixel in the part image, $x_i (i = 1, \cdots, 8)$ are the gray levels of the 8 neighbours of the current pixel. Next, we compute the variance of the image window as [72]:

$$var = \frac{1}{8}\left[\sum_{i=1}^{8} (x_i - mean)^2 + (y - mean)^2\right] \tag{5.7}$$

Then we can obtain the standard deviation value as:

$$std\_dev = \sqrt{var} \tag{5.8}$$

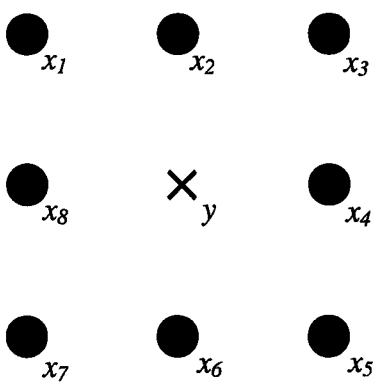


Figure 5.8 Diagram of a 3 by 3 local window [74].

Figure 5.9 shows an example of a calculated standard deviation image. We can see that the defects have the highest gray values, the smooth areas have the lowest gray values, and the values due to the machining marks are in the middle. So we should be

able to detect the defects on Surface #1 while ignoring the machining marks using standard deviation images.

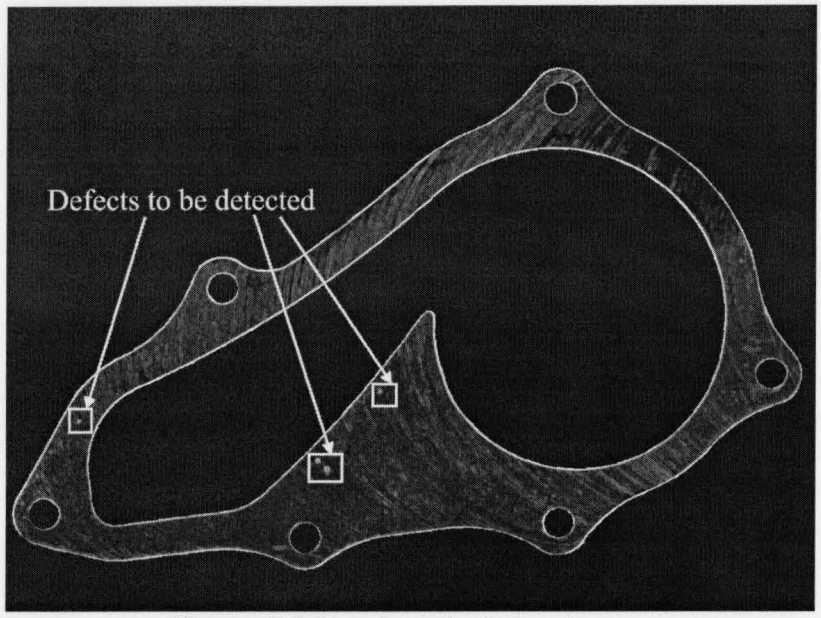


Figure 5.9 Standard deviation image.

Next, we tried to segment the defects on Surface #1 based on the obtained standard deviation image. Global thresholding with a fixed threshold was tested first as it is the most computationally efficient approach. The results varied with the selected threshold. Figures 5.10 and 5.11 show two segmentation results with different thresholds (negative images are shown in the figures). In Figure 5.10, a small threshold of 25 was selected to segment the defective and non-defective pixels on Surface #1. We can see that all the defects are detected. However, some of the machining marks are also detected. In Figure 5.11, a larger threshold of 55 was applied. The machining marks are ignored in the image result, but some defects are also missed. Therefore, a properly chosen threshold is very important for our inspection problem. A method to adaptively choose a global threshold will be introduced in the next section.

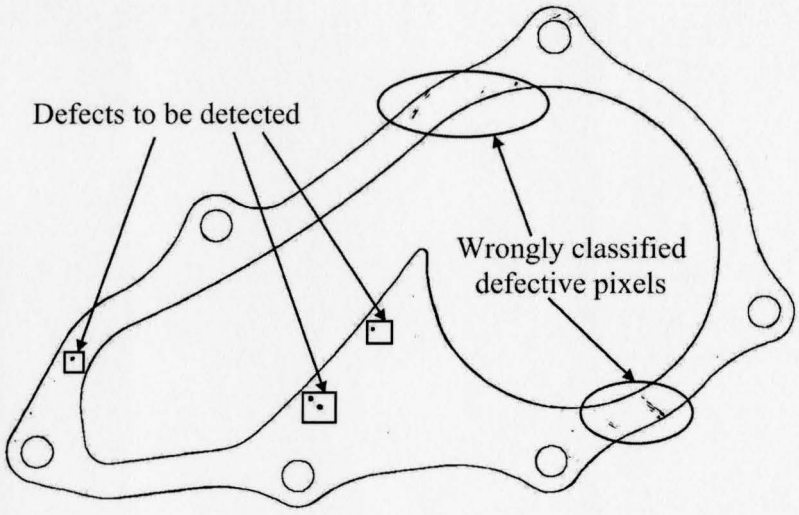


Figure 5.10 Thresholded standard deviation image using a small global threshold.

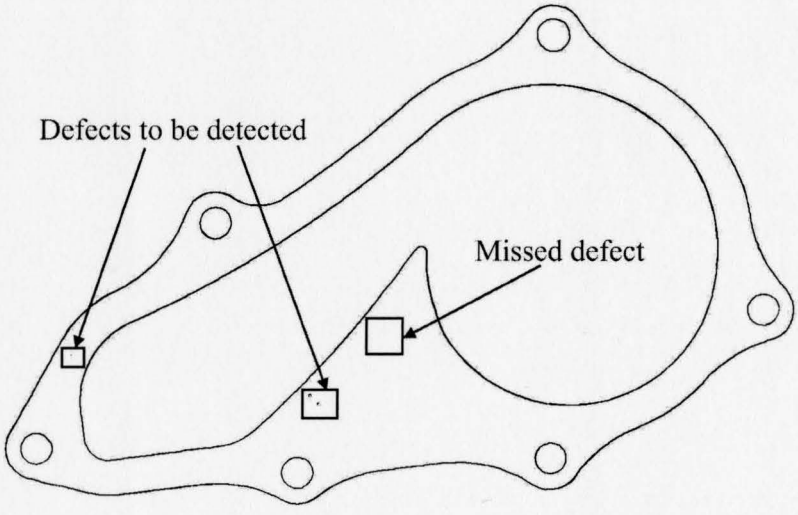


Figure 5.11 Thresholded standard deviation image using a large global threshold.

5.2.5 Local Standard Deviation Followed by Adaptive Global Thresholding

In order to detect the defects and ignore the machining marks at the same time, we will use adaptive global thresholding to segment the defective pixels from the standard deviation image. By applying the histogram-based thresholding method employed in the simplified image registration algorithm we can segment the standard deviation image to detect the potential defects on Surface #1 adaptively. Specifically the threshold was empirically set equal to 0.77 times the mean of the left mode plus 0.23 times the mean of right mode. The algorithm is presented at the end of this section. The segmentation result for the ROI shown in Figure 5.1 is shown in Figure 5.12.

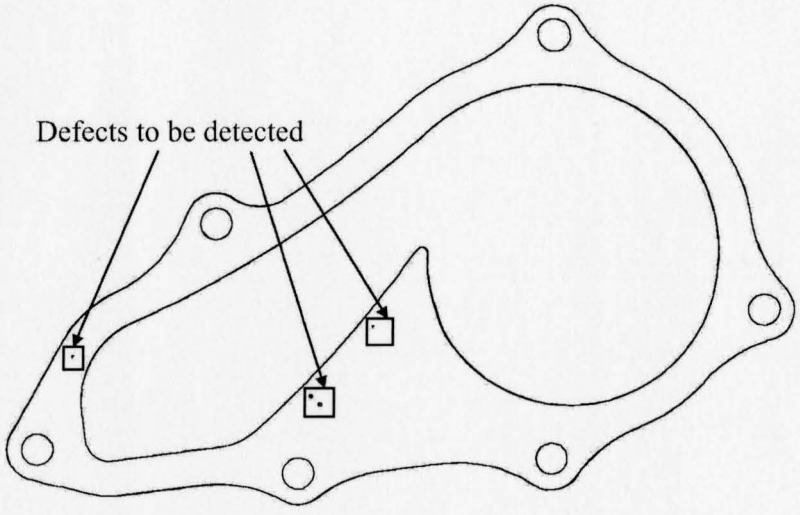


Figure 5.12 Segmented ROI obtained by local standard deviation followed by adaptive global thresholding method.

In Figure 5.12, the machining marks are ignored while the defective pixels are detected. Unfortunately, the edge contours of Surface #1 are also detected and the pixels depicting the contours are wrongly detected as defective. This is because the mask generated by applying the seeded region growing algorithm is larger than Surface #1. An algorithm to remove the falsely classified pixels outside the edges of Surface #1 is presented in the following section.

Algorithm 5.1 Local Standard Deviation followed by Adaptive Global Thresholding

Step 1: Initialize the standard deviation image with zeros;

Step 2: For every pixel in the acquired image, check the corresponding pixel in the mask image: If the grayscale value of the pixel in the mask image equals 0, start again with the next pixel. Else, calculate the standard deviation of the 3 by 3 local window of the pixel, and save the local standard deviation value in the standard deviation image;

Step 3: Calculate the smoothed histogram for the generated standard deviation image and calculate the threshold from the smoothed histogram by applying the method described in the simplified registration algorithm; and

Step 4: Segment the standard deviation image into defective and non-defective pixels by applying the obtained global threshold. The thresholded pixels with the binary values equal to 1 indicate the location of defective pixels. The non-defective pixels are set equal to zero.

5.3 Removal of Falsely Classified Edge Pixels

5.3.1 Algorithm Overview

We applied a mask to eliminate the unimportant areas outside of Surface #1 to reduce the computation cost. However, the mask generated using Algorithm 4.1 is slightly larger than Surface #1. As a result, the pixels outside the edges of Surface #1 are wrongly classified as defective by Algorithm 5.1. To remove the falsely classified edge pixels from the segmented ROI image we use the contours of the mask image as approximations of the edges of Surface #1. The algorithm to remove the pixels at the edges is presented in this section.

Since the mask is a binary image and the ROI image is a gray scale image, it is easier to obtain the contours of the mask. Also, by using Algorithm 4.2, the mask matches Surface #1 accurately. It is therefore reasonable to take the contours of the mask as a good approximation of the edges of Surface #1. We check the neighbourhood of every defective pixel to determine whether the defective pixel is on the edge of Surface #1 or not. If the pixel does not belong to Surface #1 then it has been wrongly classified as defective and reclassified as non-defective.

5.3.2 Assumptions

The algorithms in this section assume the following:

1. The window size used to check the edge pixels is defined;
2. The contours of the mask are closed.

The first assumption was satisfied by setting the window size manually using images of the 23 parts that were available during the software development.

Since the algorithm to detect the contours of the mask ends when the searching procedure reaches the first pixel for the second time, we require that each contour of the mask be closed. Properly adjusting the camera and the automotive part fixture when setting up the work cell such that a high contrast image is available to generate the mask and the field of view includes all of Surface #1 satisfies this assumption.

5.3.3 Algorithm

We used the left-right search algorithm from [73] to find these contours. An example of the mask contours is shown in Figure 5.13. In Figure 5.13, the contours of the mask are shown in red while Surface #1 is shown in gray scale. We can observe that the registered mask contours are very good approximations for the edges of Surface #1.

Figure 5.13 Mask contours shown in red, overlaid on a gray scale image of Surface #1.

After computing the contours of the mask, we need to remove pixels outside the edges of Surface #1 that have been wrongly classified as defective. They cannot belong to defects, since they do not belong to Surface #1. The algorithm is given below.

Algorithm 5.2 Removal of Wrongly Classified Edge Pixels

Step 1: Push the defective pixels onto the stack and count the total number of the defective pixels;

Step 2: Set the number of the correctly classified defective pixels equal to 0;

Step 3: Set the number of popped defective pixels equal to 1;

Step 4: Pop one of the defective pixels out of the stack;

Step 5: Check the local neighbourhood of the defective pixel with a 9 by 9 window size. If there is no neighbourhood belonging to one of pixels of the mask contours, set the class of the pixel as non-defective and continue. Else, increase the number of correctly classified defective pixels by 1 and continue;

Step 6: Update the number of popped defective pixels; and

Step 7: If the number of defective pixels is less than the total number of the defective pixels, go to Step 4. Else, stop.

Since the mask is accurately matched to Surface #1 by applying Algorithm 4.2, the pixels along the edges of Surface #1 are very close to the corresponding pixels on the contours of the mask. To determine whether a pixel in the segmented image produced by Algorithm 5.1 is an edge pixel of Surface #1, we need to calculate the closest distance between the pixel and the pixels of the mask contours. However, the computational cost to calculate this distance is very large and impractical. We use a practical method to estimate the closest distance between a pixel and the pixels of the mask contours.

It is very convenient to process pixels in a rectangular or square window in an image. In particular, the size of the window can be used to indicate the displacement between two pixels. For these reasons, we apply a window with a predefined size to every pixel to estimate the closest distance between it and the contours of the mask in Step 5. We determined the appropriate size of the window manually using the 23 part images available during software development. The window should be sized to remove wrongly classified pixels at the edges as much as possible. No window size worked correctly with all 23 images. A size of 9 by 9 pixels^2 was found to be the most effective, working with 22 of the 23 images. The experimental results before and after applying Algorithm 5.2 are shown in Figures 5.14 and 5.15. Pixels classified as defective are shown in red and non-defective pixels are shown in green in these and subsequent figures of experimental results.

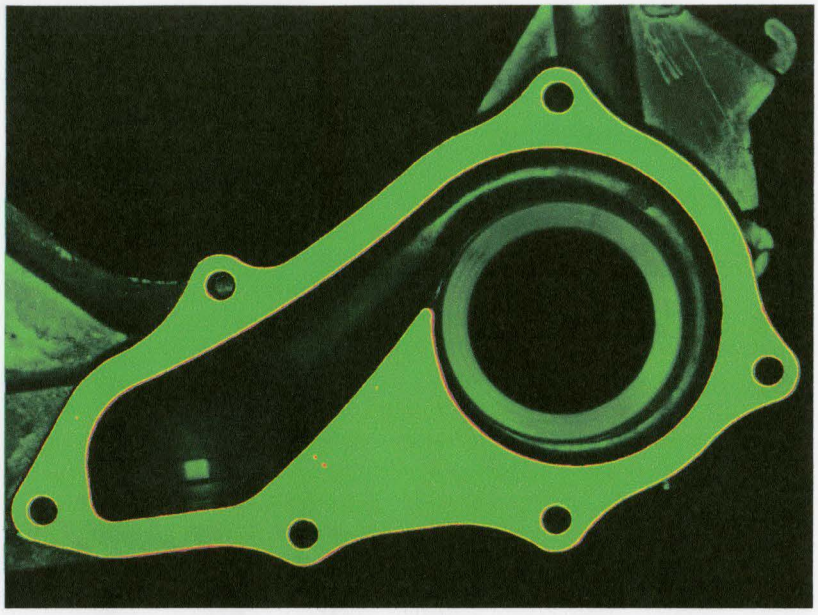


Figure 5.14 Segmented image with wrongly classified pixels.

Figure 5.15 Segmented image after applying Algorithm 5.2.

In Figure 5.14, not only the pixels belonging to the defects but also the pixels outside the edges of Surface #1 have been classified as defective pixels. After applying Algorithm 5.2, only the pixels belonging to the defects remain, as shown in Figure 5.15.

5.4 Defect Size Measurement

5.4.1 Algorithm Overview

For the automotive water pump housing part surface inspection problem, the classification of the parts is according to the size and location of the potential defects on Surface #1. By using Algorithm 5.1, we can detect the defective pixels on Surface #1. However, very small surface imperfections, e.g. appearing as an isolated defective pixels, are not considered defects according to the manufacturer's quality control specifications. Therefore, we need to measure the size of the defects.

First of all, we need to cluster the defective pixels belonging to individual defects. After clustering the defective pixels, we need to calculate the size of the defects. In industrial applications the size of the defect is measured in mm. But in image processing the object is measured in pixels. So the transformation between the real world coordinate system and the image coordinate system must be determined. In Chapter 3, we obtained the transformation matrices between the image coordinate system and the real world coordinate system. By using the transformation matrix $\mathbf{H}_2$, we can obtain the coordinates of the defective pixels in mm. Then we can figure out the size of the potential defects in mm.

5.4.2 Assumptions

To determine the size of the potential defects on Surface #1, we assume that:

1. The pixels depicting the potential defects on Surface #1 are known;
2. The maximum gap among the pixels depicting the same potential defect is known;
3. The transformation matrix from the image coordinate system to the real world coordinate system is known.

The first assumption is satisfied by applying Algorithms 5.1 and 5.2. To determine the size of the potential defects on Surface #1, we need to cluster the defective pixels into distinct clusters describing individual defects. Since the pixels belonging to

one defect are very close to each other and their binary values are different from the binary values of the non-defective pixels, we can use a type of seeded region growing to cluster the defective pixels. Ideally, the defective pixels belonging to one defect should be connected. Unfortunately, there often exist gaps among the defective pixels describing the same defect. It is not sufficient to use only the 8-connected neighbours in the seeded region growing algorithm. We need to apply a larger neighbourhood window to cluster the pixels. The size of the neighbourhood window should be determined according to the maximum gap among the pixels depicting the same potential defect. A larger window is the more conservative choice since it will cause more pixels to be clustered in the same defect and a large defect will result in the part being classified as defective. We chose a 9 by 9 pixel window which corresponds a maximum gap of about 0.8 mm between defective pixels. The third assumption was satisfied using the camera calibration technique described in Chapter 3.

5.4.3 Algorithms

The algorithm for clustering the defective pixels belonging to individual defects is presented below.

Algorithm 5.3 Clustering of Defective Pixels

Step 1: Set all the cluster numbers of the defective pixels equal to 0 and push all the defective pixels into the stack;

Step 2: Pop the first defective pixel out of the stack and set the cluster number of the pixel equal to 1;

Step 3: Check the neighbour pixels in the 9×9 window. If there is no defective pixel in the window, continue. Else, set the cluster numbers of the defective pixels with a cluster number of zero equal to the cluster number of the popped defective pixel. Check the neighbour pixels of the newly numbered defective pixels until all defective pixels in the neighbourhood windows have been numbered;

Step 4: If there are no more defective pixels in the stack then stop. Else, pop a

defective pixel out of the stack;

Step 5: If the cluster number of the popped defect pixel is not equal to zero, go to step 4. Else, increase the cluster number by 1, and go to step 3;

Using Algorithm 5.3, we can find the potential defects on Surface #1. The clustering result for the ROI shown in Figure 5.1 is shown in Figure 5.16. In this figure, the red numbers indicate the labels of the four detected defects. The pixels belonging to the same defect are surrounded by a red rectangle.

Figure 5.16 Clustering result obtained using Algorithm 5.3.

After the individual defects are found we can calculate the size of each one. The size of the potential defects is calculated using the algorithm given below.

Algorithm 5.4 Defect Size Measurement

Step 1: Apply Algorithm 5.3 to cluster the defective pixels and get the total number of potential defects;

Step 2: Set the counter equal to 1;

Step 3: Apply transformation matrix $\mathbf{H}_2$ to the defective pixels whose cluster number equals the counter. Save these transformed pixels for use in Algorithm 5.5;

Step 4: Calculate the maximum distance in mm between any pair of points from the set produced by Step 3. Save this as the size of the potential defect;

Step 5: Increase the counter by 1;

Step 6: If the counter is smaller than the total number of the clusters of defective pixels, go to Step 3. Else stop.

By applying the transformation matrix from the image coordinate system to the real world coordinate system in Step 3, we can convert the displacement in pixels into mm. The transformation matrix is named $\mathbf{H}_2$ in Chapter 3. Given a pixel in an image, the coordinates of the pixel in the real world coordinate system are calculated as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{H}_2 \times \begin{bmatrix} j \\ i \\ 1 \end{bmatrix} \tag{5.9}$$

where the pixel is at the ith row and the jth column of the image, and (x, y) is the coordinates of the pixel in the real world coordinate system.

We calculate the size of every potential defect on Surface #1 in Step 4. First, we need to create a definition of the size of a defect. Since the potential defects are irregular in shape, we define the maximum distance between any pair of pixels belonging to the same potential defect as the size of the defect. We apply the Euclidean distance as the distance between two pixels. The Euclidean distance is calculated as:

$$d_E = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{5.10}$$

where d_E is the Euclidean distance between two points in a plane, (x_1, y_1), (x_2, y_2) are the coordinates of the two points in mm, respectively. The size of the defect is then set equal to maximum of the d_E values.

Potential defects larger than a specific size will be classified as defects and the remainder will be ignored. The sizing results for the defects shown in Figure 5.16 are presented in Figure 5.17.

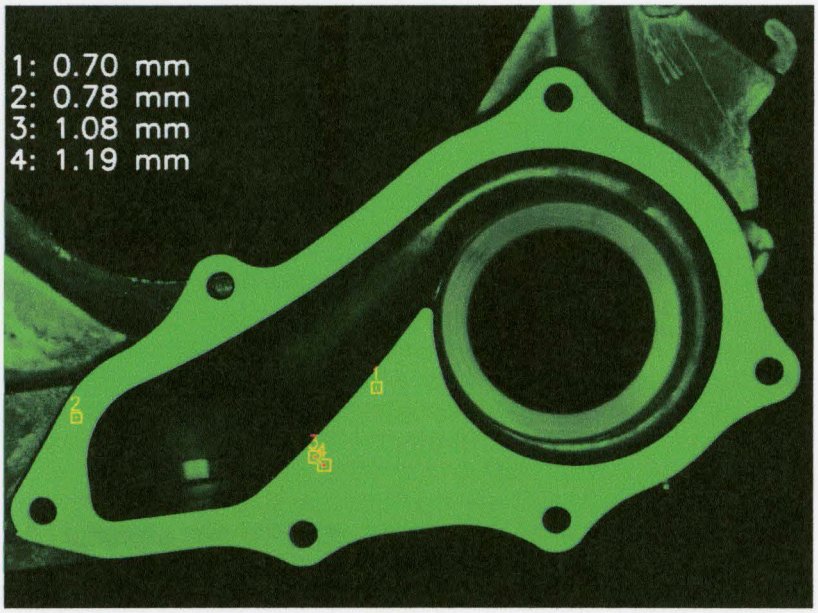


Figure 5.17 Result with measured sizes of the potential defects.

5.5 Sealing Area Identification and Defect Classification

For the automotive water pump housing surface inspection problem, Surface #1 is categorized into two areas: the sealing area (also known as the critical area) and the non-critical area. According to the manufacturer's quality control specifications, the definitions of defects in the sealing area and the non-critical area are different. So the sealing area on Surface #1 must be identified in each image before the potential defects can be classified as actual defects or non-defects.

In this section, an algorithm to identify the sealing area and classify the defects of Surface #1 is developed. After describing the basic idea of the algorithm, a computationally efficient version is described.

5.5.1 Algorithm Overview

According to the manufacturer of the parts, if the distance between a defect and the inner edge of Surface #1 is smaller than 5mm, the minimum size of a defect is 0.5mm. Secondly, if the distance between the defect and the inner edge of Surface #1 is larger than 5mm, the minimum size of a defect is 1mm. The inner edge of Surface #1 is defined as a sealing edge, which is shown in red in Figure 5.18. The sealing area of Surface #1 is defined as follows:

> Assuming A is a point on Surface #1, if A satisfies the condition that the distance between A and the sealing edge is less than or equal to 5 mm, then A is a point in the sealing area of Surface #1.

Figure 5.18 Sealing edge shown in red.

As mentioned above, a potential defect is represented by a cluster of defective pixels in an image. We define the distance between a potential defect and the sealing edge of Surface #1 in an image as the minimum of the distances between every pixel belonging to the potential defect and every pixel describing the sealing edge. As we can see in Figure 5.18, the sealing edge is not a regular shape. It is impossible to describe the minimum distance between a pixel to the sealing edge using a simple equation. We need to find another way to define the distance.

An obvious approach would be to first calculate the distance between every pixel belonging to the potential defect and every pixel of the sealing edge. Then pick the minimum distance as the distance between the potential defect and the sealing edge. However, the computational cost to calculate the distance by applying this approach is very high. Timing results will be given in Section 5.6.1. To reduce the running time of the program, it was necessary for us to develop a more computationally efficient algorithm.

5.5.2 Assumptions

To identify the sealing area and classify the defects of Surface #1, the following assumptions must hold:

1. The size of the potential defect is known;
2. The points of the sealing edge are known;
3. The transformation matrix of the camera parameters, $\mathbf{H}_2$, is known;
4. The relevant quality control specifications are known.

The first and second assumptions were satisfied by applying Algorithm 5.4 and the left-right search algorithm, respectively. The third assumption was satisfied using the camera calibration technique from Chapter 3. The last assumption was satisfied by consulting the manufacturer of the parts.

5.5.3 Algorithm

We developed an efficient algorithm to identify the sealing area and classify the defects of Surface #1 of the automotive water pump housing part. The algorithm is presented below.

Algorithm 5.5 Sealing Area Identification and Defect Classification

Step 1: Set the counter equal to 1;

Step 2: Load the transformed pixels and defect size corresponding to the counter;

Step 3: If the size of the potential defect is less than 0.5mm go to Step 7; else continue;

Step 4: For each pixel belonging to the potential defect, check the 72-pixel diameter circle of pixels around it. If there are pixel(s) depicting the sealing edge then the potential defect is classified as a defect in the sealing area; else, the potential defect is in the non-critical area and go to Step 6;

Step 5: Go to Step 7;

Step 6: If the size of the potential defect is greater than or equal to 1mm then classify it as a defect, else classify it as non-defect; and

Step 7: If the counter is smaller than the total number of potential defects then increase the counter by 1 and go to Step 2, else stop.

As previously mentioned, we must know whether a potential defect is inside the sealing area or not before we can classify it. It is inside the sealing area if the minimum distance between it and the sealing edge is less than or equal to 5mm. The challenge is to efficiently calculate the minimum distance for each pixel belonging to a potential defect. The efficient and accurate method we propose is to check the 10mm diameter circle of pixels centred at each defective pixel for intersections with the sealing edge. It is necessary to convert the diameter to pixels, using the $\mathbf{H}_1$ matrix calculated in Chapter 3, as follows:

$$D_{image} = \left\| \mathbf{H}_1 \times \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix} \right\| \approx 72 \text{ pixels} \tag{5.11}$$

where D_{image} is the diameter in pixels. The corresponding circle contains 236 pixels. Only these pixel locations need to be checked in the image of the sealing edge to determine if a pixel belonging to a potential defect is inside the sealing area. The experimental result for the ROI shown in Figure 5.1 is shown in Figure 5.19. The characters inside the brackets indicate whether the defect locates in the sealing area (critical area). The character "C" means the possible defect is in the critical area while the character "N" means it is out of the sealing area. Note that since all of the potential defects are inside the critical area and are larger than 0.5mm they were all classified as defects. A part with one or more defects is classified as bad or defective.

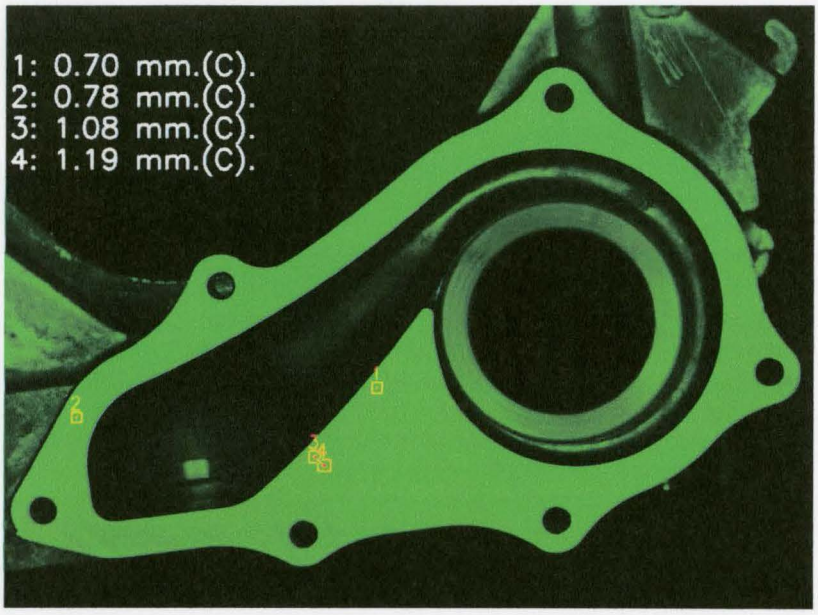


Figure 5.19 Sealing area identification and defect classification result

5.6 Experimental Results

5.6.1 Timing Results

The software designed for the automotive water pump housing part surface inspection problem was developed in C under Visual Studio 6.0. The software was tested on a computer with the processor of AMD Athlon (TM) xp 1800 +, 1.5 GHz and 512 Mb RAM. The detailed timing results are listed in Table 5.1.

Table 5.1 Timing Results

Algorithm	Running Time
Generate mask using Algorithm 4.1	62ms
Generate mask using OpenCV library	16ms
Search for the six holes in the entire image	18656ms
Search for the six holes using method in Algorithm 4.2	140ms
Algorithm 4.2: Accurate image registration	188ms
Algorithm 4.3: Simplified image registration	55ms
Algorithm 5.1: Local standard deviation followed by adaptive global thresholding	125ms
Find contours using left-right search technique	Less than 1ms
Algorithm 5.2: Removal of Wrongly Classified Edge Pixels (Average time for the 23 parts available)	19.22ms
Algorithm 5.3: Clustering of Defective Pixels (Average time for the 23 parts available)	17.96ms
Algorithm 5.4: Defect Size Measurement (Average time for the 23 parts available)	48.96ms
Algorithm 5.5: Sealing area identification and defect classification	Less than 1ms
Sealing area identification by checking distance between every defective pixel and every sealing edge pixel	2578ms
Average inspection time[1] (for the 23 parts available)	688.23ms

[1] Not including the image acquisition time.

From Table 5.1, we can have several observations. First, it took a longer time to generate the mask using Algorithm 4.1 than using the function in OpenCV library. Since we only generate one mask for the entire surface inspection procedure, we concern more about the quality of the mask. So the running time is not critical in the mask generation process. Second, the search speed using the method in Algorithm 4.2 is much faster than searching the entire image. This makes the proposed method a computationally efficient one. Third, it is very easy to conclude that the simplified image registration algorithm runs three times faster than the accurate image registration algorithm. Fourth, the running time for the method in Algorithm 5.5 is less than 1ms, which is fast enough for the manufacturing application. Finally, by applying the software to the 23 automotive water pump housing part images available during the software development, the average inspection time (not including the image acquisition time) is less than 1 second, making it fast enough for use in the automotive manufacturing industry.

5.6.2 Inspection Results

Totally 1,713 images were obtained and inspected using the adaptive machine vision system developed in this thesis. The definition and source of the measured quantities are presented in Table 5.2. The inspection results for the 1,713 images are listed in Table 5.3.

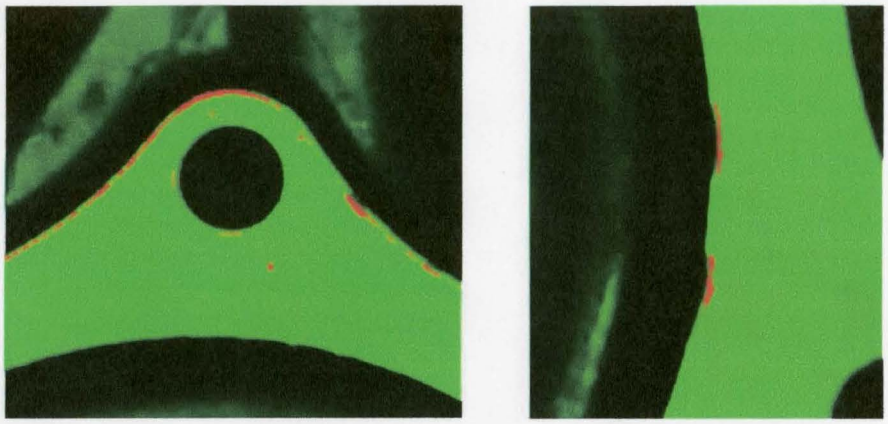
Table 5.2 Definition and source of measured quantities listed in Table 5.3

Parameter	Parameter Definition	Source of Parameter
N_{Total}	Total number of inspected parts	Manually counted
N_G	Number of good parts	Manually inspected
N_D	Number of defective parts	Manually inspected
N_P	Number of parts with pores	Manually inspected
N_{PD}	Number of parts with detected pores	Adaptive machine vision system and manual judgement
N_{PM}	Number of parts with missing pores	Adaptive machine vision system and manual judgement
N_O	Number of parts with other defects	Manually inspected
N_{OD}	Number of parts with detected other defects	Adaptive machine vision system and manual judgement
N_{OM}	Number of parts with missing other defects	Adaptive machine vision system and manual judgement
N_T	Truely classified part number	Adaptive machine vision system
N_F	Falsely classified part number	Adaptive machine vision system
N_{FR}	Falsely rejected part number	Adaptive machine vision system
N_{FA}	Falsely accepted part number	Adaptive machine vision system
N_{TM}	Number of truely classified parts with missing defects	Adaptive machine vision system and manual judgement

Table 5.3 Inspection results

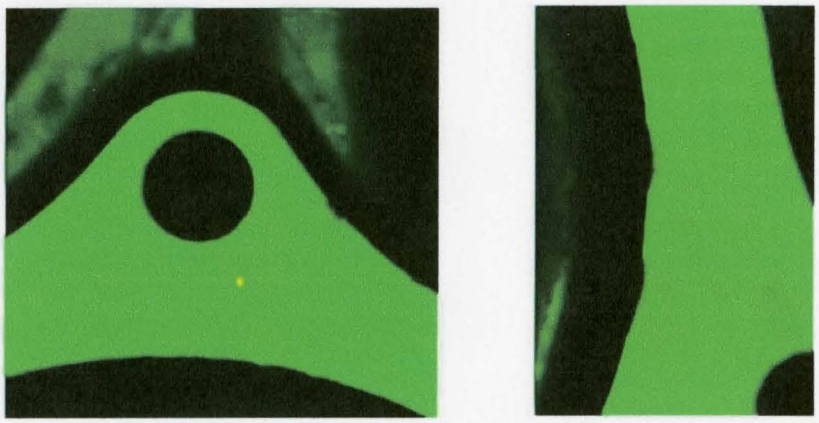
Category	Number	Percentage
Total Number	$N_{Total} = 1713$	100%
Good	$N_G = 653$	$N_G/N_{Total} = 38.12\%$
Defective	$N_D = 1060$	$N_D/N_{Total} = 61.88\%$
Defective (Pores)	$N_P = 806$	$N_P/N_D = 76.04\%$
Pores Detected	$N_{PD} = 749$	$N_{PD}/N_P = 92.93\%$
Pore Missed	$N_{PM} = 57$	$N_{PM}/N_P = 7.07\%$
Defective (Other)	$N_O = 254$	$N_O/N_D = 23.96\%$
Other Detected	$N_{OD} = 105$	$N_{OD}/N_O = 41.34\%$
Other Missed	$N_{OM} = 149$	$N_{OM}/N_O = 58.66\%$
True Classification	$N_T = 1447$	$N_T/N_{Total} = 84.47\%$
False Classification	$N_F = 266$	$N_F/N_{Total} = 15.53\%$
False Rejection	$N_{FR} = 97$	$N_{FR}/N_{Total} = 5.55\%$
False Acception	$N_{FA} = 169$	$N_{FA}/N_{Total} = 9.87\%$
True Classification with Missed Defects	$N_{TM} = 37$	$N_{TM} / N_{Total} = 2.16\%$

In this surface inspection problem, the majority of the defects were pores. From Table 5.3, the pores were correctly classified in 92.93% of the images. There were some wrong inspection results. Figures 5.20 to 5.25 show several difficult cases that sometimes generated wrong results.



(A) (B)

Figure 5.20 Inspection results for defects close to the edge of Surface #1. (A) Edge pixels are wrongly classified as defective (Shown in red). (B) Defects close to edge are correctly detected (Shown in red).



(A) (B)

Figure 5.21 Inspection results for the same automotive parts shown in Figure 5.20. (A) Edge effect is removed. (B) Defects close to edge are missed.

One difficult case is to detect defects close to the boundary contour. Figure 5.20 shows two examples. In Figure 5.20 (B), the defects close to the sealing edge are correctly detected. However, by applying the same parameters, the edge pixels are

wrongly classified as defective in Figure 5.20 (A). Such wrong detection is caused by different sizes of Surface #1s for the two parts in these images. Recall that Surface #1 is roughly 180 mm by 110 mm, while the defects can be less than 1mm in diameter. The difference between the large inspection area and the small defect size makes it difficult to correctly classify all the automotive part images.

By adjusting the window size used in Algorithm 5.2 from 9 by 9 to 13 by 13, we can remove the edge effect and obtain correct result for the case shown in Figure 5.20 (A). The corresponding result is shown in Figure 5.21 (A). However, under the same parameters, the detected defects in Figure 5.20 (B) are missed, which is shown in Figure 5.21 (B). Actually, the edge effect is the main reason to cause the wrong inspection for the other defects in this inspection problem. There are two possible ways to solve this problem. One way is to apply more complicated motion estimation models and develop more complicated image registration algorithms to make the mask match up with every Surface #1 more accurately. Another way is to adjust some software parameters adaptively with respect to the inspected surface. Both of these methods can be developed as future work.

Machining marks are also a challenging problem for the inspection problem. For some automotive water pump housing parts, the machining marks on the well-machined surface are very similar in appearance to the defects and are very hard to distinguish. Figure 5.22 shows a Surface #1 with obvious machining marks. In this figure, the machining marks are wrongly detected as defects. After increasing the weight used in Algorithm 5.1 from 0.77 to 3.43, the machining marks are removed in the inspection result shown in Figure 5.23. Unfortunately, some defects on Surface #1 are missed by applying the same modified parameter. One possible way to solve this problem in future would be to apply texture-based inspection techniques, which are much more complicated and slower to compute.

The final challenging problem is to detect the very subtle defects, such as subtle dents or scratches, which are difficult to detect even for human beings. Figure 5.24 shows one example. Although we partially detected the subtle dent on the Surface #1, we also

wrongly classified the machining marks as defective. Figure 5.25 shows the result for the same part but increasing the weight used in Algorithm 5.1 from 0.77 to 2.14. Unfortunately, the subtle dent and some porosity were not detected. Since we only use one image of the automotive part to inspect the defects, we cannot change the viewing angle like a human being does. Using a multiple-image based method with adjustable lighting may improve the inspection results in future.

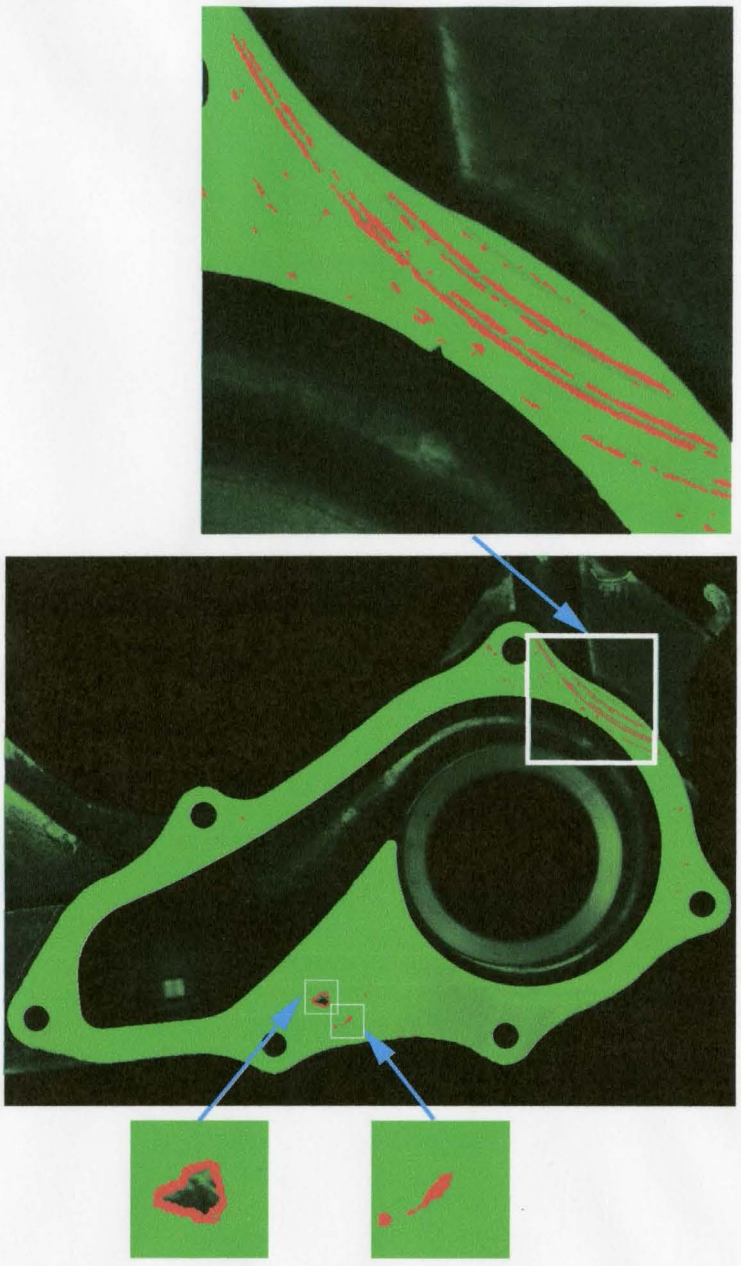


Figure 5.22 Result where obvious machining marks are wrongly detected as defects. The obvious machining marks are shown in red in the top close-up sub-image. The left close-up sub-image at the bottom shows a detected defect (in red). The bottom right one shows a detected defect that is similar in appearance to the machining marks (in red).

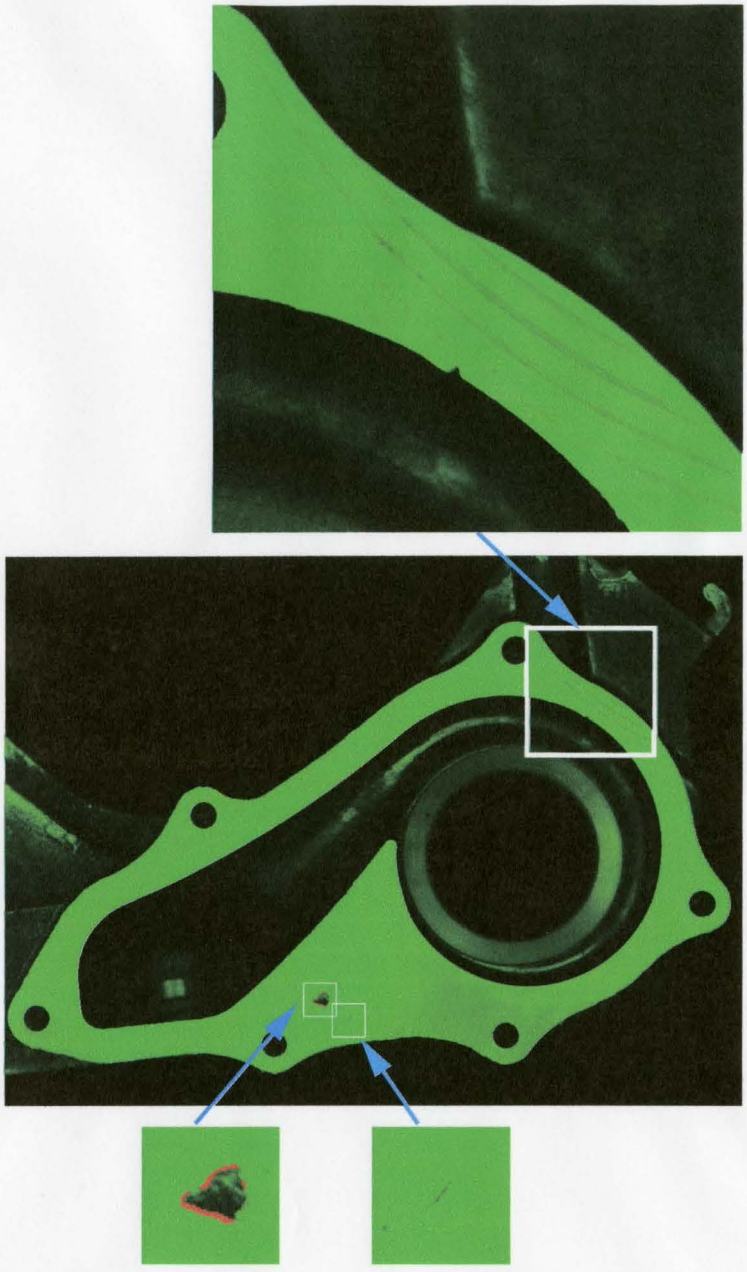


Figure 5.23 Result where obvious machining marks are ignored. The top close-up sub-image shows the ignored machining marks (no red pixel). The left sub-image at the bottom shows the detected defect (in red). The bottom right one shows the detected defect that is similar in appearance to the machining marks (no red pixel).

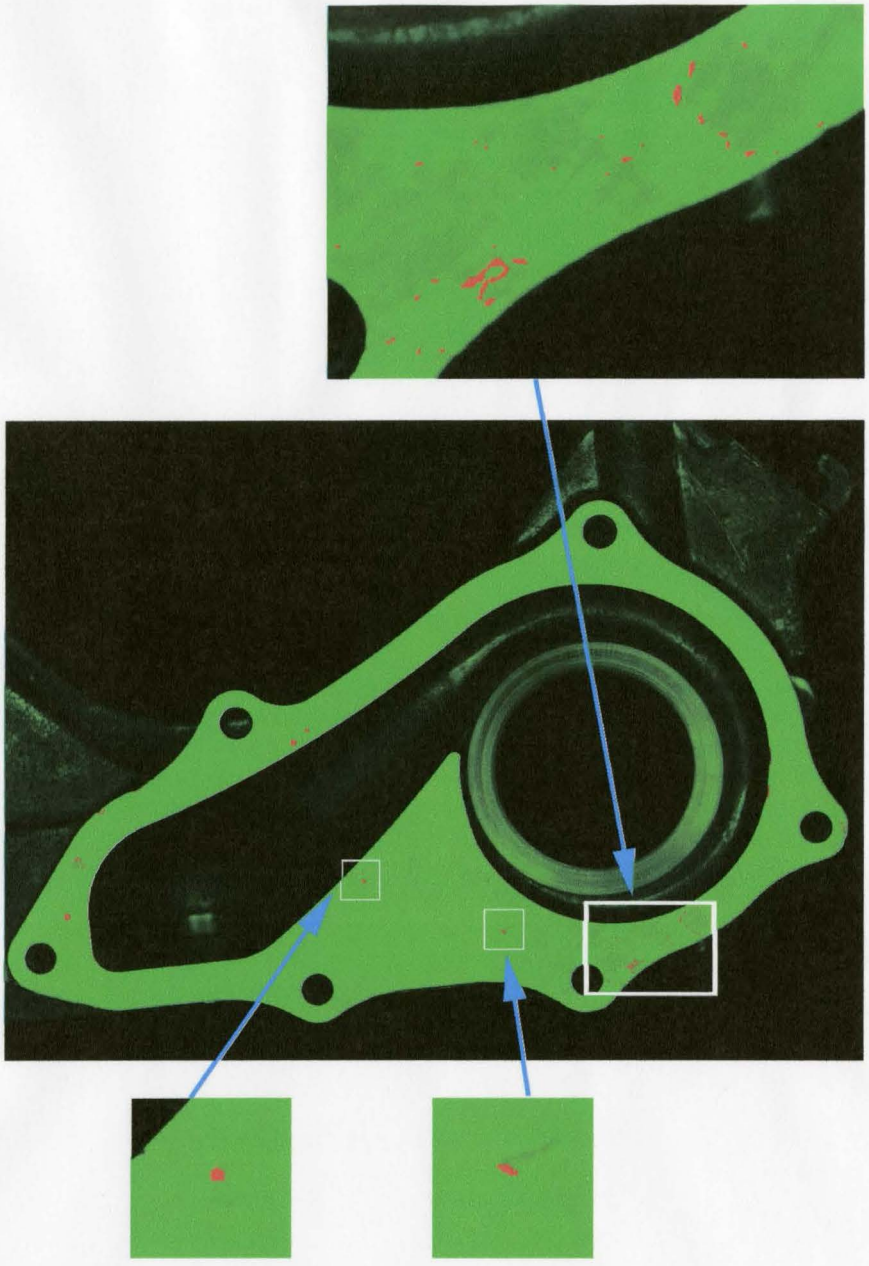


Figure 5.24 Result where subtle dent is partially detected. The top zoomed in sub-image shows the wrongly detected machining marks (in red). The left sub-image at the bottom shows the detected porosity (in red). The bottom right one shows the partially detected subtle dent (in red).

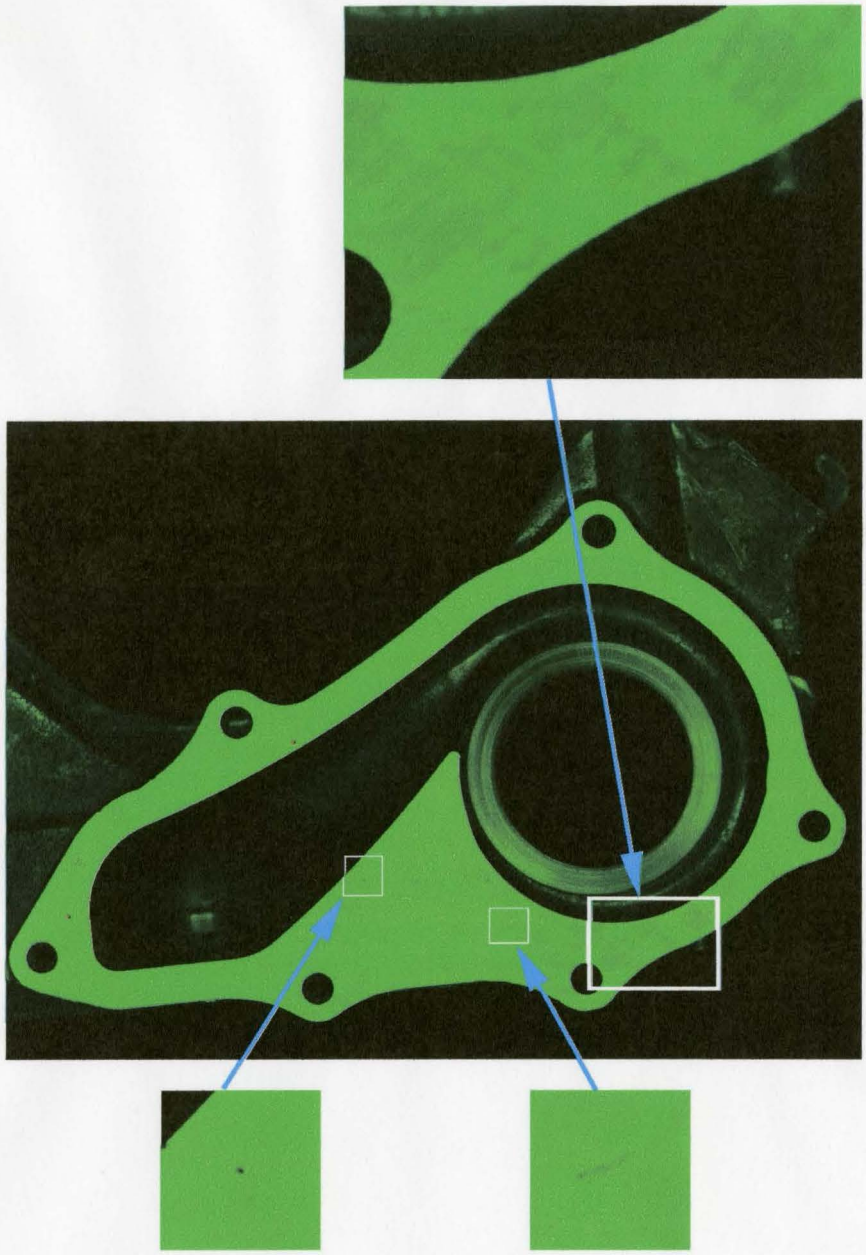


Figure 5.25 Result where subtle dent and pores are missed. The top zoomed in sub-image shows the ignored machining marks (no red pixel). The left sub-image at the bottom shows the missed porosity (no red pixel). The bottom right one shows the missed subtle dent (no red pixel).

5.7 Summary

In this chapter, we presented a local standard deviation followed by adaptive global thresholding algorithm to detect the defective pixels on Surface #1. An algorithm using the contours of the mask was proposed to remove wrongly classified pixels outside the edge of Surface #1. We also presented the algorithm to measure the size of the defects. Then the algorithm to identify the sealing area and classify the automotive parts was discussed. We tested the adaptive machine vision system described in chapters 3 to 5 on 1,713 images and obtained an accuracy of 93% in the porosity classification.

Chapter 6

CMAC Neural Network Based Machine Vision System

6.1 Introduction

A neural network is well known as a powerful tool for machine vision. During the past several decades, more and more neural networks were used in machine vision systems to solve the complexity and nonlinearity of the vision problems. Based on our review of the literature, the CMAC neural network was selected for this thesis. The CMAC neural network is based on the table look-up technique, which makes it fast enough to use in a real-time machine vision system for inspecting manufactured parts. In this chapter, a CMAC based machine vision system is designed for an automotive beam clip present/absent inspection problem.

We organize the chapter as follows. First, the clip present/abscent problem is described briefly. Second, the basic idea of the CMAC neural networks is introduced. After that, the machine vision system based on CMAC neural network is discussed. The chapter closes with a discussion of the experimental results, and conclusions.

In this context, "real-time" means that the inspection of a part should require less time than it takes to manufacture that part in order to maintain the production rate. Since the typical production rate is roughly one part per minute and each part can include several inspection problems, the time to perform an inspection problem should be less than 10 seconds.

6.2 Problem Description

In Chapters 3-5, we described the development of an adaptive machine vision system for the automotive water pump housing part surface inspection problem. In this surface inspection problem, the rules to categorize the defects of the automotive parts were well defined. The computer vision algorithms based on the pixel-by-pixel image processing were successfully applied to this problem. But if the inspection problems

possess strong nonlinearity and complexity, or the categorization rules are difficult to obtain manually, the computer vision algorithms based on pixel analysis are unsuitable.

Neural networks are powerful tools for this class of inspection problems. The automotive beam clip present/absent problem is a problem that belongs to this class. The clip present/absent problem is illustrated in figures 6.1 and 6.2.

In the automotive beam clip present/absent problem, we need to determine whether there is a clip present with the correct position and orientation. One hundred and forty five images of parts with the clip present and another 55 images of parts with the clip absent were provided by the manufacturer, Van-Rob. The position and the orientation of the clips change in different images. Furthermore, the background of the images is complex, changing and is not always distinct from the appearance of the clips. So there exists strong nonlinearity and complexity in this machine vision problem. As a result, we developed a neural network based machine vision algorithm to solve it.

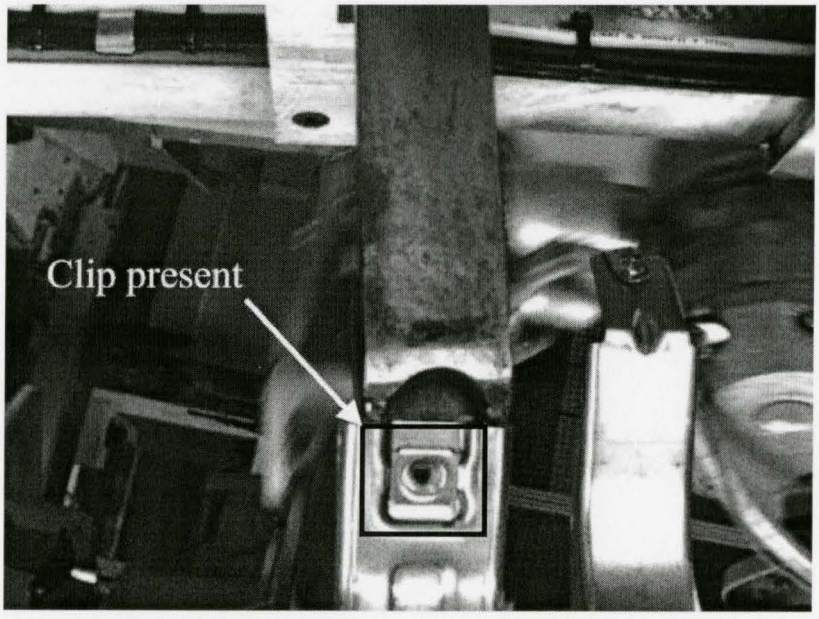


Figure 6.1 Automotive beam with clip present.

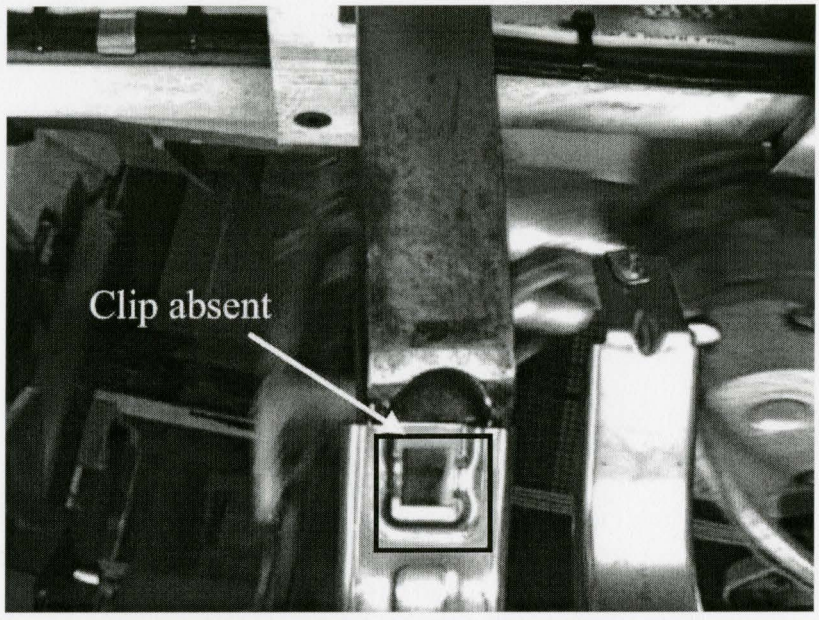


Figure 6.2 Automotive beam with clip absent.

For an industrial inspection application, the computer vision algorithm should be suitable for the real-time implementation. CMAC is a neural network based on the table look-up technique for representing complex, nonlinear functions. As a result, a CMAC neural network is capable of learning and executing with the required speed.

A manually defined ROI containing the location of the clip will be used to eliminate most of the unimportant areas of the image from further processing. This will have the benefits of making the inspection system perform faster and more reliably. After being preprocessed, the pixels inside the ROIs are input into the CMAC neural network. In the automotive beam clip present/absent problem, the CMAC neural network works as a classifier. The output of the network will describe whether there is a clip present in the image or not.

6.3 CMAC Neural Network

6.3.1 Basic CMAC Module

Basically, CMAC uses a table look-up technique for representing a complex, nonlinear function. It also employs the concept of sensory encoding using local receptive

fields, which imitates some functions of the human brain. The basic computational scheme of a CMAC neural network is shown in Figure 6.3.

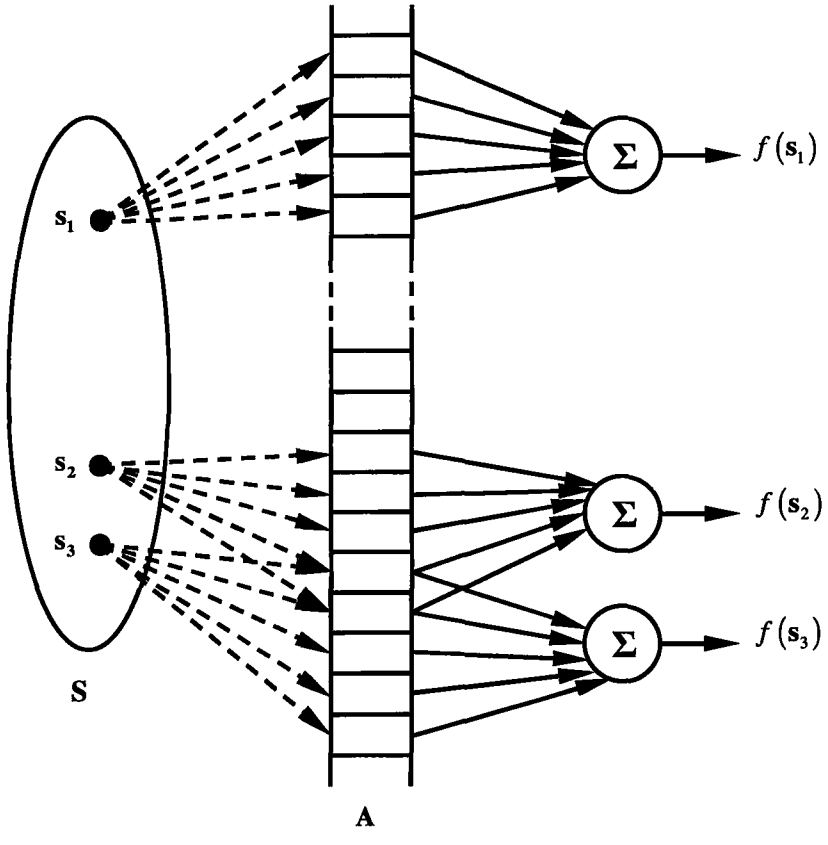


Figure 6.3 Diagram of the basic CMAC mapping. The mapping of three state vectors is shown [52].

As shown in the figure, each point in the input state space *S* maps to a certain number of locations in the N-dimensional memory *A*. Each point is a state vector. The value of the complex nonlinear function is then determined by the average of the addressed weights [54]:

$$f(\mathbf{s}) = \frac{1}{C}\sum_{i=1}^{C} W(A_i) \tag{6.1}$$

where $f(\bullet)$ represents the complex nonlinear function, $\mathbf{s}$ is the state vector in the input state space, C is the number of memory addresses corresponding to $\mathbf{s}$, A_i is the ith memory address, and $W(A_i)$ is the weight stored at address A_i.

Define D as the summed absolute difference between the components of two input state vectors:

$$D = \left\| \mathbf{s}_i - \mathbf{s}_j \right\|, (\mathbf{s}_i, \mathbf{s}_j \in S) \tag{6.2}$$

where $\| \bullet \|$ is the L^2-norm of the vector. If D is greater than C, there will be no overlap between the two corresponding sets of memory locations in A. Otherwise, the number of shared memory locations in A will be approximately $D - C$. As a result of the shared memory locations, the output of the network will exhibit generalization between points close to each other in the input state space S, and no generalization between distant points. The extent of generalization within the state space is determined by the parameter C. Thus, it is easily concluded from the prior discussions and Figure 6.3 that the CMAC neural network is a non-fully-connected neural network, which makes it different from a typical neural network. The unique structure makes the CMAC converge very fast when learning.

After training, the online implementation of the basic CMAC proceeds as follows [54]:

1. Identify the C memory addresses excited by the input state;
2. Load the C adjustable weights for those addresses from the pool of stored weights;
3. Compute the average of the C adjustable weights.

6.3.2 Complete CMAC Module

In last section, we discussed the mapping from the input state space S to the CMAC memory A. The mapping will result in the memory A that contains on the order of R to the power N memory locations, where each of the N variables in S can take on any one of R discrete values. It is obvious that the size of memory A can be unreasonably large for practical values of N and R. For example if S contains 100 8-bit integers then the

size of A will be 256^{100}. As a result, the management of the memory becomes an important problem for a CMAC neural network. In fact, while the theoretical size of the input state space S and the resulting memory A can be very large, the number of different states that are encountered in solving a practical problem is typically much smaller. So the number of memory locations that will actually be addressed is much smaller.

Based on this, it is reasonable to perform a uniform, random mapping of the large memory A into a smaller memory A'. The idea of this mapping is shown in Figure 6.4. As a result of the random mapping from the larger memory A to the smaller memory A', distant input states have a finite probability of sharing one or more location in A' (i.e. mapping collisions), producing an undesirable generalization between distant input states. Most implementations of CMAC neural networks use some form of pseudo-random hashing to transform the given state vector to a scalar address of the corresponding weight storage.

At the beginning of the learning procedure, the memory A' is set equal to zero. The locations in the memory A' are used randomly and labelled. If all the locations are used then mapping collisions will occur that will degrade the classification performance. Pseudo-random hashing is used to make it the fewest mapping collision happen. Even with hashing, using more memory will reduce the occurrence of collisions. As a result, enlarging the memory size will typically improve the classification performance.

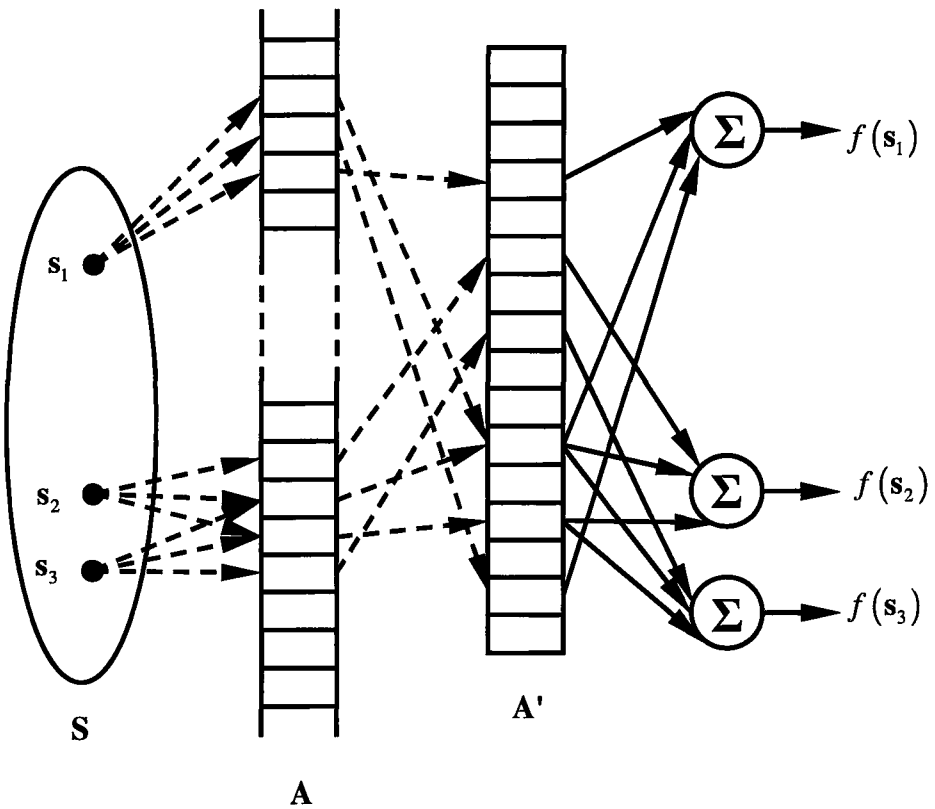


Figure 6.4 Diagram of the complete CMAC mapping [52].

6.3.3 Adjustment of the Weights

The weights stored in the CMAC neural network are adjusted by using the error correction strategy when the network is training. For one state vector **s** in input state space *S*, the output of the CMAC $f(\mathbf{s})$ can be computed using (6.1). The adjustment of the weights in the network is dependent on the difference between the average of the *C* adjustable weights and the desired response $f_d(\mathbf{s})$ for the state vector:

$$\Delta W = \beta \times \left[f_d(\mathbf{s}) - f(\mathbf{s}) \right] \tag{6.3}$$

where β is a constant training gain in the range of [0,1]. If β is 1.0, the weights are adjusted to force the output of the network to be exactly the same as the desired response. If β equals 0.5, the CMAC output is adjusted to fall halfway between the old output

value and the desired response. If β is zero, the weights will not be changed. Note that the same value ΔW is added to each of the C memory locations accessed in the computation of $f(\mathbf{s})$ [77].

The output of the CMAC neural network is the average over C adjustable weights. The adjustment of the weights is calculated to reduce the errors between the real network output and the desired network response. If we add the adjustment calculated by using (6.3), we cannot control the individual contribution for each of the C weights. As a result, some adjustable weights could reach an unreasonable value that will degrade the network performance. To fix this problem, a penalty factor is introduced to the adjustment of the network weights. The equation (6.3) is then replaced by

$$\Delta W_i = \beta_1 \times \left[f_d(\mathbf{s}) - f(\mathbf{s}) \right] + \beta_2 \times \left[f(\mathbf{s}) - W\left[A_i'\right] \right] \tag{6.4}$$

where the individual weight adjustment value ΔW_i is added to $W\left[A_i'\right]$, the ith weight stored in the memory A' that is accessed during the network training. Therefore, the weight adjustment is different for individual weight. The contribution of the individual weight is calculated as a penalty factor to prevent the weight from reaching unreasonable values [77].

6.4 CMAC Based Machine Vision System

6.4.1 Overview

As previously discussed, in the automotive beam clip present/absent problem, we need to detect whether there is a clip on the automotive beam in the current image. After manually analyzing the 200 images we have, we found that:

1. The lighting condition of the image always changes in different images;
2. The colour of the automotive beam clips is different in different images;
3. The position and orientation of the clips are always changing;
4. Strong environmental disturbances exist in the working environment, such as people walking, assembly devices moving, etc.

So there exists strong complexity in this automotive beam clip present/absent problem. It is very difficult to manually define rules for classifying the images. The CMAC neural network was adopted as a potential solution.

To use the CMAC neural network as the classifier in the neural network based machine vision system, the checked image should be converted to the desired format for the CMAC. So the image should be preprocessed before being input to the network. When an image in the desired format is input to the CMAC neural network, the weights of the network will be adjusted by applying a feedback mechanism. Also, the CMAC weights will be trained by being supervised offline. This is known as supervised training. Specifically, 50 images with clips and 50 images without clips were merged to form as set of 100 images. These images were manually classified. Pseudo-random sequences of these images will be used to train the CMAC. After being well trained, the weights stored in the CMAC will be used to classify the good parts (i.e. parts with clip present) and bad parts (i.e. parts with clip absent) online. Since the output of the CMAC is the summation of the related weights stored in the network, the classification speed should be very fast. A block diagram of the CMAC based machine vision system is shown in Figure 6.5. As shown in the figure, the CMAC based machine vision algorithm includes three main parts: the image preprocessing unit, the CMAC training unit and the CMAC classifier unit. The

image-preprocessing unit converts the original images to the desired input images for CMAC. The CMAC training unit is used to adjust the network weights by applying the feedback mechanism. The training procedure is supervised offline. The structure of the CMAC classifier unit is same as the CMAC training unit. The classifier applies the weights trained in the training unit and is used to categorize the unused part images.

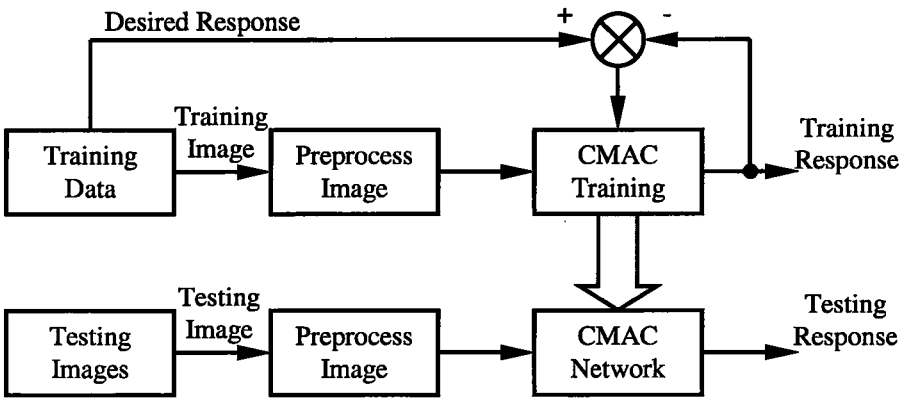


Figure 6.5 Block diagram of the CMAC based machine vision system.

6.4.2 Assumptions

The CMAC based machine vision system is based on these assumptions:

1. The position of the clip in the image does not change over a large range;
2. The size of the clip in the image does not change over a large range;
3. The size of the ROI is predefined;
4. The dimension of the CMAC neural network input fits the available memory and is predefined;
5. The images used as the training data are correctly classified.

A manual analysis confirmed that the first and second assumptions are true for the 200 images. The reason is that the camera's position is fixed and the parts are positioned using a robot.

After the first and the second assumptions are satisfied, we can define the size of the ROI for the images. By adjusting the ROI size manually, we can uniformly apply it to segment all the automotive part images we have.

The fourth assumption can be satisfied by correctly choosing the parameters for the CMAC software (see Section 6.5). The fifth assumption was satisfied by manually classifying the images.

6.4.3 Image Preprocessing

If a large dimension input vector is used with the CMAC then the speed of training will be slow and the number of input vectors required for the training to properly converge will be large. Therefore a low dimension input vector is highly desirable. With computer vision one approach to reduce the dimension would be to replace a large image with a small vector of its features. Unfortunately no consistent features were observed in the images. For this reason, we decided to use a condensed version of the image as the input vector.

As previously mentioned, an ROI containing the location of the clip was manually defined. The ROI is 128 by 128 pixels^2, centred at the expected location for the clip. Keeping only the pixels within the ROI will reduce the data from the $640\times480=307,200$ pixels in the original image to 16,384 pixels. This image format is still too large to be used as the input vector. We decided to use Gaussian pyramid decomposition (GPD) [51] to further reduce the data dimension.

It is well known that neighbouring pixels are highly correlated in an image. Therefore, it is inefficient to represent the image directly in terms of the pixel values since most of the encoded information is redundant. That means it is possible to represent the important information from the given image using a smaller image. By using the GPD algorithm, we should be able to reduce the size of the segmented sub-image while keeping needed to solve the inspection problem.

GPD is a procedure of image down-sampling. We first apply a low-pass filter to the original image and then down-sample it. The resulting image is a condensed version of the original one in both resolution and sample density. The next, more condensed image is generated by repeating the filtering and down-sampling, and so on. The low-pass filtering is performed by a procedure similar to convolution with one of a family of local, symmetric weighting functions. Since the 2-D Gaussian probability distribution is an

important member of this family, the sequence of down sampled images is termed the Gaussian pyramid.

Suppose the original image is represented by the array $\mathbf{I}_0$ with c columns and r rows of pixels. We define this image as the bottom or zero level of the Gaussian pyramid. The condensed version of $\mathbf{I}_0$, $\mathbf{I}_1$ is defined as the level 1 of the pyramid and will have $c/2$ columns and $r/2$ rows. Each down-sampled pixel value of $\mathbf{I}_1$ is calculated as a filtered value of $\mathbf{I}_0$ within a 5 by 5 window. By using the same method, we generate the upper levels of the Gaussian pyramid.

Assume the Gaussian pyramid is composed of N_G levels. We define the level-to-level filtering and down-sampling process as a function R :

$$\mathbf{I}_k = \mathrm{R}(\mathbf{I}_{k-1}), \quad \forall\, 1 < k \le N_G \tag{6.5}$$

For each pixel of $\mathbf{I}_k$, the function R can be described as:

$$\mathbf{I}_k(i,j) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} g(m,n)\mathbf{I}_{k-1}(2i+m, 2j+n) \quad \forall 1 < i \le r2^{-k}, 1 < j \le c2^{-k} \tag{6.6}$$

where $\mathbf{I}_k(i,j)$ is the pixel in the ith row and jth column of the kth level within the pyramid. $g(\bullet)$ is the function of 2-D Gaussian probability distribution, which is defined as [76]:

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.7}$$

where σ is the standard deviation of the distribution. For the automotive beam clip present/absent problem, we will apply a Gaussian filter with the standard deviation value of 0.5. The shape of the Gaussian probability distribution (with $\sigma = 0.5$) is shown in Figure 6.6.

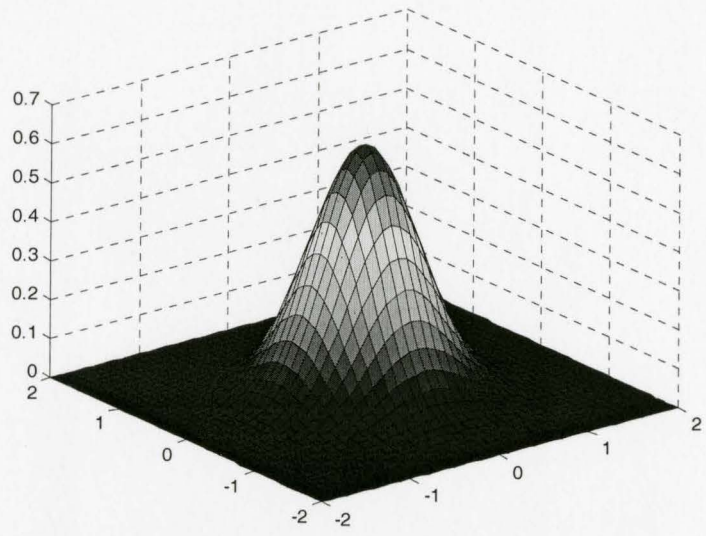


Figure 6.6 The shape of Gaussian distribution used for filtering.

We applied GPD with $N_G = 3$ to reduce the size of each ROI image from 128×128 pixels^2 to 16×16 pixels^2. An example of the four level Gaussian pyramid is given in Figure 6.7.

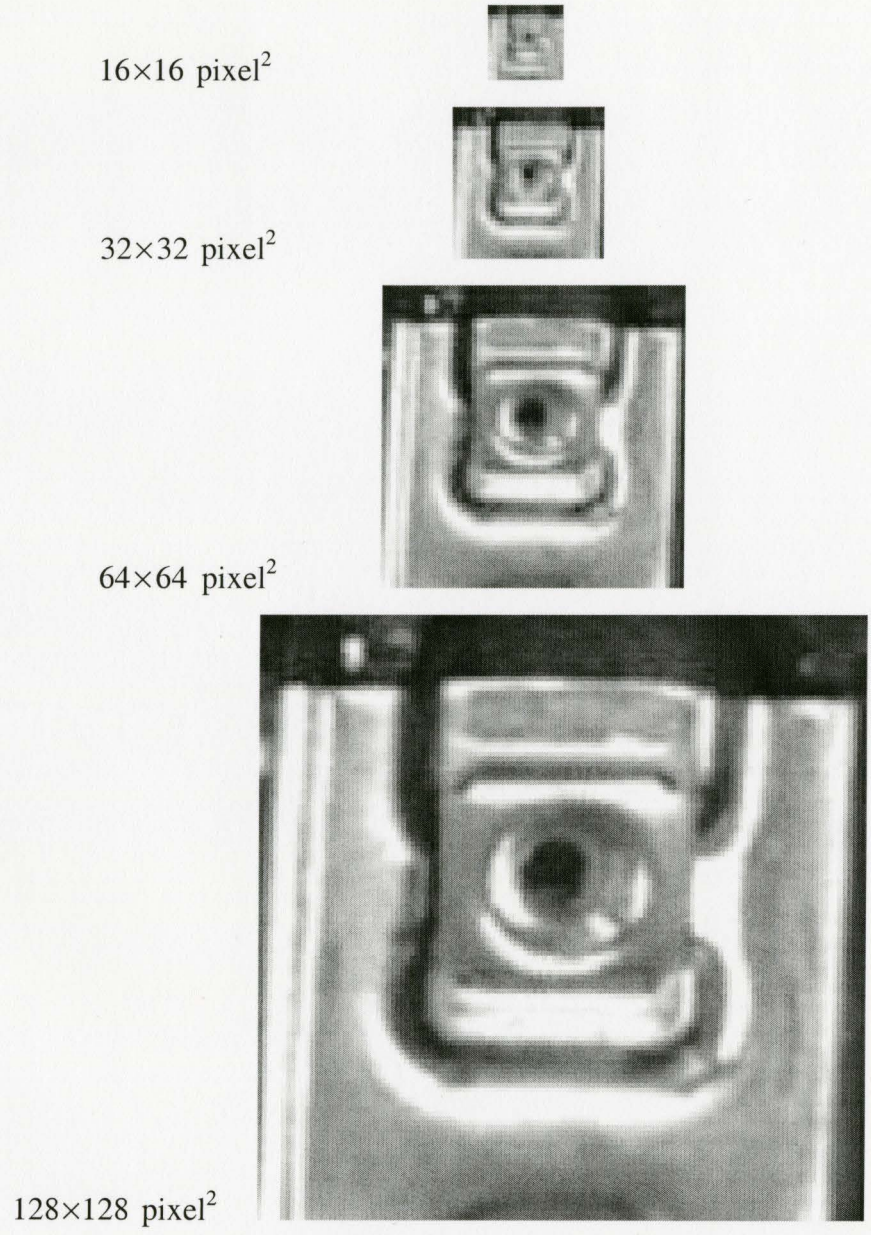


Figure 6.7 Four level Gaussian pyramid

To further reduce the amount of input data the images were binarized from 8-bit gray scale using a manually chosen threshold of 100. The final size of the input vectors equalled 16×16×1 bit/pixel = 256 bits. Examples of original and condensed images are compared in Figure 6.8. It can be observed that sufficient information remains for classifying the two cases.

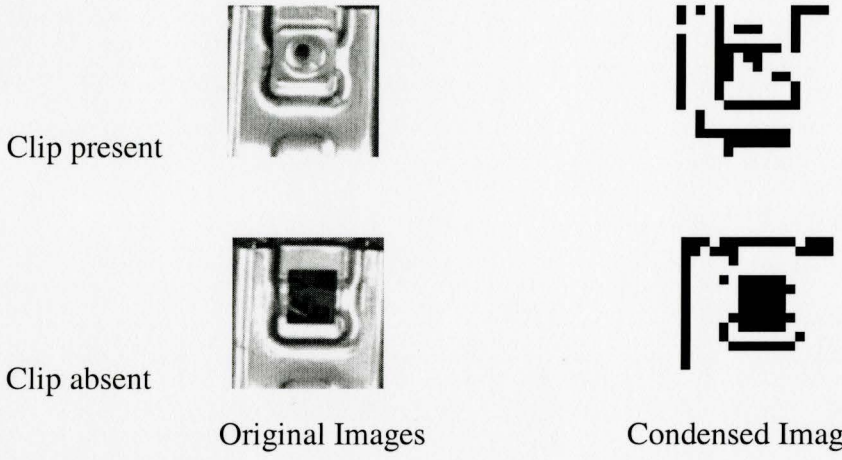


Figure 6.8 Comparison between the original and condensed images.

6.4.4 System Training

The CMAC based machine vision training algorithm is summarized below.

Algorithm 6.1 CMAC Based Machine Vision Training Algorithm

Step 1: Randomly choose a number between 1 and 100 as the index of the image to be checked;

Step 2: Segment the ROI from the selected image;

Step 3: Down sample and then binarize the ROI to convert the ROI into the desired input format;

Step 4: Input the image to the CMAC neural network and calculate the error between the network output and the desired classification value;

Step 5: Adjust the network weights with respect to the error;

Step 6: If the number of the training cycles is larger than the predefined limit then continue; Else go to step 1 to run a new training cycle;
Step 7: Save the weights of the CMAC neural network, and stop.

After preprocessing the original automotive part images, we input the condensed images into the CMAC neural network to train it. In this application the CMAC neural network functions as a classifier, and the network will be trained under supervision. Before training the network, we needed to classify the original images manually. The CMAC software, described further in section 6.5, uses integers for the weights and the network output for faster processing so the manual classification (or desired response) should use integers. Images with clips were assigned a value of 1000 while images without clips were assigned a value of -1000.

After manually classifying the training images, we input them and the classification values into the neural network. The training images should be learned in a random order. If the training images are input in a certain sequence, the sequence of the images will also be learned and the classification performance will be worse. To ignore the order of the sequence, we need to train the network by applying the training images in a random sequence. Note that every training image will appear multiple times in this sequence. To generate the random image sequence, we applied the standard C function rand() to obtain a pseudo-random sequence for the image indices. Then we trained the network using the pseudo-random ordered training images.

By observing the mean absolute error between the actual output value and the desired output value of the network, we could monitor the training procedure. The mean absolute training errors are shown in Figure 6.9. Figure 6.9 is also termed the training curve. We can observe that the training error converges very quickly.

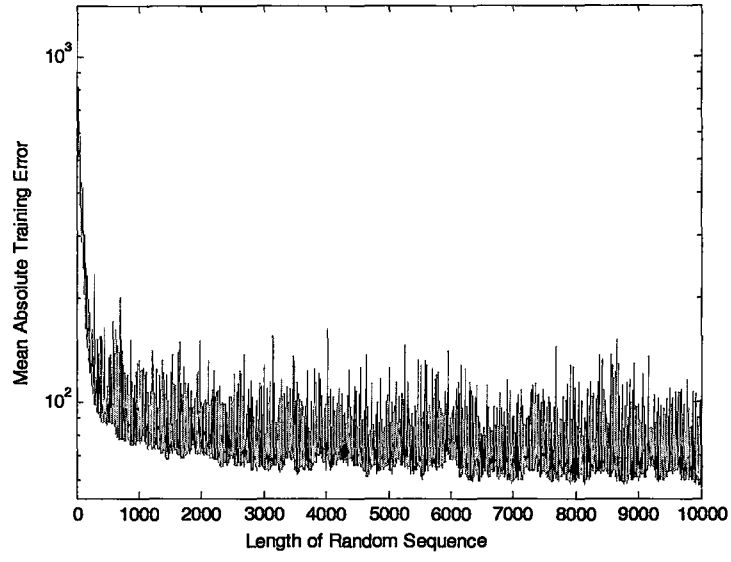


Figure 6.9 Training curve.

In order to monitor the network training procedure, we use an image that does not belong to the training image set as the testing image to test the trained network step by step. After one training image is input into the network and the adjustable weights are adjusted, the test image is input into the CMAC to test the network trained by one step. The absolute error between the CMAC output and the desired response is recorded. The absolute testing errors for each step are shown in Figure 6.10. The shape of the absolute testing error curve is similar to the training curve. It converges very fast. The absolute testing errors are larger than the training errors since the testing image is not in the training set.

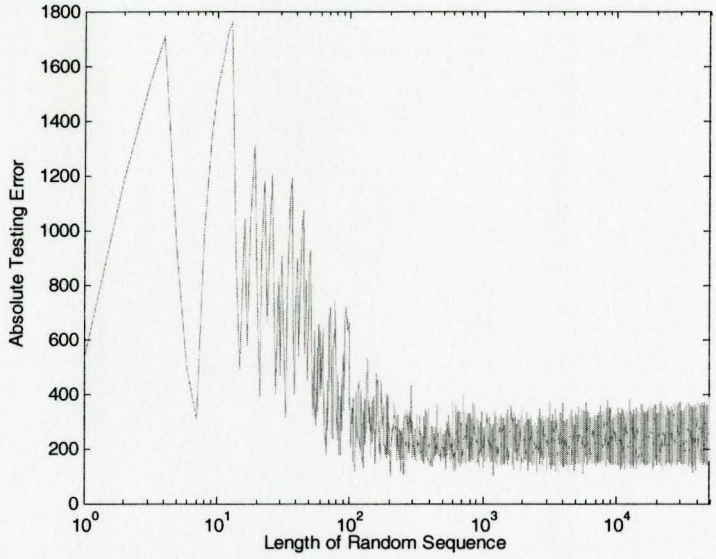


Figure 6.10 Testing curve.

For a neural network, over-training is an important problem to be mentioned. It is very easy to understand the over-training problem. If a neural network is trained by applying the training data set too many times, the output of the neural network will more and more accurately represent the training data. However, if the neural network learns the given features too accurately, larger errors will be generated when new features appear in the testing data. As the result, the generalization ability of the neural network will be degraded by over-training and this can produce more classification errors. It is important to avoid this over-training. In this thesis, the over-training problem was not significant since the images were preprocessed before they were input to the CMAC neural network for training and testing. The preprocessed images are binary ones and a fixed threshold was used to binarize them. As a result, the difference between the images for training and the ones for testing is so slight that fewer new features will appear in the testing images. So the over-training problem is very difficult to be observed. We can also draw this conclusion from Figure 6.10.

6.5 CMAC Software and Testing Procedure

In this thesis, we used the CMAC neural network training and testing software implemented in C language by Miller [77]. In this software, there are several main input parameters as follows:

- NUM_CMACS: the maximum number of independent CMACs used in the software. The default value of this parameter is set as eight. In this thesis, we only use one CMAC to classify the images. So this parameter is set as the default value;
- MAX_STATE_SIZE: the maximum number of input dimensions for the input state space. In this thesis, the size of the original ROI is 128 by 128. So we set this parameter with the value of 16,384;
- MAX_RESPONSE_SIZE: This parameter determines the maximum number of output dimensions. In this thesis, the output of the CMAC is the classification number of the image that is a scalar. We set this parameter as 1;
- MAX_GEN_SIZE: This parameter describes the maximum number of the memories connected to one input state vector. We set this parameter with a large value (65,536) to observe the relationship between the CMAC network performance and the number of the memories connected to one input;
- num_state: the number of input dimensions for the input state space. Since the size of preprocessed ROI that is input to the network is 16 by 16, we set this parameter with a value of 256;
- num_response: the number of the dimensions of the CMAC output. The output of the network is the classification number of the image, which is a scalar. So the parameter num_response is set equal to 1;
- num_cell: the number of memory cells connected to one input state vector. A small value for this parameter makes the CMAC training procedure fast with less accurate output. A large value for this parameter makes the

network more accurate but the training procedure is slow. By manually adjust the parameter, we assign the parameter num_cell with the value of 1,024 to obtain satisfactory experimental results;

- memory: the maximum number of the memory cells can be used in the CMAC software. In this thesis, we care more about the software performance than the physical hardware specifications, so we set this parameter a large value (10,000);
- field_shape: This parameter sets the design of the CMAC receptive fields, using predefined constants. Different designs will make different training and classification results. After making experiments on different values of the parameter field_shape, we chose *RECTANGULAR* to obtain the best performance. By using the parameter equal *RECTANGULAR*, the on-off receptive fields in the CMAC are created in a uniform arrangement;
- collide_flag: This parameter is set TRUE or FALSE to indicate whether or not to allow hashing collisions. If the hashing collisions are allowed, less memory cells will be occupied. But the performance of the network is worse than the network without hashing collision. Since we are concerned more about the CMAC performance, the parameter collide_flage is set FALSE in this thesis;
- beta1 and beta2: These two parameters are used to control the adjustment of the CMAC weights. Their meanings were described in section 6.3.3. In this software, beta1 and beta2 are right shift factors (beta1 = 1 means $\beta_1 = 0.5$, beta2 = 2 means $\beta_2 = 0.25$, etc). As discussed before, a large β_1 forces the CMAC output close to the desired response and a large β_2 makes the CMAC performance more stable. At the same time, it will take longer time to train the network when a large β_2 is used. With tradeoff between the CMAC performance and the software running time, we chose

a large β_1 ($\beta_1 = 0.5$) and a relative small β_2 ($\beta_2 = 2^{-10}$) in the experiments.

Note that the data type used in this CMAC software is integer in order to make the software run faster. So proper gains must be multiplied to reach the desired accuracy.

After training the CMAC neural network, we use the trained network as the image classifier for the automotive beam clip present/absent problem. We apply the trained weights stored in the network to test the unused images. Actually, the testing procedure of the CMAC neural network is almost the same as the training procedure without adjustment of the weights.

6.6 Experimental Results

The CMAC based machine vision system was developed and tested on a computer with the processor of Intel P4 2.8GHz and 512Mb RAM. The software to preprocess the images was developed under Matlab 6.5 and the average time to preprocess an image is 65 ms.

In this automotive beam clip present/absent inspection problem, we apply 100 images as the training data set. In the training images, there are 50 images with the clips and 50 images without the clips. From this set of 100 images we generated pseudo-random sequences with lengths up to 50,000. We apply the 100 unused images to test the network when the length of the pseudo-random sequence equal 1,000, 5,000, 10,000 and 50,000, respectively. The softwares to train and test the network were developed under Microsoft Visual C ++ 6.0. The average time to classify a preprocessed image using the trained network is 11ms. The CMAC network has the parameters described in Section 6.5. The training results and the test results are listed in Table 6.1.

Table 6.1 Experimental results

Length of Random Sequence	Training Time (Sec.)	False Classification No.	Uncertain No.	Used Memory (Cells)
1,000	16	3	3	1,218
5,000	74	0	3	1,218
10,000	151	0	0	1,218
50,000	759	0	1	1,218

From Table 6.1, we can observe that with the increment of the length of the random sequence, the training time increases linearly, and the number of the false classification on the testing data decreases. However, the number of the uncertain classification is increased a little bit when the length of the testing data increases to 50,000. This increment is due to the occurance of slight overtraining since we trained the

neural network by applying the same training data set for too many times. Table 6.1 also shows that the number of the memory cells used to store the weights in the CMAC neural network is unchanged for the listed four cases. The used memory curve for the whole training procedure is shown in Figure 6.11. In the figure, we can see that the number of the used memory increases sharply from the beginning of the network training, and it reaches a constant number very quickly. This also demonstrates the fast learning speed of the CMAC neural network. After training the network in 151 seconds, the classification accuracy can reach 100%. However, in our training and testing set, the manufacturer only provided us with 55 images without the clips. Further experiments should be done on a larger set of images to better test the performance of this inspection system.

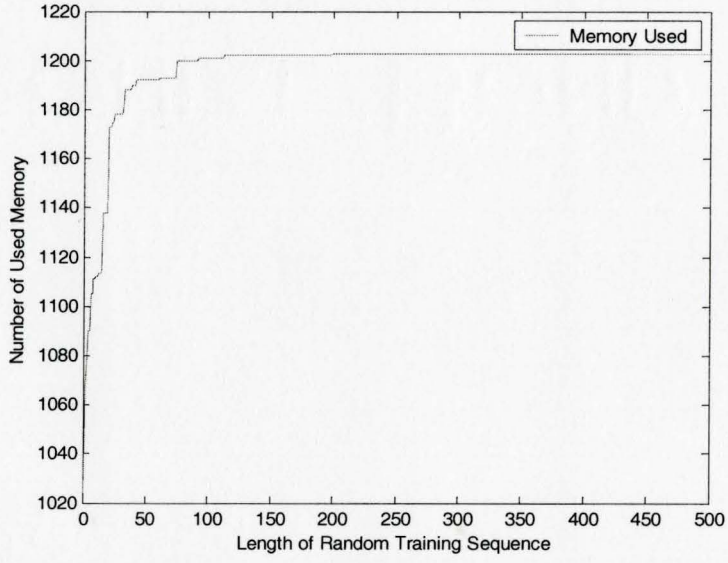


Figure 6.11 Used memory curve.

6.7 Summary

In this chapter, we developed a CMAC neural network based machine vision system for the automotive beam clip present/absent inspection problem. A preprocessing procedure was introduced to convert the images to down-sampled binary images. These condensed images were used as the input to the CMAC neural network for training and testing. Experiments showed that after training in 151 seconds, the classification accuracy of the network reached 100% for the current testing set provided by the manufacturer. And the average time to classify an image was found to be 11ms, which is fast enough for real-time manufacturing applications.

Chapter 7

Conclusions and Recommendations

7.1 Summary

In this research the developments of a pixel-by-pixel analysis based adaptive machine vision system and a neural network based machine vision system were presented. The pixel-by-pixel analysis based adaptive machine vision system was created for an automotive water pump housing part surface inspection problem and the neural network based vision system was created for an automotive clip present/absent inspection problem. The pixel-by-pixel analysis based vision system is suitable for the inspection problems where the inspection rules can be expressed mathematically. The neural network based vision system is suitable for inspection problems whose rules are difficult to define and can be solved using down-sampled images. The experimental results demonstrated the performance of both machine vision systems.

7.2 Achievements

The main achievements of this thesis are summarized as follows.

(1) This research studied the adaptive image acquisition. By applying a controlled lighting work cell and a novel adaptive camera control algorithm images with higher quality than possible with fixed camera parameters were acquired.

(2) A novel adaptive seeded region growing method was presented. The histogram based seeded region growing algorithm was shown to segment the pixels belonging to the holes better than the adaptive thresholding method and the fixed seeded region growing method.

(3) An LMS based image registration algorithm was demonstrated to match up the mask with the ROI accurately. The accurately registered mask was used to remove the areas outside of the ROI, resulting in faster processing.

(4) A novel local standard deviation followed by adaptive global thresholding algorithm was demonstrated to detect defects on the inspected surface. It was also able to separate the machining marks from the defects in the majority of cases.

(5) With the automotive water pump housing part surface inspection problem, the defects can be less than 1mm in diameter while the surface is roughly 180mm by 110mm. The proposed vision system can find detects as small as 0.15mm. In experiments, the system has been tested with over 1,700 images. The majority of the defects were pores. These pores were correctly classified in 93% of the cases.

(6) The surface inspection system can inspect the ROI in less than 1 second, making it fast enough for use in the automotive manufacturing industry.

(7) A CMAC neural network based vision system was demonstrated to be well suited to an automotive clip present/absent inspection problem. The system was able to correctly classify 100% of the 100 test images after training for only 151 seconds. However, it should be tested with more images to better evaluate its performance.

(8) The clip present/absent inspection system can inspect a part in less than 80ms, making it fast enough for use in the automotive manufacturing industry.

7.3 Recommendations for Future Work

(1) The quality of the image has a big impact on the inspection accuracy. With the controlled lighting work cell, we fixed the position and orientation of the lighting sources. A dynamically adjustable lighting source should be investigated to improve the image quality. The combination of several images obtained under different lighting conditions for one automotive part should improve the image quality.

(2) The accuracy of the image registration has a significant effect on the inspection accuracy of the surface inspection problem. The ROI mask is not only applied to segment the inspected surface, but also as a reference for the surface contour and the sealing contour. The accuracy of the image registration is critical when classifying defects near the edges and when identifying the sealing area. A more

accurate image registration algorithm should be developed to improve the performance of the surface inspection system.

(3) In the surface inspection problem, the vision system was able to correctly find the defects, while ignoring the machining marks, in the majority of cases. If the machining marks are very obvious they confuse the system. Texture based inspection algorithms should be studied as a potential solution to this problem.

(4) The fixed increment used with the adaptive camera control algorithm made its performance slow. A better algorithm should be developed to speed up the image acquisition process.

(5) Larger sets of training and testing images should be used to better train and validate the neural network based vision system.

Reference

[1] G. Norel, "Measurement and inspection in boreholes by ultrasound," in *3rd International Symposium on Non-Destructive Inspection Methods*, pp. 1 - 5,1974.

[2] Y. Okawa, "Automatic inspection of the surface defects of cast metals," in *Computer Vision, Graphics, and Image Processing*, v 25, n 1, pp. 89-112, Jan, 1984.

[3] A. A. Uglov, V. A. Grebennikov, V. G. Panaetov, M. B. Ignat'ev, "Inspecting the porosity of components in powder metallurgy using the laser beam," in *Physics and Chemistry of Materials Treatment*, v 20, n 4, pp. 320-321, July-Aug. 1986.

[4] Yu. E. Khoroshailo, N. E. Khoroshailo, O. I. Stoyan, "Inspecting the porosity of thin nonferromagnetic films by an eddy current method," in *Radiotekhnika*, n 94, pp. 126-30, 1991.

[5] C. Fernandez, C. Platero, P. Campoy, R. Aracil, "Vision system for on-line surface inspection in aluminium casting process," in *Proceedings of the IECON '93., International Conference on Industrial Electronics, Control, and Instrumentation, 1993*, v 3, pp. 1854 - 1859, Nov. 1993.

[6] G. Zeichen, H. Hufnagl, M. Berger, "Flexible industrial inspection of surface defects using a transputer image processing system," in *Proceedings of SPIE - The International Society for Optical Engineering*, v 2183, pp. 112-119, 1994.

[7] C. Platero, C. Fernandez, P. Campoy, R. Aracil, "Surface analysis of cast aluminium by means of artificial vision and A. I. based techniques," in *Proceedings of SPIE - The International Society for Optical Engineering*, v 2665, pp. 36-46, 1996.

[8] P. D. Dean, "Integrated nondestructive inspection (INDI): the HMO framework for the next millennium," in *Process Control and Sensors for Manufacturing*, v 3399, pp. 173 - 177, 1998.

[9] D. Tsai, C. Tseng, "Surface roughness classification for castings," in *Pattern Recognition*, v 32, n 3, pp. 389-405, Mar. 1999.

[10] D. Mery, D. Filbert, "Automated flaw detection in aluminium castings based on the tracking of potential defects in a radioscopic image sequence," in *IEEE Transactions on Robotics and Automation*, v 18, n 6, pp. 890-901, December, 2002.

[11] D. Tsai, T. Huang, "Automated surface inspection for statistical textures," in *Image and Vision Computing*, v 21, n 4, pp. 307-323, Apr 1, 2003.

[12] V. Kaftandjian, G. Lecomte, E. Cendre, J. Rheinlander, "Combination of x-ray imaging and spectra measurements for improving automatic inspection of aluminium cast components," in *Proceedings of SPIE - The International Society for Optical Engineering*, v 5132, pp. 410-421, 2003.

[13] "Real-time x-ray integrates porosity measurement and flaw detection," in *Foundry Trade Journal*, v 179, n 3629, p 271, November 2005.

[14] A. Lamarre, O. Dupuis, M. Moles, "Complete inspection of friction stir welds in aluminium using ultrasonic and eddy current arrays," in *CINDE Journal*, v 27, n 4, pp. 14-22, July-Aug. 2006.

[15] D. Steiner, R. Katz, "Measurement techniques for the inspection of porosity flaws on machined surfaces," in *Journal of Computing and Information Science in Engineering*, v 7, n 1, pp. 85-94, March, 2007.

[16] P. Gamage, S. Q. Xie, "A real-time vision system for defect inspection in cast extrusion manufacturing process," in *Mechatronics and Machine Vision in Practice*, pp. 240 - 245, Dec. 2007.

[17] D. Mery, M. Carrasco, "Automated multiple view inspection of metal castings," in *Proceedings of SPIE - The International Society for Optical Engineering*, v 6356, Eighth International Conference on Quality Control by Artificial Vision, p 63560C, 2007.

[18] R. Y. Tsai, "A versatile camera calibration technique for high-frequency 3D machine vision metrology using off-the shelf TV cameras and lenses," in *IEEE Journal of Robotics and Automation*, 3(4), pp. 323 - 344, Aug. 1987.

[19] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," in *The International Journal of Computer Vision*, 8(2), pp. 123 - 152, Aug. 1992.

[20] O. Faugeras, *Three-Dimensional Computer Vision: a Geometric Viewpoint*, MIT Press, 1993.

[21] R. I. Hartley, "An algorithm for self calibration from several views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 908 - 912, Seattle, WA, June 1994.

[22] Q. T. Luong and O. Faugeras, "Self-calibration of a moving camera from point correspondences and fundamental matrices," in *The International Journal of Computer Vision*, 22(3), pp. 261 - 289, 1997.

[23] S. Bougnoux, "From projective to Euclidean space under any practical situation, a criticism of self-calibration," in *Proceedings of the 6th International Conference on Computer Vision*, pp. 790 - 796, Jan. 1998.

[24] Z. Zhang, "A flexible new technique for camera calibration," in *IEEE Trans. Pattern Anal. Mach Intell.*, 22(11), pp. 1330 - 1334, 2000.

[25] J. F. Blinn, M. Newell, "Texture and reflection in computer generated images," in *Communications of the ACM (SIGGRAPH 76 Proceedings)* 19, 10, pp. 542–547, Oct. 1976.

[26] J. F. Blinn, "Computer display of curved surfaces," Ph.D. Thesis, University of Utah, Salt Lake City, Utah, USA, 1978.

[27] S. Mann, "Compositing multiple pictures of the same scene," In *Proceedings of the 46th Annual IS&T Conference*, Cambridge, Massachusetts, May 9-14 1993. The Society of Imaging Science and Technology.

[28] T. Malzbender, D. Gelb, H. Wolters, "Polynomial texture maps," in *Proc. ACM Siggraph*, ACM Press, pp. 519-528, 2001.

[29] R. Raskar, A. Agrawal, J. Tumblin, "Coded exposure photography: motion deblurring using fluttered shutter," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, Vol. 25, No. 3, pp. 795-804, 2006.

[30] A. Levin, R. Fergus, F. Durand, W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2007.

[31] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, J. Tumblin, "Dappled photography: mask enhanced cameras for heterodyned light fields and coded aperture refocusing," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2007.

[32] C. K. Chow, T. Kaneko, "Automatic boundary detection of the left ventricle from cineangiograms," in *Computers and Biomedical Research 5*, pp. 388 - 410, 1972.

[33] Y. Nakagawa, A. RosenFeld, "Some experiments on variable thresholding," in *Pattern Recognition*, Vol. 11, pp. 191 - 204, 1979.

[34] J. M. White, G. D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction," in *IBM J. Res. Dev.*, 27(4), pp. 400–411, 1983.

[35] A. P. N. Plummer, F. Dale, *The Picture Processing Language Compiler Manual*, National Physical Laboratory, Teddington, 1984.

[36] J. Bernsen, "Dynamic thresholding of gray level images," in *ICPR'86: Proc. Intl. Conf. Patt. Recog.*, pp. 1251 - 1255, 1986.

[37] N. B. Venkateswarluh, R. D. Boyle, "New segmentation techniques for document image analysis," in *Image Vis. Comput.* 13, pp. 573 - 583, 1995.

[38] M. Sezgin, B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," in *Journal of Electronic Imaging*, 13(1), pp. 146 - 165, January 2004.

[39] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, 3rd Edition, Amsterdam, Boston, Elsevier, 2005.

[40] J. M. S. Prewitt, B. S. Lipkin, A. Rosenfeld, "Object enhancement and extraction," in *Picture Processing and Psychopictorics*, Academic Press, New York, pp. 75 - 149, 1970.

[41] I. Sobel, G. Feldman, "A 3x3 isotropic gradient operator for image processing," presented at a talk at *the Stanford Artificial Project in 1968*, unpublished but often cited, orig. in *Pattern Classification and Scene Analysis*, Duda, R. and Hart,P., John Wiley and Sons, pp. 271-272, 1973.

[42] J. Canny, "A computational approach to edge detection," in *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 8, pp. 679-714, 1986.

[43] R. Adams, L. Bischof, "Seeded region growing," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 6, pp. 641 -647, June 1994.

[44] J. Gao, M. Zhao, H. Wang, "A threshold and region growing combined method for filament disappearance area detection in solar images," in *Conference on Information Sciences and Systems*, March 2001.

[45] B. Zitova, J. Flusser, "Image registration methods: a survey," in *Image and Vision Computing*, Vol. 21, pp. 977 - 1000, 2003.

[46] A. Sinha, X. Wu, "Fast generalized motion estimation and super-resolution," in *ICIP 2007*, Vol. 5, pp. 413 - 416, 2007.

[47] K. Karhunen, Kari, "Über lineare Methoden in der Wahrscheinlichkeitsrechnung," in *Ann. Acad. Sci. Fennicae. Ser. A. I. Math.-Phys.*, No. 37, pp. 1 – 79, 1947.

[48] B. Widrow, "Generalization and information storage in networks of adaline 'neurons'," in *Self-Organizing Systems* (M. C. Yovitz, G. T. Jacobi, and G. D. Goldstein, eds.), pp. 435 - 461, Washington, D. C.: Sparta, 1962.

[49] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," in *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* , Vol. 97, pp. 220-227, 1975.

[50] J. S. Albus, "Data storage in the cerebellar model articulation controller (CMAC)," in *Transactions of the ASME* , Vol. 97, pp. 228-233, 1975.

[51] P. J. Burt, E. H. Adelson, "The Laplacian pyramid as a compact image code," in *IEEE Transactions on Communications*, Vol. COM - 31, No. 4, pp. 532 - 540, April, 1983.

[52] W. T. Miller, F. H. Glanz, L.G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," in *The International Journal of Robotics Research*, Vol. 6, No. 2, pp. 84 - 98, 1987.

[53] J. Moody, C. J. Darken, "Fast learning in networks of locally tuned processing units," in *Neural Computation*, Vol. 1, pp. 281-294, 1989.

[54] W. T. Miller, F. H. Glanz, "Cerebellar model arithmetic computer," in *Fuzzy Logic and Neural Network Handbook*, Vol. 1, Chapter 26, McGraw-Hill, 1996.

[55] S. J. Julier, J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls 3*, 1997.

[56] J. Geng, T. N. Lee, "Freeway traffic incident detection using fuzzy CMAC neural networks," in *The 1998 International Conference on Fuzzy Systems Proceedings, IEEE World Congress on Computational Intelligence*, Vol. 2, pp. 1164 - 1169, May 1998.

[57] J. Carusone, G. M. T. D'Eleuterio, "The feature CMAC: a neural network based vision system for robotic control," in *Preceedings of the 1998 Ieee, International Conference on Robotics and Automation*, pp. 2959 - 2964, May 1998.

[58] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.

[59] D. Cornforth, "Building practical classifiers using cerebellar model associative memory neural networks," in *ANNES*, 2001.

[60] H. Jaeger, H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," in *Science 2*, Vol. 304. No. 5667, pp. 78 - 80, April 2004.

[61] H. Fashandi, M. S. Moin, "Face detection using CMAC neural network," in *ICAISC 2004, LNAI 3070*, pp. 724 - 729, 2004.

[62] C. S. Williams, O. A. Becklund, *Optics: a Short Course for Engineers and Scientists*, Toronto: Wiley-Interscience, 1972.

[63] Point Grey Research Inc. Technical Staff, *Scorpion Technical Reference Manual*, Point Grey Research Inc., 2004.

[64] Sony Corporation Technical Staff, *ICX267AL Datasheet*, Sony Corporation.

[65] PENTAX Corporation Technical Staff, *Machine Vision Lens*, PENTAX Corporation.

[66] Intel Corporation, "Open source computer vision library", November, 2006, http://sourceforge.net/projects/opencvlibrary/.

[67] P. V. C. Hough, "Method and means of recognizing complex patterns", U. S. Patent 3069654, 1962.

[68] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Fannery, *Numerical Recipes in C: The Art of Scientific Computing*, Second edition, New York: Cambridge University Press, 1992.

[69] A. M. Tekalp, *Digital Video Processing*, Upper Saddle River, NJ: Prentice Hall, 1995.

[70] G. M. Bone, "Robotics," class notes for Mech Eng 6K03, Department of Mechanical Engineering, McMaster University, Fall 2005.

[71] Image Processing and Multimedia Laboratory, Democritus University of Thrace, "Hough Transform", 2007, http://www.papamarkos.gr/index/category/135/.

[72] L. Lapin, *Statistics: Meaning and Method*, New York, Harcourt Brace Jovanovich Inc., 1975.

[73] M. P. Groover, *et. al*, *Industrial Robotics, Technology, Programming, and Applications*, pp. 177, McGraw Hill, 1986.

[74] R. C. Gonzalez, P. Wints, *Digital Image Processing*, Second Edition, Mass., Addison-Wesley Publishing Company, 1987.

[75] D. C. Chang, W. R. Wu, "Image contrast enhancement based on a local standard deviation model," in *IEEE Nuclear Science Symposium & Medical Imaging Conference*, v 3, pp. 1826-1830, 1996.

[76] R. M. Haralick, L. G. Shapiro, *Computer and Robot Vision: Volume II*, pp. 581, Reading, Mass., Addison-Wesley Publishing Company Inc., 1993.

[77] Robotics Laboratory, University of New Hamsphire, "CMAC neural network code", 1996, http://www.ece.unh.edu/robots/unh_cmac.c.