

Development of automated ultrasonic systems

Jiao, Shuxiang

2005

Jiao, S. (2005). Development of automated ultrasonic systems. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/5828>

<https://doi.org/10.32657/10356/5828>

Nanyang Technological University

Downloaded on 25 Aug 2022 00:20:21 SGT

DEVELOPMENT OF AUTOMATED ULTRASONIC SYSTEMS



JIAO SHUXIANG

SCHOOL OF MECHANICAL & AEROSPACE ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

2005

Development of Automated Ultrasonic Systems

Jiao Shuxiang

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2005

Abstract

Non-Destructive Testing is necessary in areas where defects in structures emerge over time due to wear and tear and structural integrity is necessary to maintain its usability. However most Non-Destructive Testing tasks are done manually. The operators set operational parameters manually to identify the defect. That results in many limitations of the current ultrasonic testing: high training cost, long training procedure, and worse, the fatigue of the operators can lead to unreliable and inconsistent test results. A prime objective of this project is to develop an Automatic Non-Destructive Testing System for a shaft of the wheel axle of a railway carriage.

To achieve the above objective, artificial intelligence and pattern recognition methods must be applied. In all the automated ultrasonic testing methods, the neural network is the most popular method. Some have also tried the knowledge-based system. In this thesis, a statistical pattern recognition approach, Classification Tree is applied.

Whichever approach is selected, the most important step is feature selection. It is necessary to learn about the distinguishing signal character as much as possible. For this purpose, a thorough study on the ultrasonic signals produced was carried out in the thesis.

Based on the analysis of the ultrasonic signals, three methods were developed to enhance the ultrasonic signals: Cross-Correlation with template signal, Zero-Phase filter and Averaging. The effect of the three methods is compared. The target of this step is to reduce the noise and make the signal character more distinguishable.

Four features are selected:

1. The AR Model Coefficients.
2. Standard Deviation.
3. Pearson Correlation (Pearson's "r").
4. Dispersion Uniformity Degree

And then a Classification Tree is applied to recognize the peak positions and amplitudes.

Based on this algorithm, a software package called SOFRA(Software for Research Aid) was developed to recognize the peaks, calibrate automatically and test a simulated shaft automatically. The automatic calibration procedure and the automatic shaft testing procedure are developed. SOFRA also has the capability to test defects in spot welds. The testing result of a spot weld in a Panasonic fridge compressor motor casing was found to be relatively good. A Kawasaki JS-5 robot is used to move the probe. Control program is developed to move the probe along the desired path.

Acknowledgement

The author would like to express his sincere appreciation to A/Prof. Stephen Brian Wong and A/Prof. Gerald Seet who gave the author the research direction, strong support, and invaluable advice throughout the project. Their valuable knowledge and recommendations gave the author much help.

Thanks are due to Wai Leong Yeo and Su Jialin, research students in Robotics Research Centre. Their good ideas are helpful. It has been a great pleasure to discuss issues with them.

Thanks are due to all my fellow students and the staff in the Robotics Research Centre. Without their help it would have been impossible to carry out the project.

Table of Contents

Abstract	i
Acknowledgement	ii
Table of Contents	iii
List of Figures	vi
List of Tables.....	ix
Abbreviation.....	x
Symbols.....	xi
Chapter 1 Introduction	1
1.1 Background.....	1
1.2 Objectives	3
1.3 Scope	3
1.4 Thesis outline.....	4
Chapter 2 Literature Review	8
2.1 Ultrasonic Wave Types	8
2.1.1 Wave Types	9
2.1.2 Foundation Knowledge in Ultrasonic Testing.....	11
2.2 Ultrasonic Non-Destructive Data Presentation	14
2.2.1 A Scan	14
2.2.2 B Scan.....	15
2.2.3 C Scan.....	15
2.3 A Statistical Way to Check the Similarity between Two Signals	16
2.3.1 Basic Concept.....	16
2.3.2 Pearson Correlation	17
2.4 Signal Processing.....	17
2.4.1 Signal Modeling	17
2.5 Pattern Recognition and Neural Networks	20
2.6 Classification Tree.....	27
2.6.1 Splitting Algorithm.....	29
2.6.2 Stopping Splitting Algorithm	31
2.6.3 Pruning Algorithm.....	31
2.7 Image Segmentation Methods	35

2.7.1	Watershed Segmentation	35
2.7.2	Active Contours	35
2.8	Chapter Summary	37
Chapter 3	Signal Analysis & Enhancement	38
3.1	Signal Property	38
3.2	Enhancement by Cross-Correlation	49
3.3	Enhancement by Filter	51
3.4	Enhancement by Averaging	54
3.5	Chapter Summary	55
Chapter 4	Peak Detection	57
4.1	Searching Local Maximum	57
4.2	Feature Selection	58
4.3	Using Classification Tree	61
4.4	Experiment Classification Result	67
4.5	Chapter Summary	68
Chapter 5	Automatic Testing Procedure	69
5.1	Statflow of SOFRA Software	69
5.2	Automatic calibration	71
5.3	Automatic Shaft Testing	76
5.4	Mechanical Parts Organization	83
5.5	C-Scan Testing	84
5.6	Weld Point Checking	87
5.7	Software Structure Improvement	93
5.8	Chapter Summary	94
Chapter 6	Conclusion and Future Work	95
6.1	Conclusion	95
6.2	Future Work	96
Bibliography	100	
Appendix A: Source Code of SOFRA	105	
Appendix B: Drawing of Probe Holder	106	
Appendix C: Source Code of Robot Control	109	
Appendix D: Matlab Programs	124	
Appendix E: Hardware Specification	173	

Appendix F: Source Code CD..... 175

List of Figures

Figure 1-1: Hardware Structure.....	6
Figure 1-2: Data Analysis Flow Chart.....	7
Figure 2-1: Wave Types [22]	10
Figure 2-2: Symmetric Lamb Wave [23]	10
Figure 2-3: AntiSymmetric Lamb Wave [23]	10
Figure 2-4: Reflection and Refraction [23]	11
Figure 2-5: Diffraction [23].....	12
Figure 2-6: A Scan [23]	14
Figure 2-7: B Scan [23]	15
Figure 2-8: C Scan [23]	16
Figure 2-9: AR Model	19
Figure 2-10: Neuron Structure.....	21
Figure 2-11: Three Activity Functions [30]	22
Figure 2-12: Single Neuron Perceptron.....	22
Figure 2-13: Single Layer Perceptron	22
Figure 2-14: Radial Basis Function [30]	24
Figure 2-15: Radial Basis Network	24
Figure 2-16: Tree Pruning	33
Figure 3-1: Signal from SOFRA4001H Card (Half).....	38
Figure 3-2: Signal Detail	39
Figure 3-3: Simulated Signal with 6 Cycles and its FFT	40
Figure 3-4: Rectified Simulated Signal with 6 Cycles and its FFT.....	40
Figure 3-5: FFT with 80MHz Sample Frequency	41
Figure 3-6: Actual Signal and its FFT.....	41
Figure 3-7: The initial Signal (truncated) and its FFT	42
Figure 3-8: The 2nd Signal (non-truncated) and its FFT	42
Figure 3-9: The 3rd Signal (truncated) and its FFT	42
Figure 3-10: The 4th Signal (non-truncated) and its FFT	42
Figure 3-11: The 5th Signal (non-truncated) and its FFT	43
Figure 3-12: The 6th Signal (non-truncated) and its FFT	43
Figure 3-13: The 7th Signal (non-truncated, low amplitude) and its FFT	43

Figure 3-14: The 8th Signal (very low amplitude) and its FFT	43
Figure 3-15: The 9th Signal (very low amplitude) and its FFT	43
Figure 3-16: SFFT of signal (Segment Length=40, Top view).....	44
Figure 3-17: SFFT of signal (Segment Length=40, ISO metric)	44
Figure 3-18: FFT Value (Frequency =0 MHz, Segment Length=40)	45
Figure 3-19: FFT Value (Frequency=10MHz, Segment Length=40)	46
Figure 3-20: SFFT of signal (Segment Length=20, Top view).....	47
Figure 3-21: SFFT of signal (Segment Length=20, ISO metric)	47
Figure 3-22: FFT Value (Frequency=0MHz, Segment Length=20)	48
Figure 3-23: FFT Value (Frequency=10MHz, Segment Length=20)	48
Figure 3-24: Using Cross-Correlation to Reduce Noise.....	49
Figure 3-25: FFT of Noise Signal	50
Figure 3-26: Cross-Correlation of Signal and Template Signal.....	51
Figure 3-27: Detail of Cross-Correlation	51
Figure 3-28: Frequency Response of the LP filter	52
Figure 3-29: Filter with non-Zero-phase filter	53
Figure 3-30: Filter with Zero-phase Filter.....	53
Figure 3-31: Detail of Figure 3-30 (Range 1275-1325).....	54
Figure 3-32: Signal with no average	55
Figure 3-33: Signal with 4-average	55
Figure 4-1: Feature a_1, a_2, a_3	60
Figure 4-2: σ , Pearson's "r" and DUD	60
Figure 4-3: Comparing Neural Network and Classification Tree	62
Figure 4-4: Peaks (red) and Non-peaks (blue) (1).....	62
Figure 4-5: Peaks (red) and Non-peaks (blue) (2).....	63
Figure 4-6: Whole ClassificationTree	63
Figure 4-7: Pruning Classification Tree	64
Figure 4-8: 1-SE Rule.....	65
Figure 4-9: Ultimate Classification Tree.....	66
Figure 5-1: SOFRA Main Interface.....	69
Figure 5-2: SOFRA Function	71
Figure 5-3: Meaning of "Precision"	72
Figure 5-4: Automatic Calibration Parameters and Result.....	72

Figure 5-5: Automatic Calibration	73
Figure 5-6: Automatic Shaft Testing Parameters	76
Figure 5-7: Peak Position (Shaft Testing Phrase 1).....	80
Figure 5-8: Peak Position (Shaft Testing Phrase 2).....	80
Figure 5-9: Peak Position (Shaft Testing Phrase 2) 3D.....	81
Figure 5-10: Peak Position (Shaft Testing Phrase 1) 3D.....	81
Figure 5-11: Report Result (Control Gain 36)	82
Figure 5-12: Real Defect Position	82
Figure 5-13: Probe Holder.....	83
Figure 5-14: Welding Point.....	84
Figure 5-15: C-Scan Data Expression	85
Figure 5-16: C-Scan Setting	86
Figure 5-17: Weld Point C-Scan Image	86
Figure 5-18: Original Weld Point Image.....	88
Figure 5-19: Detect Weld Point Position Using Watershed.....	89
Figure 5-20: Watershed Snake, Expanding Snake and Shrinking Snake.....	90
Figure 5-21: The Contour of Distance Function	90
Figure 5-22: Edge Map.....	91
Figure 5-23: Convergence of Expanding (A) and Shrinking (B) Snake	91
Figure 5-24: Weld Point Checking Result.....	92
Figure 5-25: Software Structure Improvement.....	93
Figure 6-1: Weld Defect Types	98

List of Tables

Table 4-1: Experiment Result.....	67
-----------------------------------	----

Abbreviation

ANN	Artificial Neural Network
AR	Autoregressive
ARMA	Autoregressive Moving Average
DAC	Distance Amplitude Correction
DUD	Dispersion Uniformity Degree
FFT	Fast Fourier Transform
GRNN	Generalized Regression Neural Network
GVF	Gradient Vector Flow
LSI	Linear Shift Invariant
LVQ	Learning Vector Quantization Network
MA	Moving Average
NDT	Nondestructive Testing
PNN	Probabilistic Neural Network
RBN	radial Basis Network
SFFT	Short-time Fast Fourier Transform
SMRT	Singapore Mass Rapid Transit
SOFRA	Software for Research Aid (Software name)
SOM	Self-Organizing Map Network
STD	Standard Deviation
TOFD	Time-of-flight diffraction

Symbols

v_l	Longitudinal wave velocity
v_s	Shear wave velocity
E_y	Young's modulus of elasticity
ν	Poisson's ratio
ρ	Material density
G	Shear modulus
V	Acoustic velocity of material
Z	Acoustic impedance
N	Length of near zone
R_{re}	Percent of the reflected energy with respect to the total energy
D	Diameter of the probe
λ	Ultrasonic wavelength in the medium
σ	Standard Deviation
μ	Mean
ρ_r	Pearson's "r"
$E[X], E[Y] \dots$	Expection value of $X, Y \dots$
$X = [x_1, x_2, \dots, x_n]'$	Input vector
$W = [w_1, w_2, \dots, w_n]$	Weights of the neuron
f_w	Neuron function
μ_l	Learning rate
E_r	Error of neural networks
E^t	Entropy
T	Classification tree
p	Probability
G_i	Impurity degree defined in Gini Diversity Index
R	Resubstitution error
$C(T)$	Cost function of tree
$\bar{v}(s)$	Active contour
E_{snake}	Active contour energy function

\vec{F}_{int}	Internal force
\vec{F}_{ext}	External force
E_{GVF}	Gradient Vector Flow energy function
E_{ext}	External energy function
$\vec{g}(x, y)$	Gradient Vector Flow vector field

Chapter 1 Introduction

1.1 Background

Most Non Destructive Testing (NDT) is designed to locate or identify defects in materials and products and as a method to control the quality and reliability of materials and products.

The essential feature of NDT is that the test process itself produces no deleterious effects on the material or structure under test.

Presently, most ultrasonic inspections are done manually. Most of the educational literature [1] on ultrasonic inspection has been written in technical language with engineering mathematics. Interpretation of ultrasonic NDT data can be a complex task, and interpreters must be highly skilled. When large engineering structures are inspected, the amount of data produced can be enormous, and a bottleneck can arise at the manual interpretation stage. Boredom and fatigue of operators can lead to unreliable and inconsistent results, where significant defects are not reported [2]. In order to improve their performance, it is recognized that there are gains to be derived from automating the ultrasonic inspection procedure by using a manipulator. Progress has been considerable in many aspects, namely, the development of equipment and techniques.

For the automatic characterization of the ultrasonic transducer, Obaidant and Abu-Saymeh compared the performance of a neural network and statistical pattern recognition techniques [3]. The neural network method obtained a better result compared with the statistical pattern recognition techniques. An automated system based on a neural network for characterizing an ultrasonic transducer was proposed in 1998 [4].

Miller has developed an automated real-time data acquisition system for robotic weld quality monitoring [5]. Abd El-Ghany developed software that has the ability to use Internet to transmit testing data [6]. The expert can evaluate the ultrasonic scanning

results for important structures without actually being present in the scanning site. Another useful tool is the simulation tool developed by Calmon [7]. The simulation tool can compute the ultrasonic fields of the realistic NDT configuration as an aid to NDT education and NDT test procedure optimization.

There are some automatic ultrasonic testing systems used for special purposes, such as an automatic system used to verify the integrity of aircraft parts [8], a multi-channel digital automatic ultrasonic testing system used in wheel detecting work [9], and some others [10].

In spite of all the modern automated inspection methods, defect classification still is a difficult task. Much work is essentially based on pattern recognition techniques and artificial intelligence methods using neural networks.

Two methods, “best-fit analysis” and “triangulation analysis”, were used to do shape classification [11]. Some success using neural networks has been reported with the use of the Hopfield [12] method and a back-propagation algorithm [13].

Drai [14] proposed a new signal processing technique known as “split spectrum processing”. It improves the signal-to-noise ratio. The cross-correlation function is used to determine the resemblance between the echoes emanating from crack edges and small volumetric defects. “Split spectrum processing” was also used by Yau [15].

Cornwel [16] used a rule-base expert system with fuzzy logic to deal with the problems of uncertainty during the automatic interpretation of ultrasonic NDT data [17]. There are also some other similar knowledge based systems [18]. Lawson describes the application of image processing and neural networks for the task of completely automating the decision making process involved in the interpretation of Time of Flight Diffraction (TOFD) images [19] [20].

1.2 Objectives

The prime objective of this project is to develop an Automatic Non-Destructive Testing System for a shaft from a wheel axle of a train carriage. The specific objectives of this research are:

1. Real-time ultrasonic test data acquisition.
2. Apply possible signal processing technologies to enhance the data acquired.
3. Select features and choose one suitable classification algorithm to identify the peaks (some are defects and some may come from the part structure) and also no peaks.
4. Develop an automatic calibration procedure as the first step of the whole automatic test procedure.
5. Develop an automatic shaft testing procedure best on the Singapore Mass Rapid Transit (SMRT) job specification [21]. This specification number will be kept confidential.

1.3 Scope

In order to complete the objectives, the author made the following active efforts during this project. Mathematical analysis, computer simulation and computer programming are used to achieve the objectives.

An ultrasonic card, SFT4001H, is used in this project. The author developed a C++ program to manipulate the SFT4001H Card by communicating with the Dynamic Link Library supplied by the ultrasonic card manufacturer.

A thorough study on the ultrasonic signal must be carried out before carrying on the project. Some Matlab programs using signal-processing technologies such as Fast Fourier Transform (FFT), Short Fast Fourier Transform (SFFT), auto-correlation, and cross-correlation, are developed to do the signal analysis. After getting the frequency character

of the signal, methods to enhance the signal (cross-correlation filter, zero-phase filter, and average filter) are compared.

Based on the frequency character and the shape character of the signal, four features are selected. In order to simplify the problem and also for the purpose of the real-time processing, the four features are reduced to three. One of the statistic pattern recognition methods, a classification tree, is applied to do the classification task. Although the neural networks have the similar function, the result of the classification tree is better for the project. The reason is that the training time of neural networks is much longer than that of classification tree. And more, the computation of classification tree is faster than that of neural networks. It is very suitable for real time processing.

Based on an ultrasonic non-destructive test standard and job specification of SMRT(Singapore Mass Rapid Transit), the automatic calibration procedure and the automatic shaft testing procedure are proposed. The two procedures are integrated in the SOFRA software developed by the author. SOFRA also has the capability to test for the defects in a spot-weld joint. The testing result of a spot weld joint of a Panasonic motor compressor casing is relatively good. The state flow of the SOFRA software is also developed which can animatedly simulate the running procedure of the software.

1.4 Thesis outline

Chapter 1: Introduces the background about ultrasonic non-destructive testing and describes the objective and the scope of the project. Shows the hardware structure of the automatic ultrasonic testing system (Figure 1-1) and the data analysis flow chart (Figure 1-2)

Chapter 2: Introduces the knowledge available about the waves applied in ultrasonic non-destructive testing, explores some common kinds of neural networks adopted for classification task, compares neural networks with other statistical pattern recognition

methods, describes a feature selection method (statistical parameter, signal modeling) and a recognition algorithm (classification tree) used in chapter 3 and chapter 4.

Chapter 3: Analyzes the property of an ultrasonic signal and, what the effect is for a respective parameter. Three methods to enhance the ultrasonic signal are studied: by filter, by correlation and by averaging. This is shown in block “Preprocessing” in Figure 1-2.

Chapter 4: Uses signal modeling to select signal features and compares those features with the statistical features. Use a classification tree to detect peaks. This is shown in block “Feature Selection” and block “Recognition” in. Figure 1-2.

Chapter 5: Describes the automatic calibration and automatic testing procedure. Shows the automatic calibration result, the automatic shaft testing result and the C-Scan result. Suggests the ways to improve the software structure. The design of the mechanical parts is shown in this chapter.

Chapter 6: Summarizes the report with recommendations for future work.

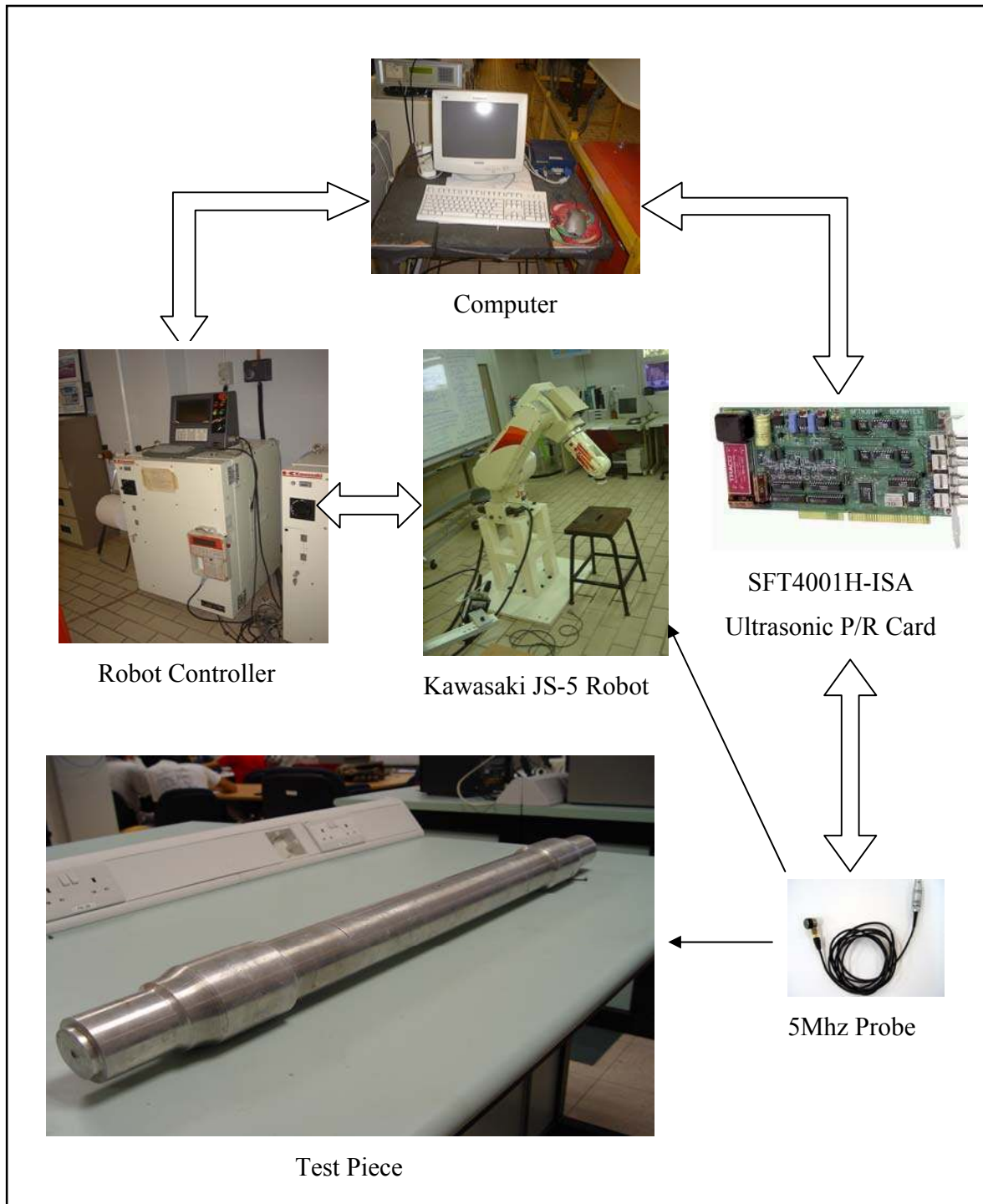


Figure 1-1: Hardware Structure

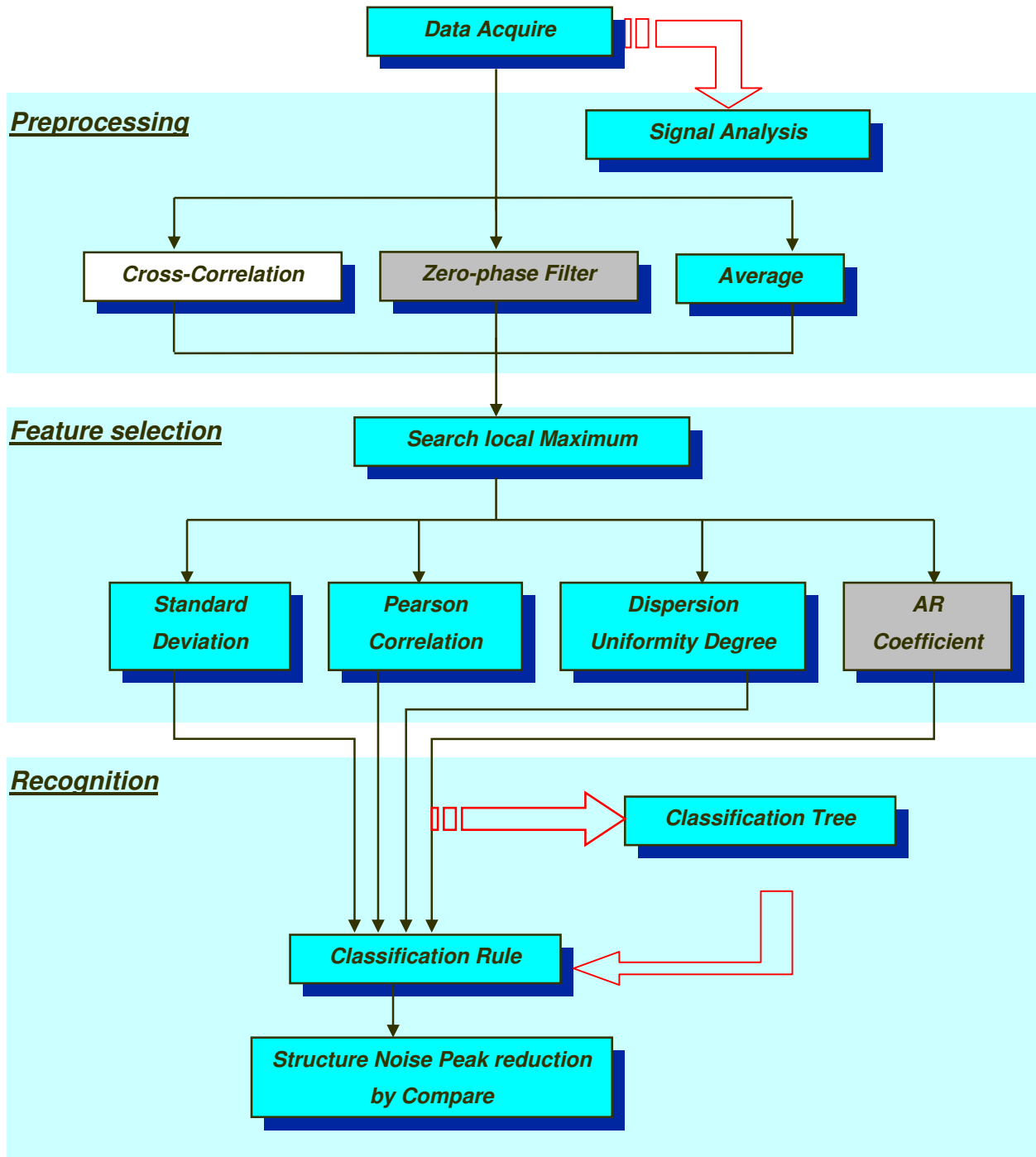


Figure 1-2: Data Analysis Flow Chart

Chapter 2 Literature Review

In this chapter, the knowledge about a wave applied in the ultrasonic non-destructive testing is introduced. Because the main target of the automatic ultrasonic testing system is to interpret the ultrasonic testing result (A-Scan, B-Scan, C-Scan and so on) automatically, the three types testing result are described. One method to get the feature of the signal is described. The methods used in Artificial Intelligence and Machine perception, neural networks and statistic pattern recognition are described also. The Classification Tree algorithm, one of the statistical pattern recognition methods is elucidated in detail. Two image segmentation algorithms, watershed and active contour, are introduced.

2.1 Ultrasonic Wave Types

Ultrasonic methods are perhaps the most versatile of the techniques for nondestructive testing available to industry. The two major methods of ultrasonic testing are:

1. Pulse-echo methods
2. Through-transmission methods

The pulse-echo method, which is the most widely used ultrasonic method, involves the detection of the echoes produced when an ultrasonic pulse is reflected from a discontinuity or an interface of a test piece. This method is used in flaw location and thickness measurements. Flaw depth is determined from the time-of-flight between the initial pulse and the echo produced by a flaw. Flaw depth can be determined by the relative transit time between the echo produced by a flaw and the echo from the back surface. Flaw sizes are estimated by comparing the signal amplitudes of reflected sound from an interface (either within the test piece or at the back surface) with the amplitude of sound reflected from a reference reflector of known size or from the back surface of a test piece having no flaws.

The through-transmission method, which may include either reflection or through transmission, involves only the measurement of signal attenuation. This method is also used in flaw detection. In the pulse-echo method it is necessary that an internal flaw reflect at least part of the sound energy onto a reflecting transducer. However, echoes from flaws are not essential to their detection. Merely the fact that the amplitude of a back reflection from a test piece is smaller than that from an identical work piece known to be free of flaws implies that that test piece contains one or more flaws. This technique of detecting the presence of flaw by sound attenuation is used in transmission methods as well as in the pulse-echo method. The main disadvantage of attenuation methods is that flaw depth cannot be made sure.

2.1.1 Wave Types

Ultrasonic testing is based on time-varying deformations or vibrations in materials, which is generally referred to as acoustics. All material substances are comprised of atoms, which may be forced into vibration motion about their equilibrium positions. Many different patterns of vibration motion exist at the atomic level. However, most are irrelevant to acoustics and ultrasonic testing. Acoustics is focused on particles that contain many atoms that move in unison to produce a mechanical wave. Provided a material is not stressed in tension or compression beyond its elastic limit, its individual particles perform elastic oscillations. When the particles of a medium are displaced from their equilibrium positions, internal (electrostatic) forces arise. It is these elastic restoring forces between particles, combined with inertia of the particles, which leads to oscillatory motions of the medium.

In solids, several types of wave propagation can occur that are based on the way the particles oscillate. Longitudinal and shear waves are the two modes of propagation most widely used in ultrasonic testing. The particle movement responsible for the propagation of longitudinal and shear waves is illustrated below.

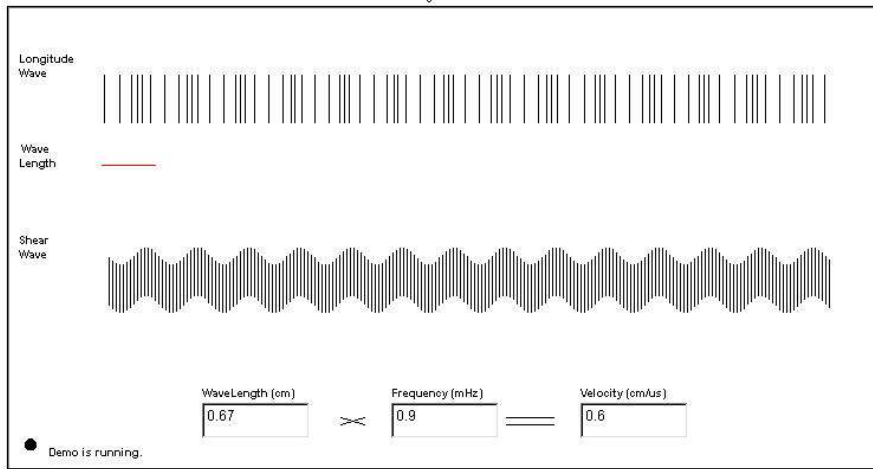


Figure 2-1: Wave Types [22]

There are two more wave types mentioned often in NDT,

1. Rayleigh wave (surface wave)
2. Lamb wave (guided wave, plate wave)

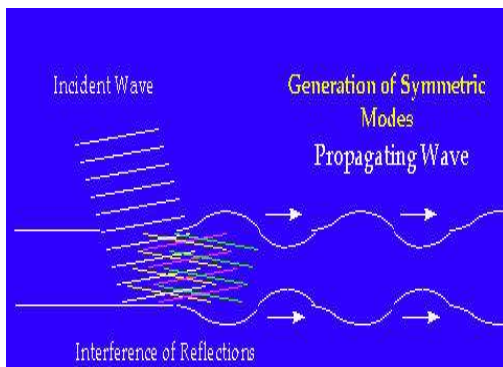


Figure 2-2: Symmetric Lamb Wave [23]

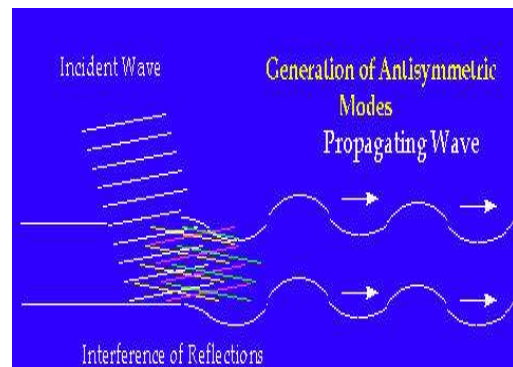


Figure 2-3: AntiSymmetric Lamb Wave [23]

Rayleigh wave is a particular type of surface wave, which propagates on the surface of a body, with decreasing amplitude exponentially with depth below the surface. The polarization of Rayleigh waves lies in a plane perpendicular to the surface. The effective

depth of penetration is less than a wavelength. When shear wave probes of 70° or over are used, surface waves may be generated at the entry surface. Those surface waves are also responsible for cross noise.

The kind of wave generated depends on many parameters: frequency, beam width, material, thickness and incident angle.

2.1.2 Foundation Knowledge in Ultrasonic Testing

2.1.2.1 Reflection and Refraction

Refraction is the change in direction of an acoustic wave as the ultrasonic beam passes from one medium into another having different acoustic impedance (sound velocity). A change in both direction and mode occurs at acute angles of incident. At small angles of incidence, the original mode and a converted one may exist in the second medium. A refracted beam occurs in the second media when an ultrasonic beam is incident at an acute angle on the interface between two media having different velocities (impedance).

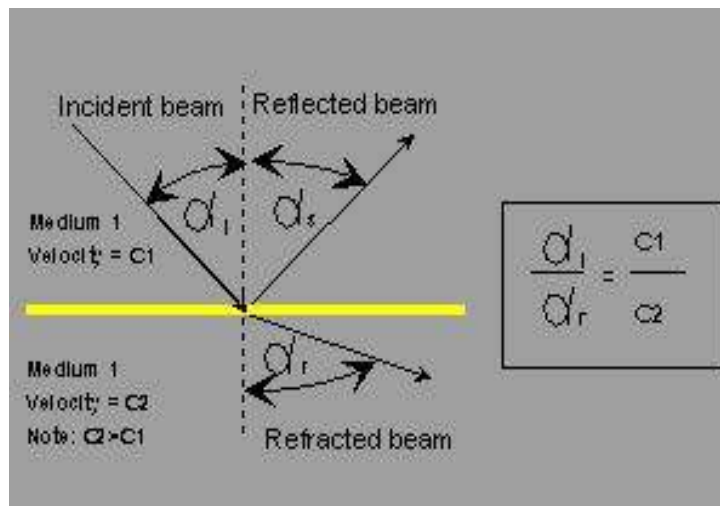


Figure 2-4: Reflection and Refraction [23]

2.1.2.2 Diffraction

The diffraction of a wave front occurs as it passes an ultrasonically opaque object and expands into the region that is behind the object and hence does not directly come from incoming waves. An extreme case arising from obstructions having very small dimensions is that of scattering. When ultrasound is incident at a linear discontinuity such as a crack, diffraction takes place at its extremities. The study of this phenomenon has led to the use of the time of flight diffraction method (TOFD) for crack sizing.

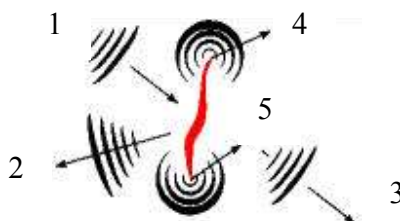


Figure 2-5: Diffraction [23]

2.1.2.3 Velocity and Acoustic Impedance

The wave velocity is determined by the material's elastic modulus and the density. As shown below:

For a longitudinal wave:

$$v_l = \sqrt{\frac{E_y(1-\nu)}{\rho(1+\nu)(1-2\nu)}} = \sqrt{\frac{E_y}{\rho}} \quad \text{Eq 2-1}$$

For a shear wave:

$$v_s = \sqrt{\frac{E_y}{2\rho(1+\nu)}} = \sqrt{\frac{G}{\rho}} \quad \text{Eq 2-2}$$

where:

v_l is the longitudinal wave velocity

- v_s is the shear wave velocity
 E_y is the Young's modulus of elasticity
 ν is the Poisson's ratio
 ρ is the material density
 G is shear modulus: $\frac{E_y}{2(1+\nu)}$

The acoustic impedance [24] Z is defined as

$$Z = \rho V \quad \text{Eq 2-3}$$

Where:

- ρ is material density
 V is the acoustic velocity of that material.

Acoustic impedance is important in [25]

1. The determination of acoustic transmission and reflection at the boundary of two materials having different acoustic impedance.
2. The design of ultrasonic transducers
3. Assessing absorption of sound in a medium.

The percent of the reflected energy with respect to the total energy is the square of the ratio of the difference of the acoustic impedances of the two materials divided by the sum of the acoustic impedances of the two materials.

$$R_{re} = \left(\frac{Z_2 - Z_1}{Z_2 + Z_1} \right)^2 \quad \text{Eq 2-4}$$

2.1.2.4 Near field

The region in an ultrasonic beam that is subject to variations of intensity due to diffraction effects. It extends from the source of radiation to a point just short of the far

field. Diffraction is a particular example of wave interference and is common to all wave motion.

The length of near zone N can be calculated by

$$N = \frac{D^2 - \lambda^2}{4\lambda} \cong \frac{D^2}{4\lambda} \quad \text{Eq 2-5}$$

where: D is the diameter of the probe

λ is the ultrasonic wavelength in the medium

$D \gg \lambda$

2.1.2.5 Attenuation of Sound Waves

It is well known that sound energy decreases with the distance traveled within the material. Sound pressure or signal amplitude, is only reduced by the spreading of the wave in idealized materials. Natural materials, however, all produce an effect that further weakens the sound. This further weakening results from two basic causes, which are scattering and absorption. The combined effect of scattering and absorption is called attenuation.

2.2 Ultrasonic Non-Destructive Data Presentation

2.2.1 A Scan

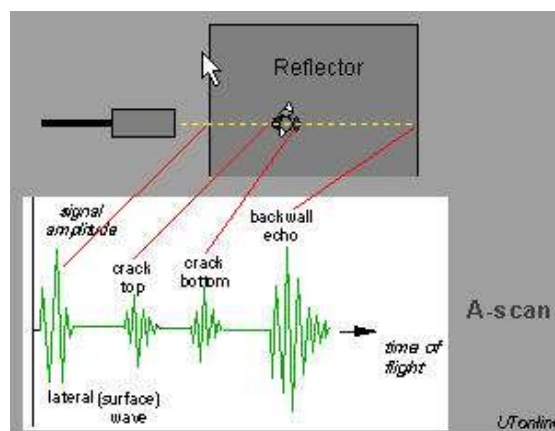


Figure 2-6: A Scan [23]

A display in which the received pulse amplitude is represented as a displacement along one axis (usually the y-axis) and the travel time of the ultrasonic pulse is represented as a displacement along the other axis (usually the x-axis). In a linear amplification system the vertical excursion is proportional to the amplitude of the signal. With the use of logarithmic scale amplifiers the y-axis presents a logarithmic scale.

2.2.2 B Scan

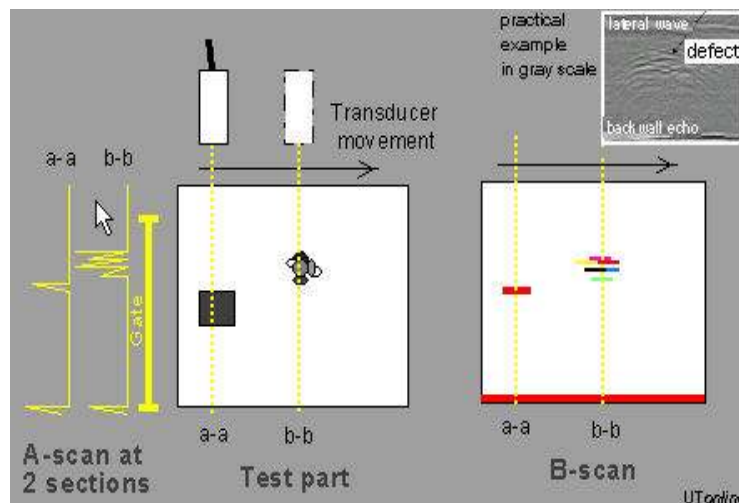


Figure 2-7: B Scan [23]

A two dimensional graphical presentation, in rectangular coordinates, in which the travel time of an ultrasonic pulse is represented as a displacement along one axis, and transducer movement is represented as a displacement along the other axis.

2.2.3 C Scan

For the old definition, C Scan is a two dimensional graphical presentation, in which the discontinuity echoes are displayed in a top view on the test surface. Now the C Scan image can be defined as a 3-dimensional image with the transducer movement plane as the X-Y plane and the time-of-flight (depth) as the Z-axis. Some visualization tool kits can be used to express the material inner information.

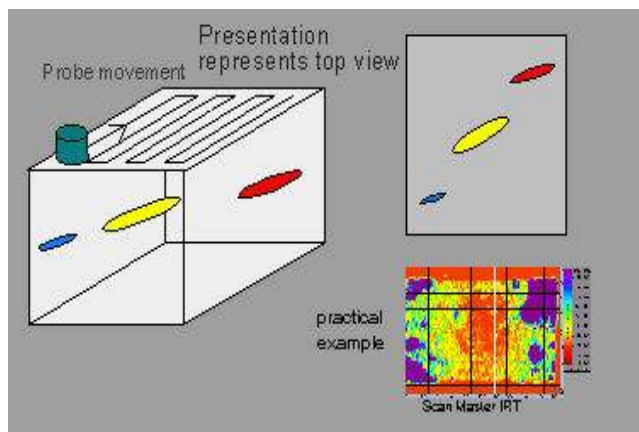


Figure 2-8: C Scan [23]

2.3 A Statistical Way to Check the Similarity between Two Signals

2.3.1 Basic Concept

- Expectation: The expectation of a random variable, X , with probability density function $p(x)$ is defined as $E(X) = \int_{x=-\infty}^{+\infty} xp(x)dx$. The discrete form is $E(X) = x_1P(X = x_1) + x_2P(X = x_2) + \dots + x_nP(X = x_n)$. When all probabilities are equal, the expectation of the random variable X is the arithmetic mean of x_1, x_2, \dots, x_n .
- Standard Deviation σ : The standard deviation σ can be calculated by:

$$\sigma^2 = E[(X - \mu)^2] \cong \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}.$$

The expectation value of a variable gives the center of the distribution of the variable. This value can be seen as the representative or average value of the variable. The standard deviation is a measure of how far the variable value is likely to be from the expectation value of the variable [26].

2.3.2 Pearson Correlation

The most widely used type of correlation coefficient is the Pearson's "r", also called linear correlation. The correlation coefficient determines the extent to which values of two variables are "proportional" to each other. The value of the correlation does not depend on the specific measurement units used; for example, the correlation between height and weight will be identical regardless of measurement units. Proportional means linearly related; that is, the correlation is high if it can be approximated by a straight line (sloped upwards or downwards). This line is called the regression line or least squares line, because it is determined such that the sum of the squared distances of all the data points from the line is the lowest possible. The Pearson correlation assumes that the two variables are measured on at least same interval scales. The Pearson product moment correlation coefficient is calculated as follows:

$$\rho_r = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sqrt{E[(X - \mu_X)^2]E[(Y - \mu_Y)^2]}} = \frac{E[XY] - E[X]E[Y]}{\sigma_X \sigma_Y} \quad \text{Eq 2-6}$$

Pearson's "r" is a measure of the association which varies from -1 to +1, with 0 indicating no relationship (random pairing of values) and 1 indicating perfect relationship, taking the form, "The more the x, the more the y, and vice versa". A value of -1 is a perfect negative relationship, taking the form "The more the x, the less the y, and vice versa".

2.4 Signal Processing

2.4.1 Signal Modeling

One method to extract the main features of the target signal to be detected is by using signal modeling. By modeling the signal several coefficients of the signal are obtained. That is, it is possible to detect the signal by those coefficients.

The model for modeling determinate signals is the unit impulse response of a causal LSI filter (Linear Shift-Invariant) [27]. In this case the input to the filter is the unit impulse function, and thus, the output is a deterministic signal that is completely determined by the LSI filter. Now in order to model a stochastic process, the unit impulse is replaced by white noise as the input of the LSI filter. The Z-transform function of the LSI filter $H(z)$ can be expressed by:

$$H(z) = \frac{B_q(z)}{A_p(z)} = \frac{\sum_{k=0}^q b_q(k)z^{-k}}{1 + \sum_{k=1}^p a_p(k)z^{-k}} \quad \text{Eq 2-7}$$

In the time domain the output $x(n)$ and the white noise input $v(n)$ of the system of $H(z)$ are related by the linear constant coefficient difference equation as follows:

$$x(n) + \sum_{l=1}^p a_p(l)x(n-l) = \sum_{l=1}^q b_q(l)v(n-l) \quad \text{Eq 2-8}$$

Multiplying both sides of equation 2-8 by $x(n-k)$ and taking the expected value:

$$r_x(k) + \sum_{l=1}^p a_p(l)r_x(k-l) = \sigma_v^2 \sum_{l=1}^q b_q(l)h(l-k) \quad \text{Eq 2-9}$$

Because this model is casual, linear and shift-invariant, $h(n)$ is zero for $n < 0$ then equation 2-9 becomes:

$$r_x(k) + \sum_{l=1}^p a_p(l)r_x(k-l) = \sigma_v^2 \sum_{l=k}^q b_q(l)h(l-k) \quad \text{Eq 2-10}$$

This is the Yule-Walker equation.

The more popular form of the Yule-Walker equation [28] is:

$$r_x(k) + \sum_{l=1}^p a_p(l)r_x(k-l) = \sigma_v^2 c_q(k) \quad \text{Eq 2-11}$$

where:

$$c_q(k) = \begin{cases} \sum_{l=0}^q b_q(l)h(l-k); & k < 0 \\ \sum_{l=k}^q b_q(l)h(l-k); & 0 \leq k \leq q \\ 0 & k > q \end{cases} \quad \text{Eq 2-12}$$

The matrix form is:

$$\begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(p) \\ r_x(1) & r_x(0) & \cdots & r_x(p-1) \\ \vdots & \vdots & \cdots & \vdots \\ r_x(q) & r_x(q-1) & \cdots & r_x(q-p) \\ r_x(q+1) & r_x(q) & \cdots & r_x(q-p+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(q+p) & r_x(q+p-1) & \cdots & r_x(q) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \sigma_v^2 \begin{bmatrix} c(0) \\ c(1) \\ \vdots \\ c(q) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{Eq 2-13}$$

AR (All-pole) modeling is somewhat simpler than pole-zero modeling. And AR models are very good for modeling narrow band signals (peaky) [29].

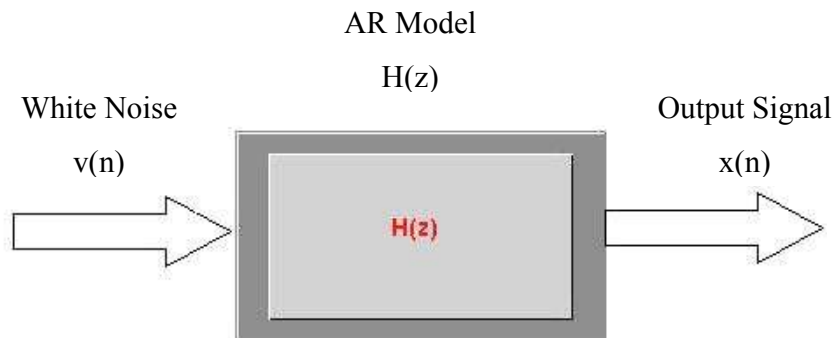


Figure 2-9: AR Model

The general form of the AR model is:

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^p a_p(k)z^{-k}} \quad \text{Eq 2-14}$$

In the time domain:

$$x(n) + a_p(1)x(n-1) + \dots + a_p(p)x(n-p) = b_0v(n) \quad \text{Eq 2-15}$$

The Yule-Walker equation [29] for the AR Model is:

$$\begin{bmatrix} r_x(0) & r_x(1) & \dots & r_x(p) \\ r_x(1) & r_x(0) & \dots & r_x(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(p) & r_x(p-1) & \dots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \sigma_v^2 \begin{bmatrix} c(0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{Eq 2-16}$$

where,

$$c(0) = b(0)h(0) = b(0)^2 \quad \text{Eq 2-17}$$

So,

$$\begin{bmatrix} r_x(0) & r_x(1) & \dots & r_x(p) \\ r_x(1) & r_x(0) & \dots & r_x(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(p) & r_x(p-1) & \dots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \sigma_v^2 \begin{bmatrix} b(0)^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{Eq 2-17}$$

Then, use the last p equations in equation 2-17 to get $a_p(1), a_p(2), \dots, a_p(p)$ and use the first equation in equation 2-17 to get $\sigma_v b(0)$. Normally the input of the AR model is white noise with $\sigma_v = 1$. Finally $b(0)$ is calculated. So all the parameters in the AR model are known.

2.5 Pattern Recognition and Neural Networks

The pattern recognition related activities using the neural network can be broadly grouped into two categories. The first group of activities consists of using the discriminatory self-organizing features of various neural network models, such as multi-layer perceptrons, Kohonen's self-organizing feature maps, etc. to build systems for recognizing different kinds of shapes, sounds and textures. Many such efforts have led to performance levels that are comparable or superior to the existing levels of performance

achieved by traditional pattern recognition methods. The second group of pattern recognition related activities centers around mapping traditional pattern classifiers into ANN architectures. The aim of such mappings is to utilize some of the key features of ANN models to obtain better classification performance in terms of speed or error rate or both. Most of the commonly used classifiers, such as linear classifiers, quadratic classifiers, tree classifiers, nearest neighbor classifiers, have been exactly or approximately mapped into layered neural network architectures.

The neuron is the basic component of all kinds of neural networks. The structure of neuron is shown in Figure 2-10:

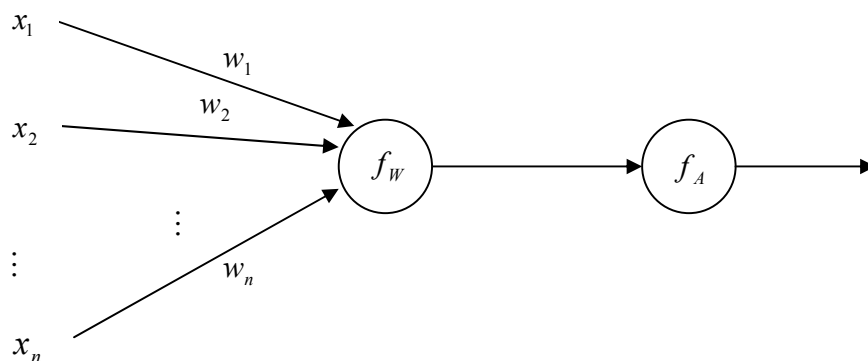


Figure 2-10: Neuron Structure

$X = [x_1, x_2, \dots, x_n]'$ is the input vector. $W = [w_1, w_2, \dots, w_n]$ is the weights of the neuron. f_w is the neuron function with the variants X and Y . Normally f_w is defined as $f_w(X, W) = WX$. f_w can be defined in other forms, for example, f_w is defined as the distance between the input vector and the weight vector in the Radial Basis Networks (RBN), the Self-Organizing Map Networks (SOM) and the Learning Vector Quantization Networks (LVQ). f_A is the activity function or the transfer function. There are three common kinds of the activity function: Hard-Limit Activity Function, Linear Activity Function and Log-Sigmoid Activity Function.

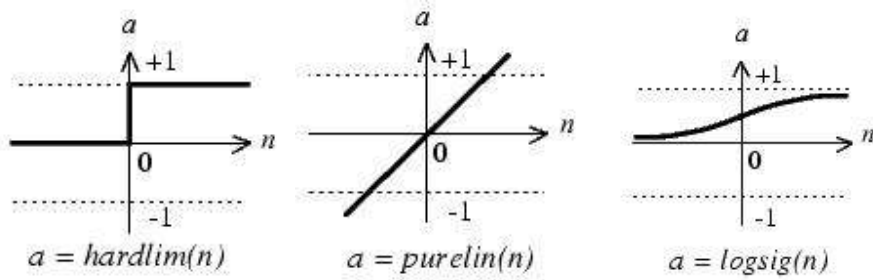


Figure 2-11: Three Activity Functions [30]

The architecture of the single neuron perceptron is shown in Figure 2-12.

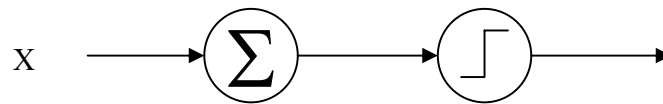


Figure 2-12: Single Neuron Perceptron

The neuron function is $f_w(X, W) = WX = \sum_{i=1}^n x_i w_i$. The activity function is the hard limit activity function.

The architecture of the single layer perceptron is shown in Figure 2-13:

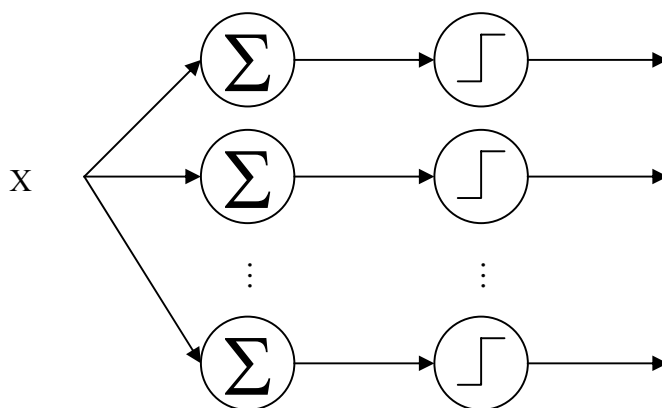


Figure 2-13: Single Layer Perceptron

We can use the perceptron rule to train the perceptron:

$$W_{n+1} = W_n + E_r X' \quad \text{Eq 2-18}$$

where: E_r is the error.

The input vector normalization can improve the performance of the perceptron learning rule. If all the activity functions are the pure line activity functions, the gradient descent method can be used to train the perceptron.

$$W_{n+1} = W_n + \mu_l E_r X' \quad \text{Eq 2-19}$$

Where: μ_l is the learning rate

E_r is the error.

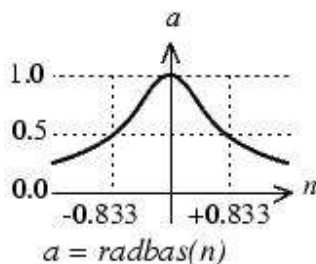
The convergence of the perceptron rule was proven. That is: if the classes used to train the perceptron are linearly separable, the perceptron is convergent with enough training. The weights are the hyperplane between the classes. As for the gradient descent method, the learning rate μ must be small enough to avoid vibrating around the target position. And more, it is possible to stop at the local minimum point, not the global minimum point.

The main limitation of the perceptron is that the perceptron can only classify linearly separable sets of the input vectors. If the input vectors are not linearly separable, training will never reach a point where all input vectors are classified properly. The other limitation is that the training process is too long, especially when the number of the neurons and the number of the training data are larger. The multi-layer neural network can overcome the two limitations of the perceptron.

The multi-layer neural network is the most popular neural network. It is also the feed-forward neural network. The input vector propagates from the input layer, through one or more hidden layers, to the output layer. The error back-propagation method is used to train the network. The gradient descent method is used to train every layer in the network for every training step. It is the reason that the training process is much shorter than that of the single layer neural networks. As mentioned above, the single layer neural

network can only classify the linear separable classes. The multi-layer neural network, if there are enough neurons, can classify any arbitrary shape in any dimension. The application of the multi-layer neural network is much wider than that of the single layer neural network. The last point is that the convergence of the back-propagation algorithm is not proven yet, although there are many successful applications of the multi-layer neural network trained by the back-propagation algorithm.

Radial basis neural network (RBN) is very much different from the neural networks mentioned above. The neuron function of the neuron of RBN is defined as the distance between the input vector and the weight vector. The activity function of the neuron of RBN is Radial Basis Function:



$$radbas(n) = e^{-n^2}$$

Figure 2-14: Radial Basis Function [30]

RBN is a multi-layer neural network:

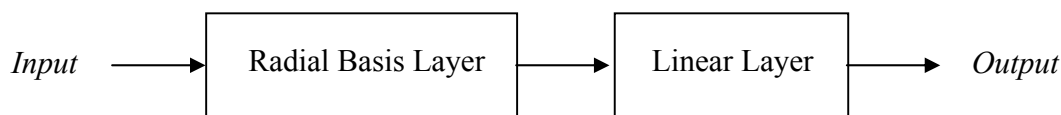


Figure 2-15: Radial Basis Network

The important point of RBN is that the nearer the input vector and the weight vector of one of the neuron in the radial basis layer, the more the output of the neuron. The RBN neuron is the fundament of Generalized Regression Neural Network (GRNN), the Probabilistic Neural Network (PNN), the SOM and LVQ. GRNN is used for function

approximation. PNN can be used for classification. And PNN is guaranteed to converge to a Bayesian classifier if it is given enough training data.

The competitive neural network can learn the distribution of the input vectors. It is formed by adding one competitive neuron to the radial basis neurons. The competitive neuron is a winner-take-all neuron. The radial basis neuron with maximum output wins the competition and outputs 1. All the others output 0. The weights of the winning neuron are adjusted with the Kohonen learning rule. During the training procedure, in order to avoid dead neurons (the weight vectors of the neurons are too far away the input vectors and never win a competition), one more bias is added to the neurons that win the competition rarely. That makes the distance neurons more likely to win in the following training procedure.

Self-organizing map neural network can also learn to classify input vectors according to how the input vectors are distributed in the input space. It differs from competitive layers because the weights of the neighboring neurons of the winning neuron in the SOM are adjusted according to the Kohonen learning rule.

The LVQ network is multi-layer neural network, one competitive layer followed by one linear layer. The competitive layer can learn up to the subclasses as many as the number of the neurons in the first competitive layer. The second linear layer combines the subclasses to the target classes by setting the weights in the second linear layer according the target classes. The LVQ network has the same capability as the multi-layer network trained by the back-propagation algorithm. That means The LVQ network can classify non-linear separable problem. In order to get better performance, the distribution of the input sample vectors must have the same distribution of the target classes.

The Elman network is a kind of recursive network. It is a two-layer network with feedback from the first-layer output to the first layer input. The recurrent connection allows the Elman network have the capability to detect the time-relative patterns.

Successful computation in all kinds of neural networks is dependent on details of the data representation, that is, on how the pattern of input and output unit activation relates to the world. Neural networks are extremely sensitive to representations. In a real sense, the data representation is the mechanism by which networks are programmed. The choice of a good data representation is of far more value toward the solution of a problem than is the choice of the learning rule or network.

In some circumstances, it may be desirable to use the 'raw' data as the input to the network. Many classification problems are difficult to solve using traditional pattern recognition partially because the task of identifying and extracting appropriate feature information is so complex and ill defined. In such cases a neural network may prove more adept at identifying underlying features or data trends than a human analyst. Consequently, there may be an advantage gained from presenting a network with large, unprocessed data vectors and expecting that the adaptive training procedure will be able to identify the underlying information. There is clearly a compromise that must be reached between these two approaches. There are few analytical methods available to assist in the decision process.

The advantages and disadvantages of the neural network compared with the statistical pattern recognition can be listed as:

Advantages of the neural network:

1. In many practical cases, the correct statistical pattern model is unknown. As for the neural network, No need to make assumptions about the distribution densities and can utilize the training data directly in order to determine unknown coefficients (weights) of the decision rule. As for the statistical pattern recognition, the model selected can be far from reality and lead to significant increase in the classification error.
2. While training the neural networks, several statistical classifiers of different complexity can be obtained.
3. In some cases, the single layer perceptron, the multi-layer perceptron and the radial basis function based classifiers can have very good small training-set size

properties. However, Complex statistical classification algorithms require estimating a large number of parameters from the training data. Therefore, suffer from the curse of dimensionality.

4. The information contained in almost correct initial weights of the single layer perceptron classifier can be saved if the training process is stopped optimally.

Disadvantages of the neural network:

1. The architecture of the network and the training parameters must be chose in advance.
2. The optimal algorithm to stop training must be chose.
3. Except for some common neural networks, the relationships among the performance, the complexity and the training size are unknown.
4. The training process is very slow normally and is affected by weights initialization and the property of the data.
5. The learning algorithm can be divided to three categories: the supervised leaning, the un-supervised learning and reinforcement learning. It seems that most of the un-supervised learning are kinds of the statistical methods. The neural network seems lack the capability to deduce.

2.6 Classification Tree

Tree-based methods for classification have traditionally been popular in fields like biology and the medical sciences and have the advantage of making the classification procedure easy to comprehend. Classification trees are essentially symbolic systems that assign symbolic decisions to samples and are built upon attributes of the patterns that are symbolic in nature or, at least, discretized if they are not symbolic to begin with [31]. The terminology associated is graph-theoretic, the root being the top node, and samples are passed down the tree. At each node decisions are made, until a terminal node, or leaf, is reached. A question is associated with each non-terminal node, based on the answer of which a split is made. The label of a classification is contained in each leaf. A classification tree partitions the pattern space (the feature space) into sub regions

corresponding to its leaves, and each unknown pattern is classified by the label of the leaf it reaches. It is not difficult to visualize that a classification tree provides a structured or hierarchical description of the knowledge base. The commonest type of tree-structured classifier is the binary tree classifier in which each non-terminal node has exactly two children.

The most important aspect of tree-based classification is the automatic construction of trees from a set of samples. Many algorithms are available. The main differences among the algorithms for tree construction lie in the rule they use to split nodes and prune the tree. Pruning the tree means removing the redundant subtrees.

A classification tree contains a root node, zero or more internal nodes, (all nodes except the root and the leaves), and one or more leaf nodes (terminal nodes with no children). For a binary decision tree, the root node and all internal nodes have two child nodes. All non-terminal nodes contain splits.

The main components of the classification tree include:

1. Nodes - tests of attributes
2. Branch - one of possible values for an attribute
3. Leaves (terminal nodes) classifications
4. Path (from the tree root to a leaf) - conjunction of attribute tests

The most important steps in building a good classification tree include splitting the tree at suitable position with suitable value and pruning the classification tree to reduce the judgment complexity and keep the error rate under a certain level at the same time.

2.6.1 Splitting Algorithm

The most common splitting algorithm is the information gain algorithm. Let us use x_i ($i = 1, 2, \dots, n$) denote the n samples and c_j ($j = 1, 2, \dots, m$) as the m classes. Every sample belongs to one of the m classes. More over, p_j is used to stand for the probability of class j in the sample collection.

The entropy (E^t) of the sample collection is

$$E^t = -\sum_{q=1}^j p_q \ln p_q \quad \text{Eq 2-20}$$

The binary number is used in computer. $-\ln p_q$ bits are needed in order to denote the class with probability p when the class happens. The expected value of bit number is $p \ln p$, the entropy value.

The entropy function will get the highest value when all the classes are present equally in the collection. And it will get zero when all the collection samples belong to the same class [32]. That means that the purer the collection is, the less the entropy function value. The entropy function value can be seen as the degree of the collection purity or impurity. When the collection is split into two sub-collections, left collection and right collection, define:

$$\Delta E^t = E^t - p_{left} E^t_{left} - p_{right} E^t_{right} \quad \text{Eq 2-21}$$

where:

$$p_{left} = \frac{n_{left}}{n} \quad \text{Eq 2-22}$$

$$p_{right} = \frac{n_{right}}{n} \quad \text{Eq 2-23}$$

$$n = n_{left} + n_{right} \quad \text{Eq 2-24}$$

E'_{left} and E'_{right} are the entropy values of the left collection and the right collection respectively.

Therefore, from equation 2-20 to 2-24:

$$\Delta E^t = -\sum_{q=1}^j p_q \ln p_q - \frac{n_{left}}{n} \left(-\sum_{q=1}^j p_{left}^q \ln p_{left}^q \right) - \frac{n_{right}}{n} \left(-\sum_{q=1}^j p_{right}^q \ln p_{right}^q \right) \quad \text{Eq 2-25}$$

It can be proved that:

$$0 \leq \Delta E^t \leq E^t$$

The more the ΔE^t , the better the splitting. The number of all the possible ΔE^t is $2n$. So all the possible ΔE^t can be calculated and the splitting with the largest ΔE^t is chosen [33].

This algorithm is suitable for not only binary splitting but also splitting with more leafs (sub-nodes). The difference is that more ΔE^t need to be calculated. For splitting with i leafs, the number of needed to calculate is i^n . When $i = n$ that means every sample is a leaf, ΔE^t will be maximum, but there is no practical signification. The choose of leaf number i is dependent on the known sample character. Normally i is chose to be 2, which means binary splitting.

Gini Diversity Index is another algorithm in information gain, which is similar to the one above. The only difference is that in Gini Diversity Index the impurity degree is defined as:

$$G_i = 1 - \sum_{q=1}^j p_q^2 \quad \text{Eq 2-26}$$

As for Gini Diversity Index and entropy, a larger mount of empirical tests were conducted. No conclusive results were found. Practically, Gini Diversity Index is better than entropy.

Another algorithm is Twoing Rule. This algorithm defines the impurity same as entropy, but defines the ΔE^t in a different way:

$$\Delta E^t = \frac{p_{left}}{p_{right}} \left(\sum_{q=1}^j |p_{left}^q - p_{right}^q| \right) \quad \text{Eq 2-27}$$

Twoing Rule can only be used in binary splitting.

2.6.2 Stopping Splitting Algorithm

One way to control splitting is to allow splitting to continue until all terminal nodes are pure or the entropy value below certain threshold.

The second way to control splitting is to allow splitting to continue until all terminal nodes are pure or contain no more than a specified minimum number of samples.

Another way to control splitting is to allow splitting to continue until all terminal nodes are pure or contain no more samples than a specified minimum fraction of the sizes of one or more classes.

All algorithms need certain knowledge about the samples. If the information is insufficient to set such thresholds, the tree will be split until every node is pure. This normally will result in a huge, complex tree, i.e. over-fitting [34].

2.6.3 Pruning Algorithm

This algorithm is a cost-complexity pruning algorithm. This means that the algorithm considers cost (error rate) and structure complexity.

The resubstitution error of tree T is the fraction of samples in the training sample that are misclassified by tree T . The resubstitution error of tree T is denoted by $R(T)$.

The total cost function of tree T is defined as:

$$C_\alpha(T) = R(T) + \alpha \tilde{T} \quad \text{Eq 2-28}$$

Where:

\tilde{T} is the number of terminal leafs in tree T

$\alpha > 0$

If α is fixed, there must be a smallest subtree $T(\alpha)$ of T_{\max} (T_{\max} is the largest tree.) that fulfills the following conditions:

1. $C_\alpha(T(\alpha)) = \min_{T < T_{\max}} C_\alpha(T)$
2. if $C_\alpha(T) = C_\alpha(T(\alpha))$ then $T(\alpha) < T$

where:

$T(\alpha) < T$ means tree $T(\alpha)$ is the subtree of tree T , tree $T(\alpha)$ can be obtained by pruning some nodes and leafs from Tree T .

When α goes from 0 to 1, there will be a finite number of subtrees of tree T_{\max} . The serial trees is denoted as:

$$T_{\max} > T_1 > T_2 > \dots > \{t_1\} \text{ when } \alpha = 0 \rightarrow \infty$$

where:

$\{t_1\}$ means the smallest tree with only the root node.

'>' means that the next tree in the sequence can be obtained by pruning the one before.

In order to get the trees T_i ($i = 1, 2, \dots, k$) and the corresponding α_i ($i = 1, 2, \dots, k$), see the Figure 2-16:

When $\alpha < \alpha_i$ the tree T_i is the tree with the smallest cost function value. When $\alpha > \alpha_i$ the tree T_{i+1} is the tree with the smallest cost function value.

Let T_p denote the tree pruned (the tree drawn with dot line) and $R(T)$ denote the resubstitution value of the root node of tree pruned.

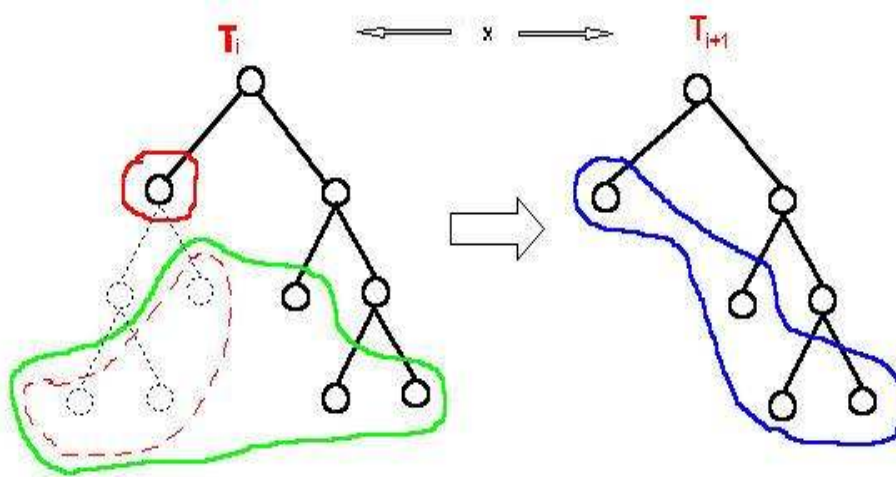


Figure 2-16: Tree Pruning

The cost function value of T_i is:

$$C_\alpha(T_i) = R(T_i) + \alpha \tilde{T}_i \quad \text{Eq 2-29}$$

The cost function value of T_{i+1} is:

$$C_\alpha(T_{i+1}) = R(T_{i+1}) + \alpha \tilde{T}_{i+1} \quad \text{Eq 2-30}$$

From Figure 2-16:

$$R(T_{i+1}) = R(T_i) - R(T_p) + T(t) \quad \text{Eq 2-31}$$

When $\alpha = x$

$$C_\alpha(T_i) = C_\alpha(T_{i+1}) \quad \text{Eq 2-32}$$

From equations 2-29 to 2-32, solve x :

$$x = \frac{R(t) - R(T_p)}{\tilde{T}_p - 1} \quad \text{Eq 2-33}$$

By the algorithm above, the trees T_i ($i = 1, 2, \dots, k$) and the corresponding α_i ($i = 1, 2, \dots, k$) can be found.

The last step is using 1-SE rule to select the final tree.

Let R_{\min} denote the minimum resubstitution error in the R_i ($i = 1, 2, \dots, k$) corresponding T_i ($i = 1, 2, \dots, k$).

(SE) value can be calculate by:

$$(SE) = \sqrt{\frac{R_{\min}(1 - R_{\min})}{n}} \quad \text{Eq 2-34}$$

where: n is the number of test samples.

The 1-SE rule is to select the smallest tree T_i which resubstitution error

$$R_i < R_{\min} + (SE) \quad \text{Eq 2-35}$$

When the number of the samples is relatively small, there is not the test samples to do the pruning algorithm. A way to avoid this problem is to use cross-validation [35].

This method divides the total samples into several sub-groups:

$$S = \{S_1, S_2, \dots, S_m\}$$

use $S - S_i$ $i \in (1, 2, \dots, m)$ to calculate tree T_k^i and corresponding error R_k^i , and then use

$$R_k = \sum_{i=1}^m R_k^i$$

to get R_k . Finally, use 1-SE rule to select the final tree.

2.7 Image Segmentation Methods

2.7.1 Watershed Segmentation

The watershed segmentation was widely and successfully applied in different domains as a powerful segmentation tool. The algorithm is simple: Suppose that a hole is punched in each regional minimum and the entire topography is flooded from below by letting water rise through the holes at a uniform rate. When the rising water in distinct basins is about to merge, a dam is built to prevent the merging. The flooding will eventually reach a stage when only the top of the dams are visible above the water line. These dam boundaries correspond to the divide lines of the watersheds. The principal application of watershed segmentation is in the extraction of nearly uniform objects from the background.

2.7.2 Active Contours

Active contours, or snakes, were first introduced by Kass, Witkin and Terzopoulos [66]. An active contour is a curve that moves inside the image pulled by internal forces originating from the curve itself and external forces generated from the image data. The internal and external forces are designed to attract the contour to the desired features in the image while maintaining the contour's smoothness. The traditional contour is a curve $\vec{v}_s = [v_x(s), v_y(s)]$, $s \in [0,1]$ that minimizes the energy function:

$$E_{snake} = \int_0^1 \frac{1}{2} (\alpha |\vec{v}'(s)|^2 + \beta |\vec{v}''(s)|^2) + E_{ext}(\vec{v}(s)) ds \quad \text{Eq 2-36}$$

where α and β are weighting parameters that control the snake's tension and rigidity, respectively. $\vec{v}'(s)$ and $\vec{v}''(s)$ denote the first and second derivatives of $\vec{v}(s)$ with respect to s . The external energy function $E_{ext}(\vec{v}(s))$ is derived from the image so that it takes on its smaller values at the features of interest, such as boundaries.

A curve that minimizes E_{snake} must satisfy the Euler-Lagrange equation:

$$\alpha \bar{v}'(s) + \beta \bar{v}''(s) + \nabla E_{ext} = 0 \quad \text{Eq 2-37}$$

It is:

$$\alpha v'_x + \beta v''_x + \frac{\partial E_{ext}}{\partial x} = 0$$

$$\alpha v'_y + \beta v''_y + \frac{\partial E_{ext}}{\partial y} = 0$$

It may be viewed as a force balance equation that can be divided into internal force and external force:

$$\bar{F}_{int} + \bar{F}_{ext} = 0 \quad \text{Eq 2-38}$$

where $\bar{F}_{int} = \alpha \bar{v}'(s) + \beta \bar{v}''(s)$ and $\bar{F}_{ext} = \nabla E_{ext}$. There are two parts in the internal force. One is elastic force: $\alpha \bar{v}'(s)$. The elastic force will pull the snake into a smooth oval. The most outlying points get pulled fastest. The other is bending force: $\beta \bar{v}''(s)$. The bending force will smooth the active contour, but will not make it contract. If the parameter β is increased, the snake will resist bending, much like a metal ribbon. If β is zero, the snake will be allowed to form sharp corners. The external force, ∇E_{ext} , will attract the snake to the desired image features. For segmentation problem, the borders among the different parts need to be identified. The edge map of original image is used as the external force. The snake will be attracted to the edges under such kind of external force.

There are two key difficulties with traditional active contour algorithm. First, the initial contour must be approach the true boundary closely. Otherwise, it will likely converge to the wrong position. Second, the traditional active contour has difficulty to progress into concave boundary regions. The concave region boundary adds forces to the snake in opposite direction. The active contour is tensed and its motion will not be affected by the internal force any more.

Xu and Prince[65] proposed a new external force field, which is defined as Gradient Vector Flow(GVF) field. The GVF field is formulated as, $\bar{g}(x, y) = [p(x, y), q(x, y)]$ that minimizes the energy function

$$E_{GVF} = \iint \mu(p_x^2 + p_y^2 + q_x^2 + q_y^2) + |\nabla f|^2 |\bar{g} - \nabla f|^2 dx dy \quad \text{Eq 2-39}$$

where f is the edge map derived from the original image.

The first item corresponds to an equal penalty on the divergence and curl of the vector field. The main advantages of the GVF field are:

- To capture the active contour from a long range
- To force it into concave regions.

There are two additional factors that are applied in active contours. One is the change rate of the snake: $\frac{d\vec{v}}{dt}$. It can control the velocity of the snake. The other is arbitrary force, expanding force or shrinking force. The arbitrary force is needed when only expanding or shrinking is expected.

2.8 Chapter Summary

Ultrasonic wave knowledge used for ultrasonic testing is introduced. The Neural networks and statistic pattern recognition methods both have advantages and disadvantages. Which method is better depends on the specific application. In this project, the Classification Tree is applied because of its tolerance to noise and simplicity. The algorithm of Classification Tree is elucidated in detail. The experiment result of Classification Trees is quite good (See Chapter 4 for the details). Image segmentation algorithms: watershed and active contours are introduced. The two algorithms will be used in chapter 5 to check the weld point.

Chapter 3 Signal Analysis & Enhancement

Preprocessing is the first step after the data acquisition. To some extent, preprocessing decides the performance of the automatic system. It is necessary to carry out a thorough study on the ultrasonic signal before carrying on the project. Matlab programs using signal-processing technologies [36] [37], such as FFT, SFFT, auto-correlation, and cross-correlation, are developed to do the signal analysis. Three possible methods to enhance the signal (cross-correlation filter, zero-phase filter and average filter) are compared based on the signal frequency character.

3.1 Signal Property

Below is the signal from the ultrasonic card: SOFRA4001H using a 5MHz compression wave probe placed on a flat aluminum specimen of 25mm thickness.

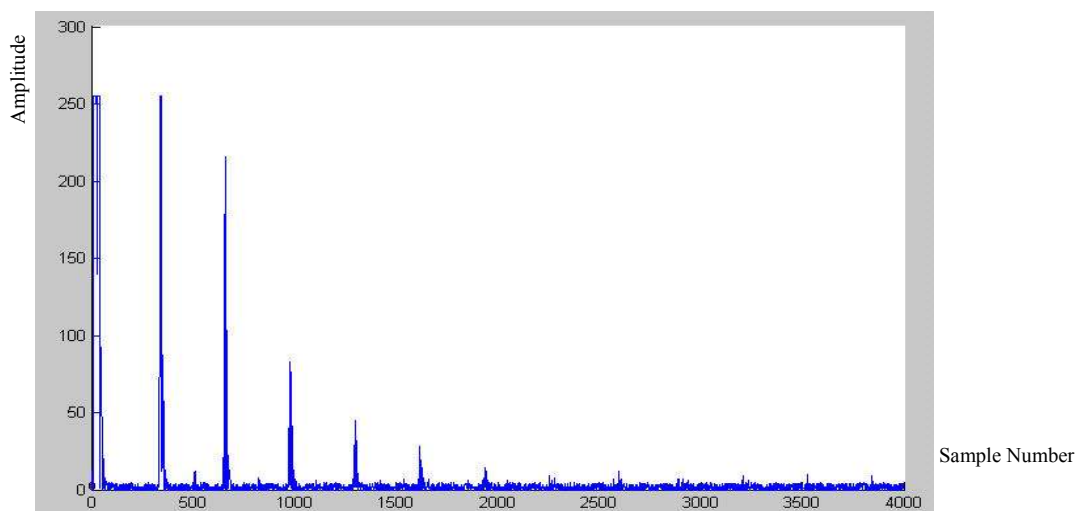


Figure 3-1: Signal from SOFRA4001H Card (Half)

The A/D converter on the card is 8 bits with a 40 MHz sample frequency. So the maximum amplitude is 255. The buffer for the signal is 8 K bytes. The digital signal is rectified. Rectification will blur the frequency character of the signal. This is one of the hardware limitations. Non-rectified signal cannot be got from the card, so all the analysis has to be based on the rectified signal. Figure 3-1 shows the signal with half length (4k). Figure 3-2 is the zoom-in image of Figure 3-1 at position 1600. More details about the signal can be seen.

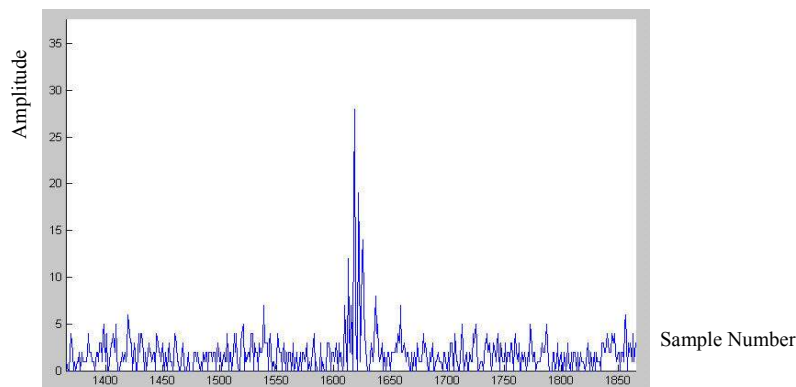


Figure 3-2: Signal Detail

When a 5MHz probe is used, the signal can be simulated in the following way:

First, a cosine signal and an arbitrary profile function are built. Then multiply the two signals to get the simulated signal. The frequency information of the simulated signal can be obtained by the FFT of the simulated signal. The magnitude and phase are displayed together. As expected, the frequency components mostly focus on about 5MHz. The shape of the FFT is decided by the following:

1. Cycles of Signal: The more the cycles, the more the energy of 5MHz. Because of the profile function, noise and attenuation of sound wave, the weak signal maybe has only one or two signals, or even less. This causes the shape of FFT to be more flat. Cycles of signal can be decided by the active pulse duration.
2. Profile Function: The profile function is decided by the active pulse and the damping of the probe. The profile function varies with time because of the attenuation of sound wave. The flatter the profile the sharper the FFT.
3. Sample Frequency: From the sense of signal processing, it is always right that the higher the sample frequency, the better [38]. More frequency information can be kept with higher sample frequency, which means that more kinds of ultrasonic probes can be used in testing, such as 10MHz, 20MHz, and 30MHz. The disadvantage of higher sample frequency is that a larger buffer is needed to cover the same range than a lower sample frequency. Another disadvantage is that the time needed to analyze the sample data will be longer.

4. Noise: For the weaker signal, the effect of noise is relatively obvious and notable.

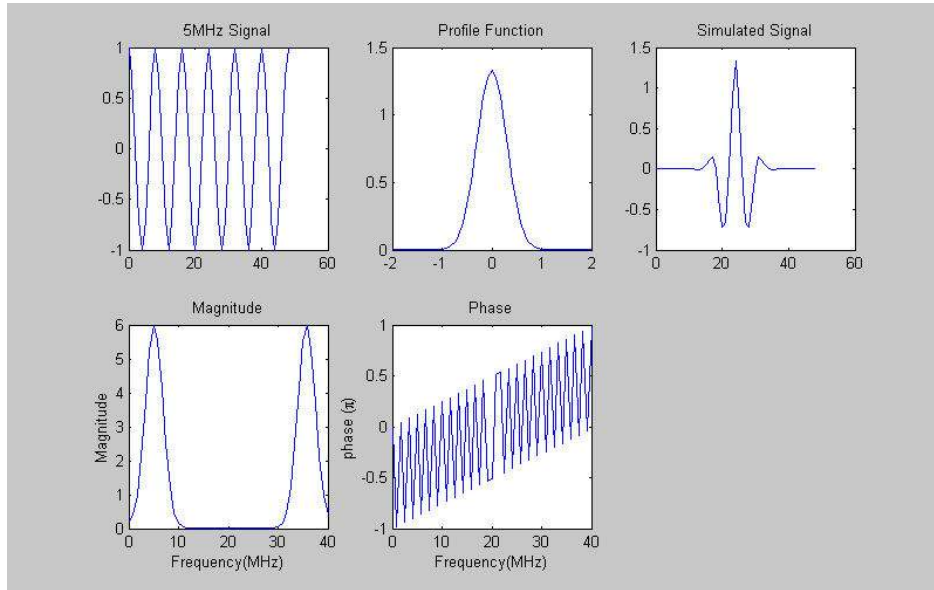


Figure 3-3: Simulated Signal with 6 Cycles and its FFT

Because the output digital signal of the SOFRA4001H is rectified, it is reasonable to take the absolute value of the cosine signal. The result is shown in Figure 3-4. The most distinguishable difference between the FFT of the rectified signal and the FFT of the non-rectified signal is the peak in the middle of the FFT magnitude picture.

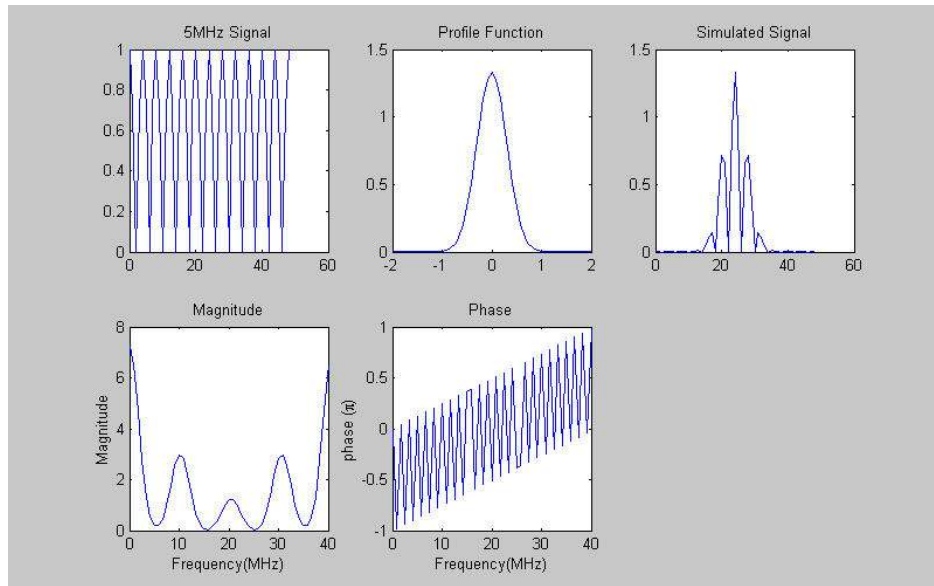


Figure 3-4: Rectified Simulated Signal with 6 Cycles and its FFT

If the sample frequency is higher, the FFT magnitude is shown in Figure 3-5. There are ripples in the middle of the FFT magnitude. Those ripples result from the rectification.

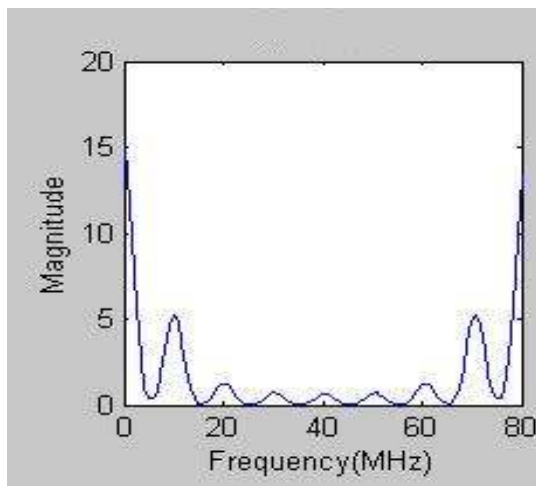


Figure 3-5: FFT with 80MHz Sample Frequency

How does the signal look like practically? Figure 3-6 shows the actual signal output from the ultrasonic card and the FFT of the actual signal. The result is consistent with the result obtained from the simulated signal perfectly.

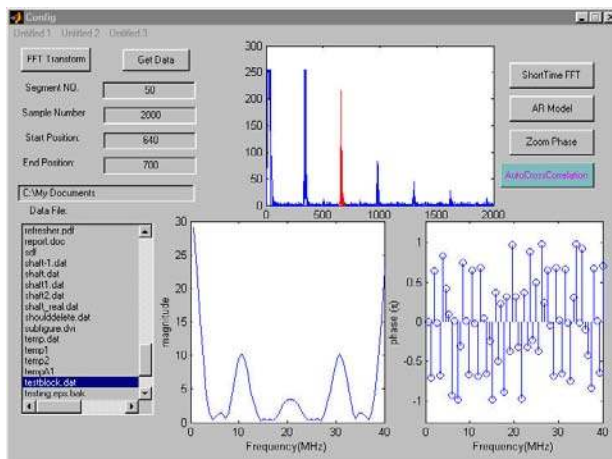


Figure 3-6: Actual Signal and its FFT

In practice, the peak signals are different from each other. Some are with higher amplitude. Because the A/D converter is only 8 bits, any value exceeding $2^8(255)$ will be truncated to 255. Some are with lower amplitude. Others with lowest amplitude are too weak and it is hard to tell those signals from the noise.

In order to understand the frequency character of the signals with different amplitudes, the FFT is done to those kinds of signals. Figure 3-7 to Figure 3-15 show the result.

Compare all the pictures, the FFT of the truncated signals are not with the character of the typical signals with amplitude about 10%-90% full screen height because of truncation. The signals with too low amplitude also are not with the character of the typical signals with amplitude about 10%-90% full screen height because the noise is too high, relatively and the cycles of the signal are too few.

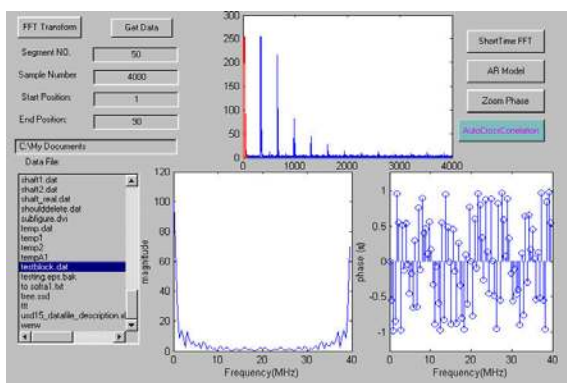


Figure 3-7: The initial Signal (truncated) and its FFT

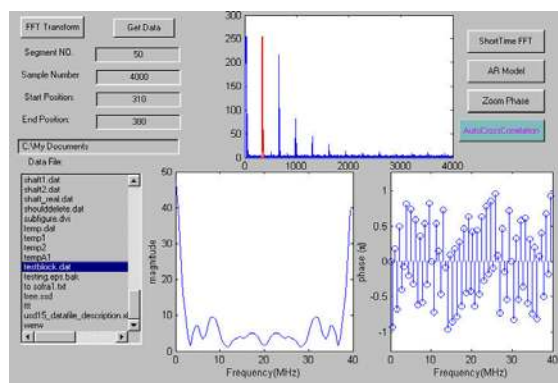


Figure 3-8: The 2nd Signal (non-truncated) and its FFT

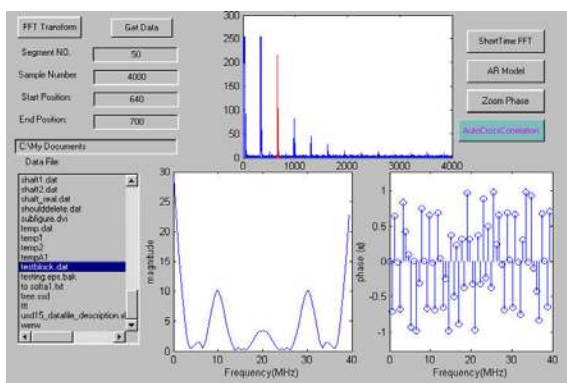


Figure 3-9: The 3rd Signal (truncated) and its FFT

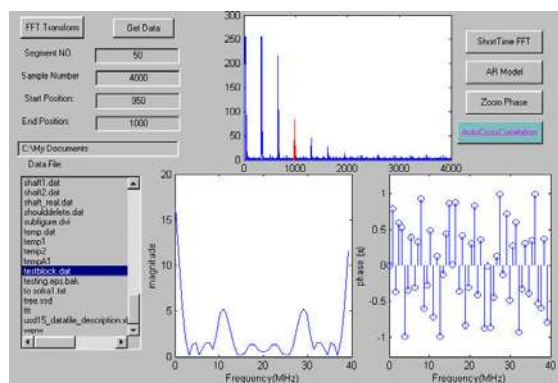


Figure 3-10: The 4th Signal (non-truncated) and its FFT

Chapter 3: Signal Analysis and Enhancement

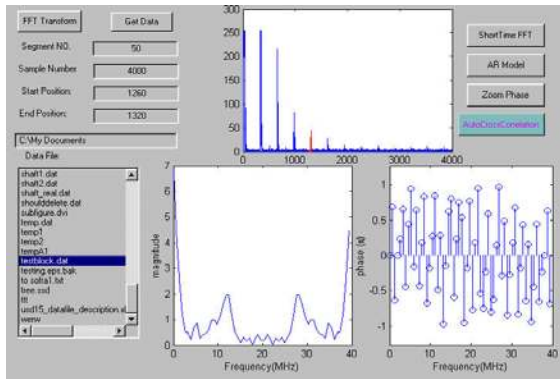


Figure 3-11: The 5th Signal (non-truncated) and its FFT

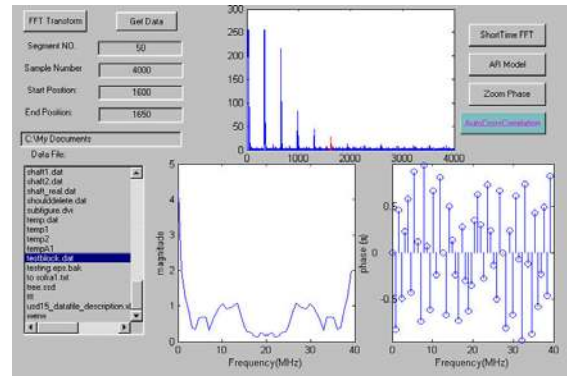


Figure 3-12: The 6th Signal (non-truncated) and its FFT

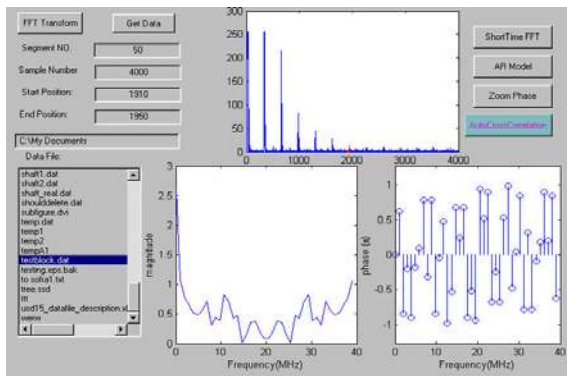


Figure 3-13: The 7th Signal (non-truncated, low amplitude) and its FFT

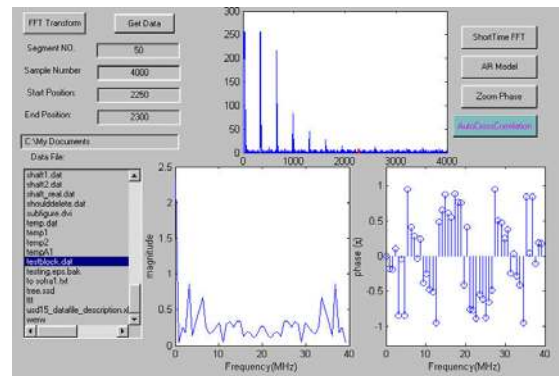


Figure 3-14: The 8th Signal (very low amplitude) and its FFT

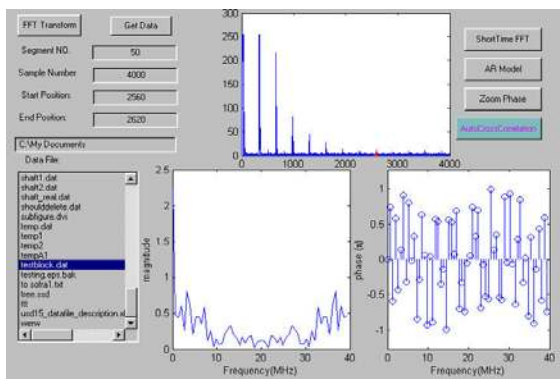


Figure 3-15: The 9th Signal (very low amplitude) and its FFT

To make the result shown on one picture, it is necessary to take the Short Time FFT of the whole output signal. First, the segment length is set to 40. The X-axis is frequency, from 0 to 40MHz (because the sample frequency is 40MHz). The Y-axis is the start position of every 40-data-long segment. This is shown in Figure 3-16. Figure 3-17 is the same as Figure 3-16. The only change is that the different FFT values are shown with different colors in Figure 3-16. The FFT value is shown along the Z-axis in Figure 3-17.

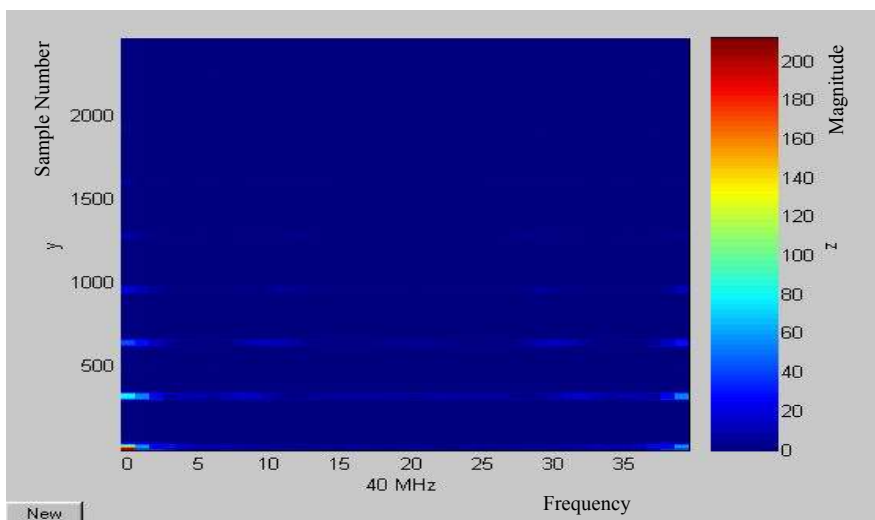


Figure 3-16: SFFT of signal (Segment Length=40, Top view)

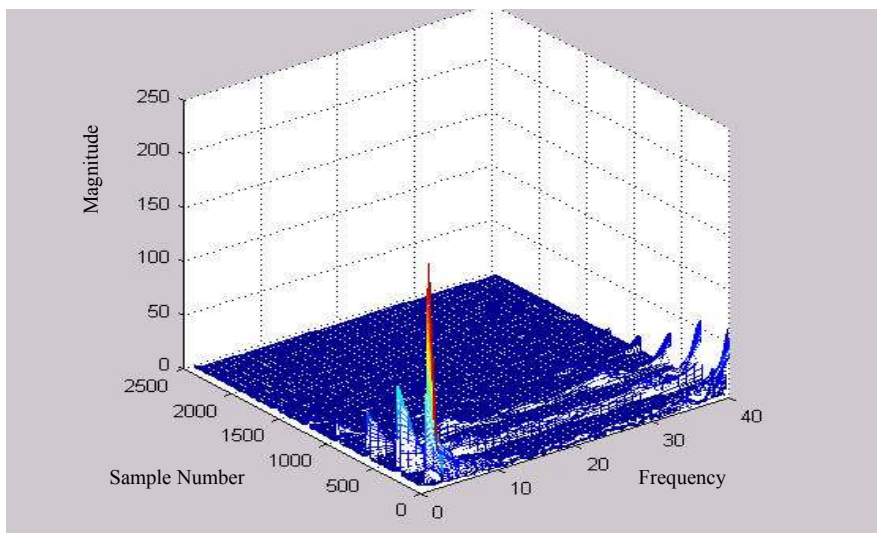


Figure 3-17: SFFT of signal (Segment Length=40, ISO metric)

There are two sets of FFT values should be paid attention to. One set is the FFT values when frequency is equal to 0, the other set is the FFT values when frequency is equal to 10MHz.

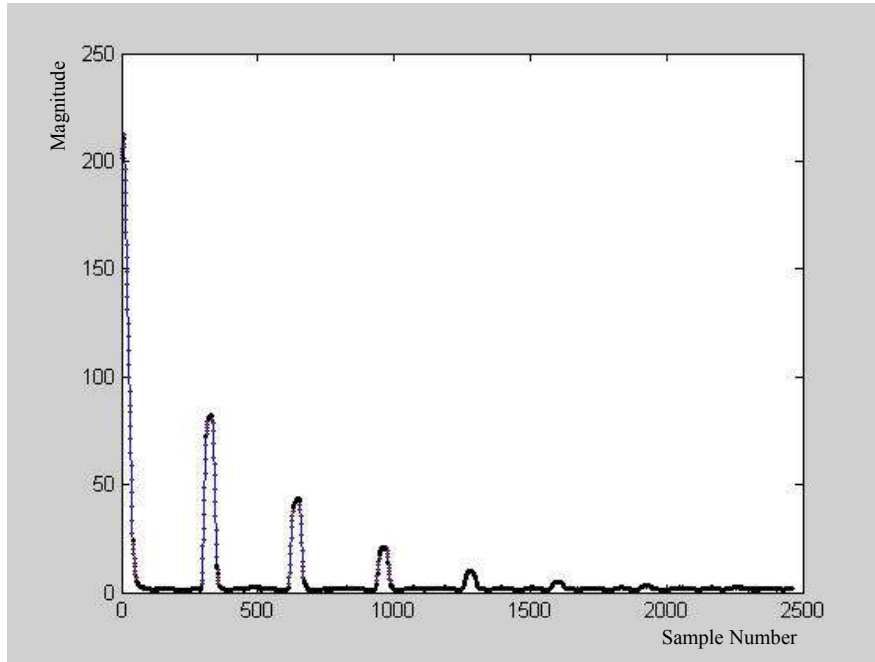


Figure 3-18: FFT Value (Frequency =0 MHz, Segment Length=40)

Figure 3-18 shows the FFT value with frequency equal to 0. The peaks in Figure 3-18 are relatively smooth. That is reasonable. According to the discrete FFT equation:

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad \text{Eq 3-1}$$

because $N=40$ and $k=0$, so

$$X[0] = \frac{1}{40} (x_0 + x_1 + \dots + x_{39}) \quad \text{Eq 3-2}$$

That is the average energy of the segment. Or it is the filtered signal after the original signal pass through a Low Pass filter. By this method a smooth peak can be got. However, the peaks with low amplitude will be blurred compared to the originals, such as the 7th, 8th, and 9th peaks.

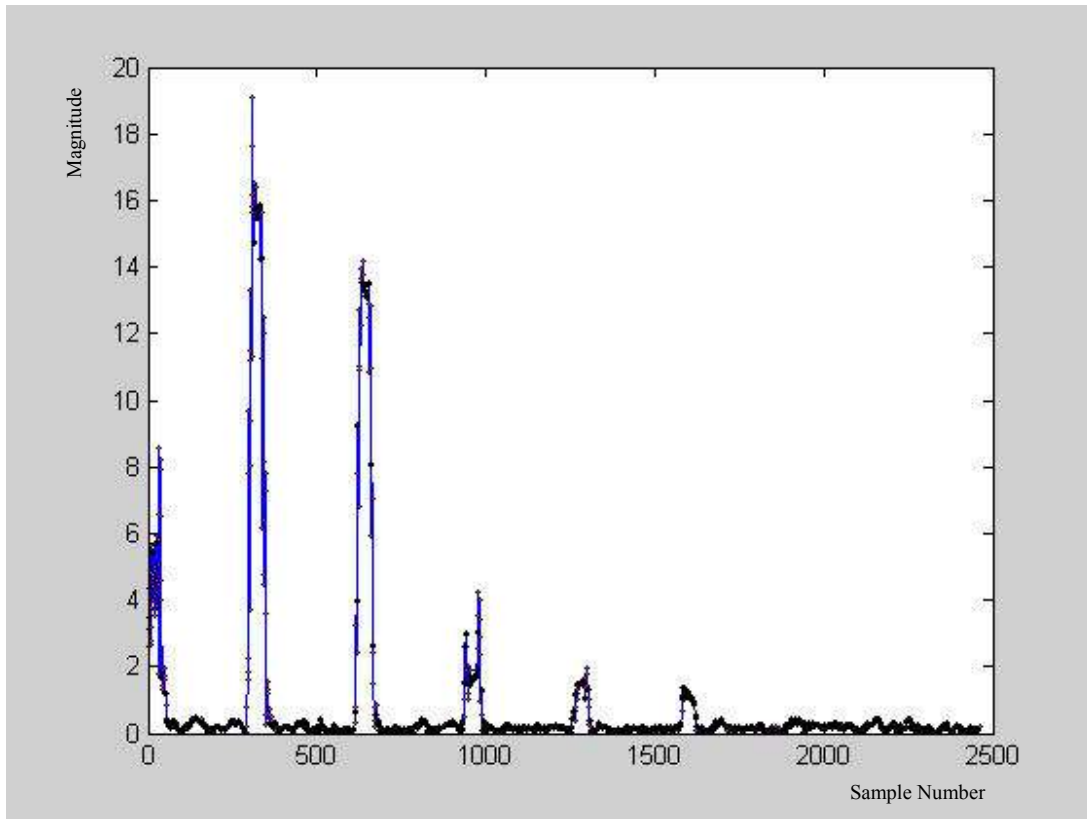


Figure 3-19: FFT Value (Frequency=10MHz, Segment Length=40)

Figure 3-19 shows the FFT value with frequency equal to 10MHz. From the analysis on page 42 and page 39, the reason that Figure 3-19 looks like this can be understood.

Figure 3-20 and Figure 3-21 are the Short Time FFT with data segment length equal to 20. Figure 3-22 and Figure 3-23 are the data sets with frequency equal to 0 and 10 MHz respectively.

The data segment length represents the resolution of the Short Time FFT on time. The shorter the data segment length, the higher the resolution. That means that the signals close to each other can be discriminated. That is the reason that Figure 3-22 is much sharper than Figure 3-18.

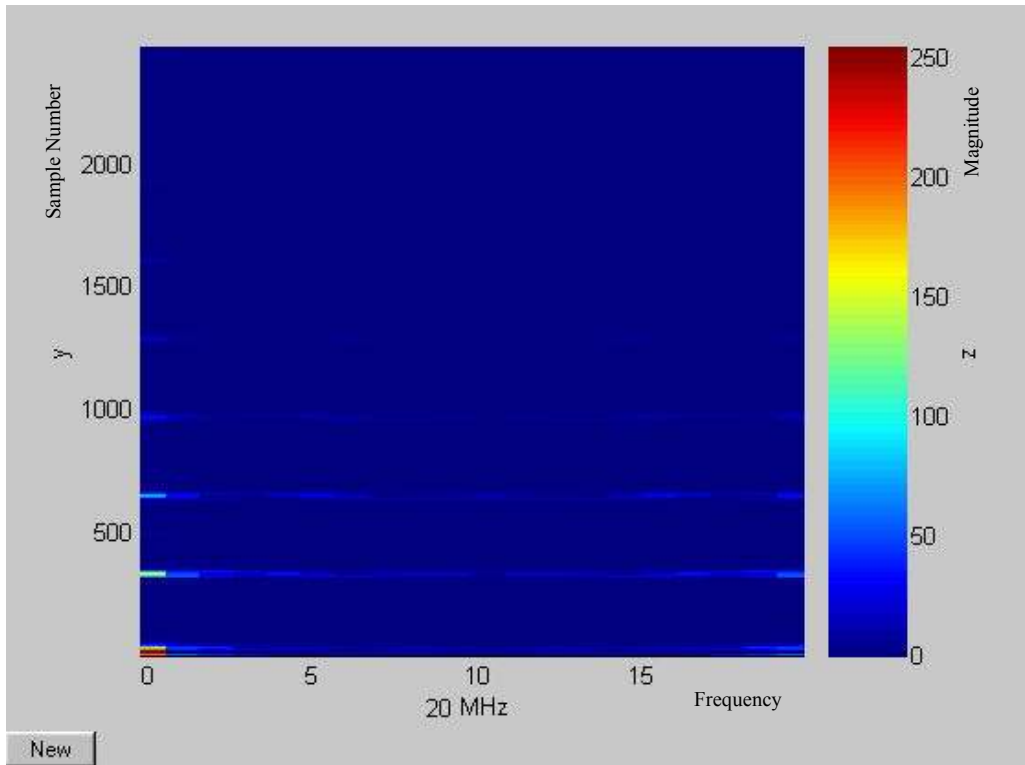


Figure 3-20: SFFT of signal (Segment Length=20, Top view)

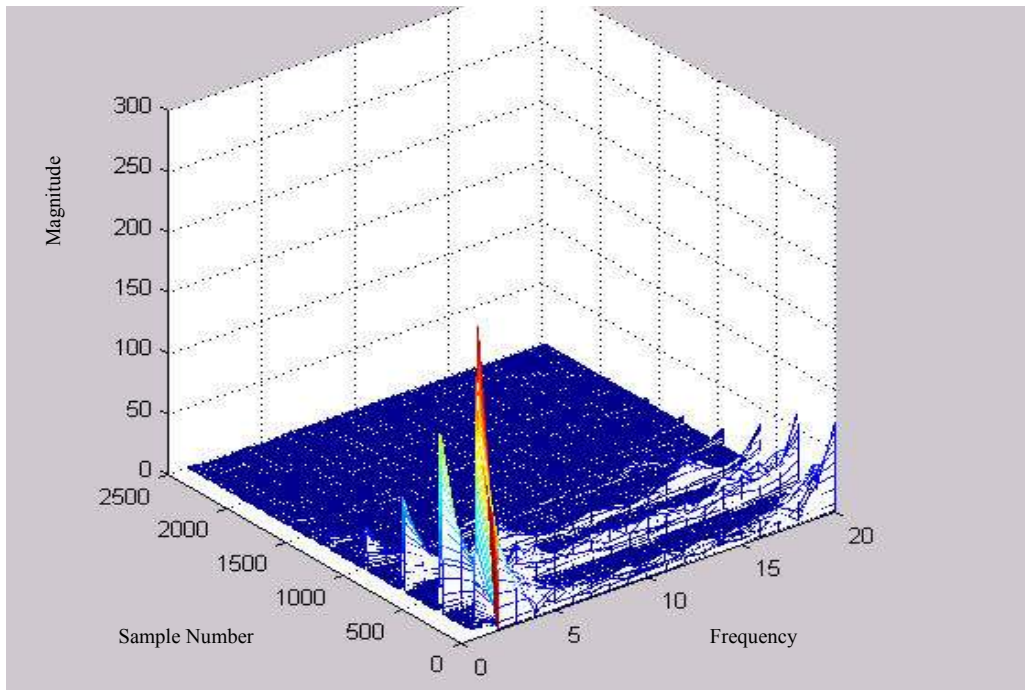


Figure 3-21: SFFT of signal (Segment Length=20, ISO metric)

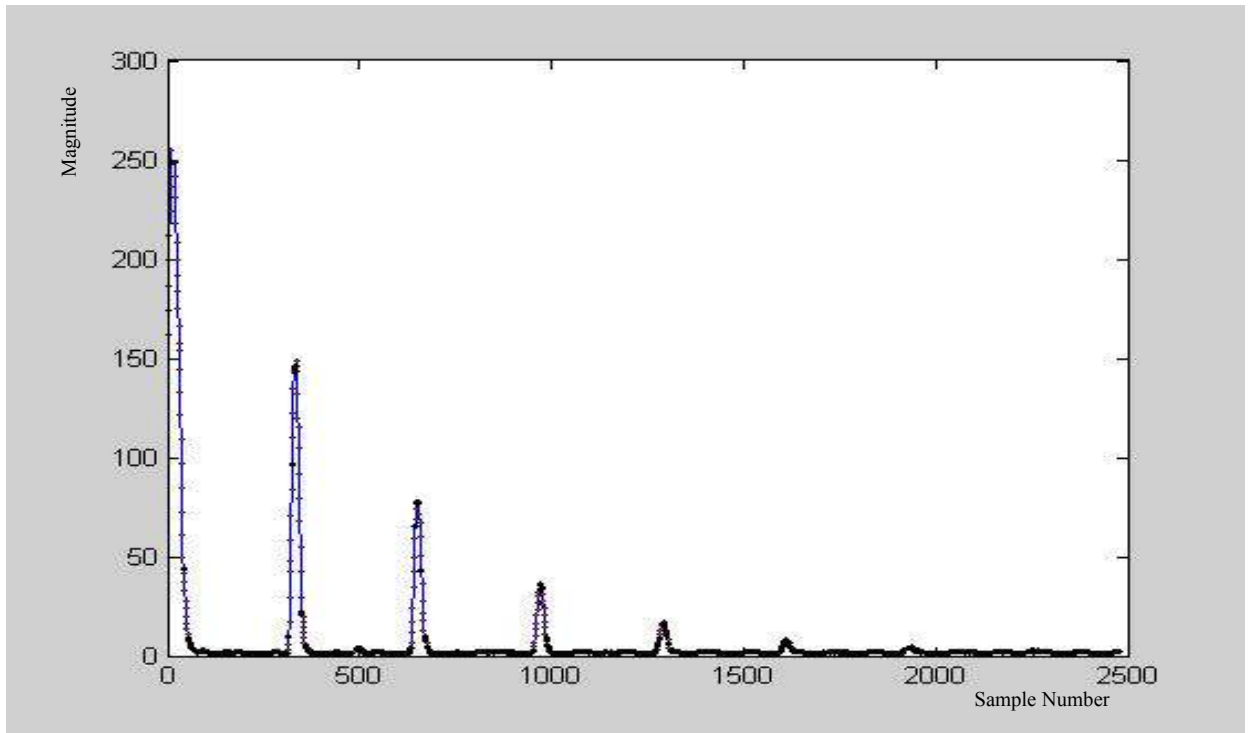


Figure 3-22: FFT Value (Frequency=0MHz, Segment Length=20)

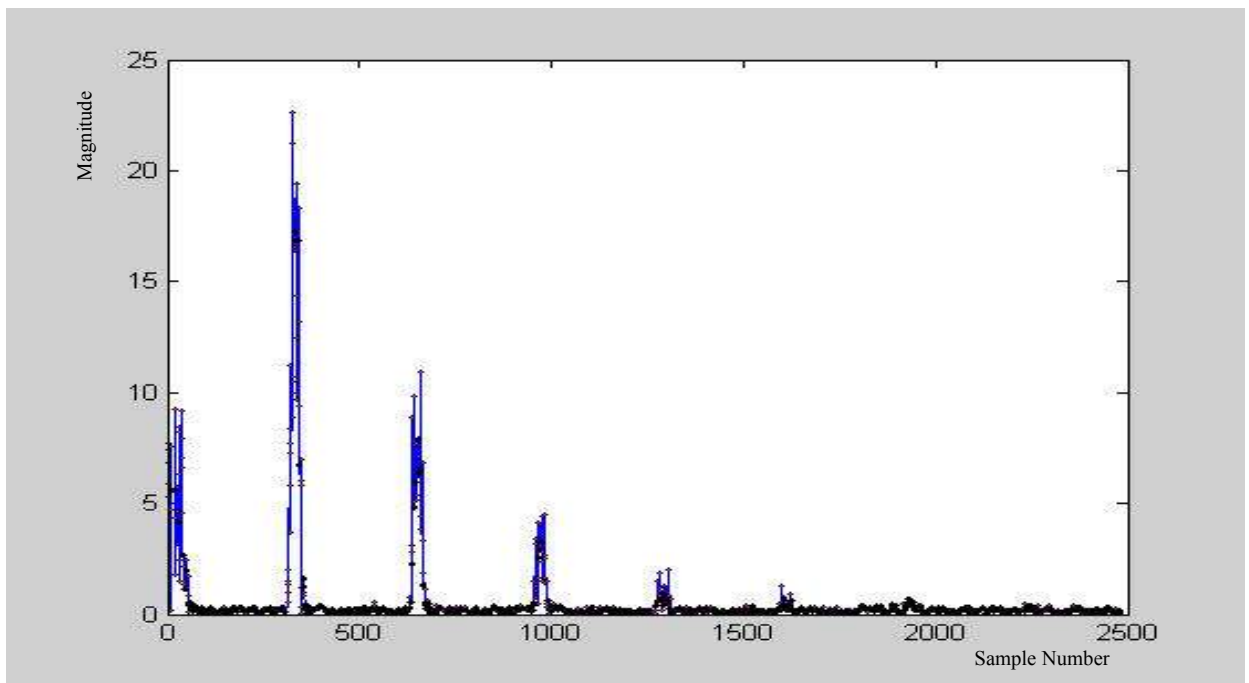


Figure 3-23: FFT Value (Frequency=10MHz, Segment Length=20)

3.2 Enhancement by Cross-Correlation

In digital signal processing, it is common to use cross-correlation to detect a signal or reduce noise [39].

This method is shown in Figure 3-24. The first row shows the signal x and the cross-correlation x and x (autocorrelation). The second row shows the white noise v , signal x and the cross-correlation v and x . The third row shows the blend signal $(v+x)$, signal x , and the cross-correlation of $(v+x)$ and x . The cross-correlation of the ideal white noise signal and signal x should be zero. So by cross-correlation the weak signal with high noise can be enhanced effectively.

The premise of the cross-correlation includes:

1. The noise is white noise.
2. The sample set is large enough, the sample frequency is high enough.
3. The target signal is known exactly.

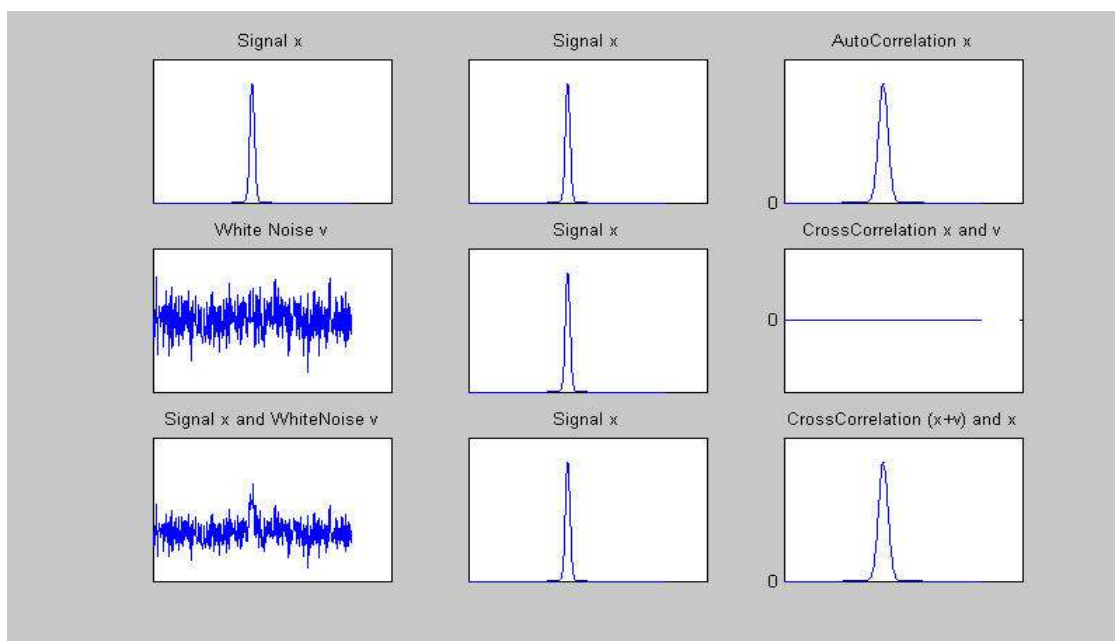


Figure 3-24: Using Cross-Correlation to Reduce Noise

It is very hard to get a real perfect white noise signal [40]. Figure 3-25 shows the result of the FFT of the practical noise signal. From the FFT magnitude and phase, the noise signal can be considered as the sum of the constant direct current component and the white noise.

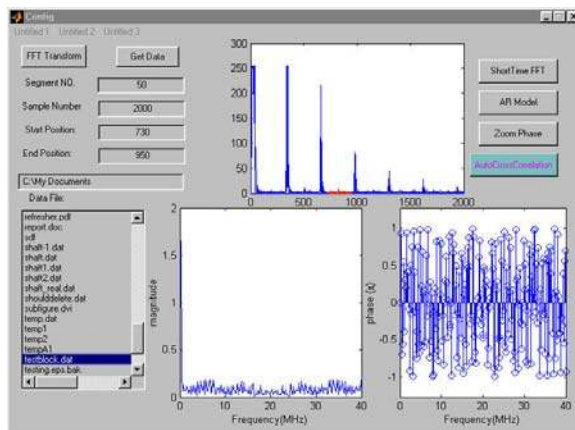


Figure 3-25: FFT of Noise Signal

In order to use cross-correlation, the reference signal (the template of the target signal) is needed in advance. According to the analysis in section 3.1, Signal Property, the simulated signal is chosen as the template signal. Figure 3-26 shows the original signal (blue color) and the Cross-Correlation (red color) of the original signal and the template signal. The difference between the two signals can be seen more clearly in Figure 3-27. From Figure 3-27 it is true that the cross-correlation does reduce the noise. However, the effect is not always so good for the peaks that are too high or too low. The main reason is that the template signal is not good for all the peaks. Another reason is that the data used to calculate the cross-correlation is too little.

Anyway, if only some peaks with certain amplitude need to be enhanced, but not all peaks, a specific template signal can be used to reduce the noise.

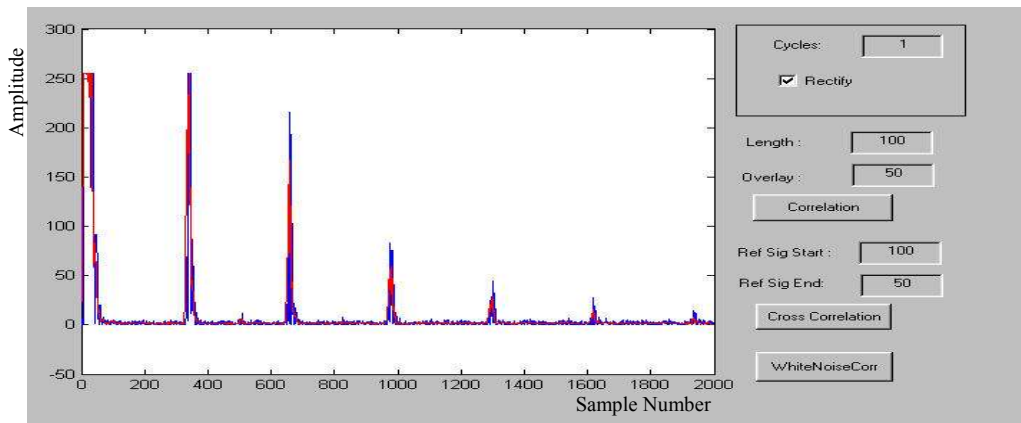


Figure 3-26: Cross-Correlation of Signal and Template Signal

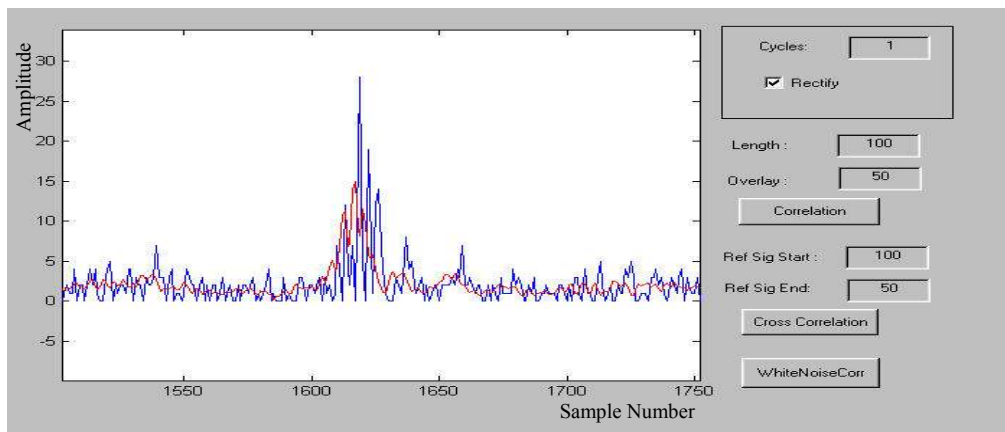


Figure 3-27: Detail of Cross-Correlation

3.3 Enhancement by Filter

Filters are very commonly used in the signal processing. Low Pass, High Pass, Band Pass, and Band Stop filters are all available, and the methods to design the filters are fully mature. Because the noise in the ultrasonic testing is mainly high frequency noise, low pass filter or band stop bank is desired.

For this special application to reduce the noise for the non-destructive signal, it is necessary to design a zero-phase shift filter, because the peaks position should not be changed after filtering.

There are two ways to achieve the target.

One is to design one MA (Move Average) filter with linear phase shift. And then shift the filter coefficients to the left half of the coefficient number. The filter will be a zero-phase shift, non-casual filter [41].

The other way is to design a filter as normal, filter the original signal, get the filtered signal F, and then revise the filtered signal F, filter the filtered signal F with the same filter, and finally, revise the result. The revision procedure is to sort the signal from $(x_1, x_2, x_3, \dots, x_n)$ to $(x_n, x_{n-1}, x_{n-2}, \dots, x_1)$. The final outcome is the filtered signal with zero-phase shift.

First the low pass filter is designed as equation 3-1 [27]:

$$H(z) = 0.0643z^{-1} - 0.1467z^{-2} + 0.2166z^{-3} + 0.7318z^{-4} + 0.2166z^{-5} - 0.1467z^{-6} + 0.0643z^{-7}$$

Eq 3-1

The frequency response of the low pass filter is shown in Figure 3-28:

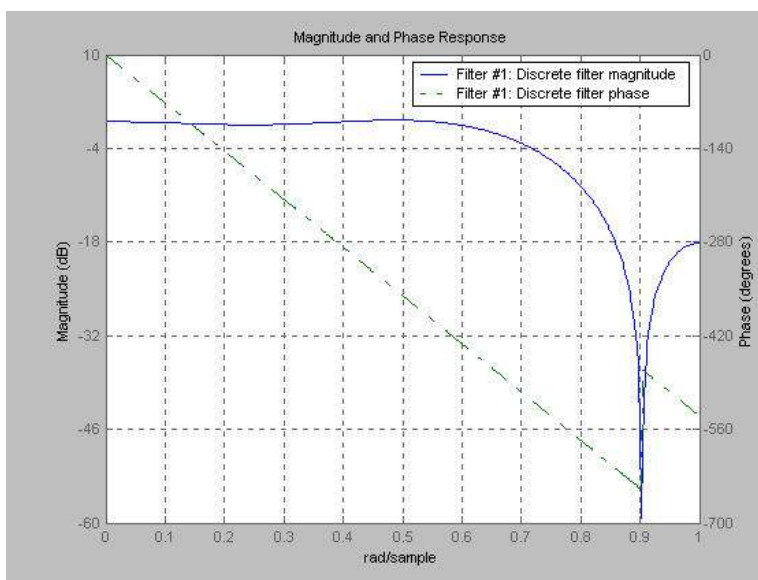


Figure 3-28: Frequency Response of the LP filter

The filter is a linear phase filter but not a zero-phase filter. When the signal is filtered with the LP filter, there is the obvious shift shown in Figure 3-29.

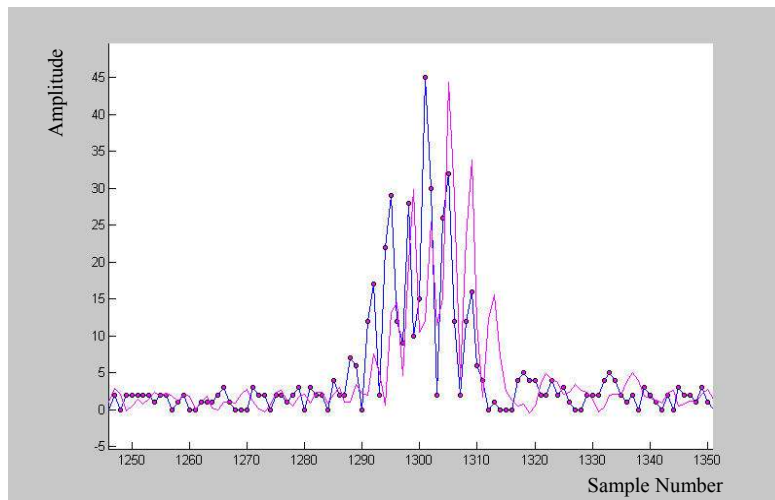


Figure 3-29: Filter with non-Zero-phase filter

Then the second method mentioned on paragraph 3 page 52 is followed. The result is shown in Figure 3-30 and Figure 3-31 is the detail of Figure 3-30. There is no shift at all in Figure 3-30 and Figure 3-31. And the high frequency is filtered (see the left and right part of Figure 3-31). However, the signal looks still noisy. That is reasonable. Recall that the FFT of the signal in Figure 3-6, the sample frequency is 40MHz, the signal frequency focuses on 10 MHz. The cutoff frequency of the LP filter is 15MHz. The noise from 0 to 15MHz still has a considerable effect on the signal.

If the sample frequency is higher and the signal is not rectified, use zero-phase filter should get a better result.

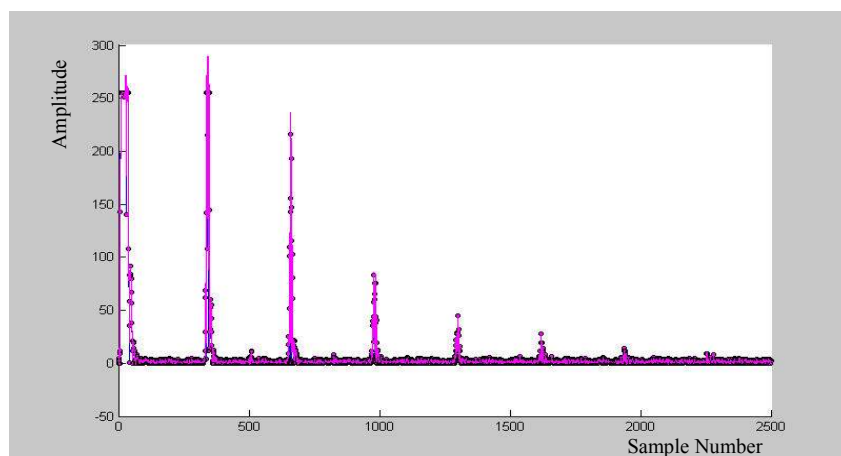


Figure 3-30: Filter with Zero-phase Filter

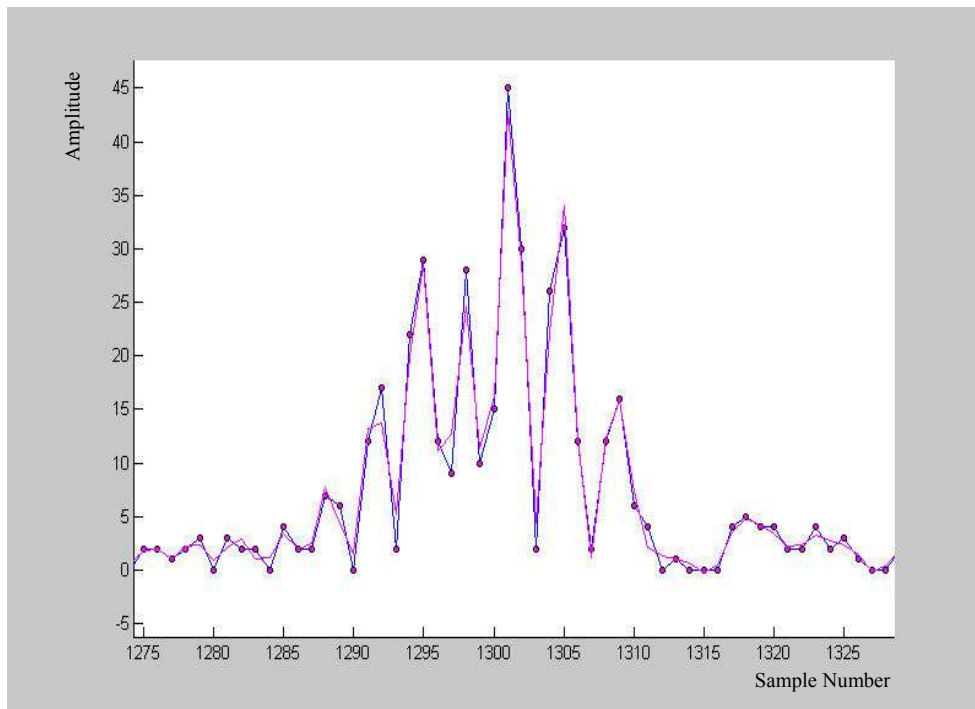


Figure 3-31: Detail of Figure 3-30 (Range 1275-1325)

3.4 Enhancement by Averaging

Averaging is always the best way to eliminate the random white noise from the signal, when:

1. There are enough data to do the averaging.
2. The time to get enough data is acceptable.
3. The time consumed by the averaging is acceptable.

The main advantage of averaging is that it will not distort the original signal.

This is the reason that the average capacity such as 100 million points per second is mentioned in many ultrasonic cards specifications.

Figure 3-32 and Figure 3-33 show the difference between signals with no averaging and signal with 4-averaging.



Figure 3-32: Signal with no average



Figure 3-33: Signal with 4-average

3.5 Chapter Summary

The frequency character of the signal is found by FFT of the simulated signal and the practical signal. The main frequency components of the practical signal are the times of the probe frequency. This is the effectiveness of rectification. Rectification blurs the frequency character of the signal and is the hardware limitation.

Based on this information about the signal, three methods to enhance the signal are discussed:

1. Enhancement by cross-correlation: Cross-correlation method need to know the reference signal beforehand. Because no definite reference signal exists, so this method is very limited in the signal enhancement.
2. Enhancement by zero-phase filter: Generally this method should be an effective method to reduce the signal noise. However, the output signal of the SFT4001H ultrasonic card is a rectified signal. There are some high frequency ripples in the

in FFT (ref Figure 3-8, Figure 3-10, Figure 3-11, and Figure 3-12). The zero-phase filter design should not remove too many of the ripples. That means that the cut frequency of the zero-phase filter should be high enough. For the specific application in this project, the difference between the cut off frequency of the zero-phase filter and the sample frequency is too small (ref Figure 3-28). So the effect of the zero-phase filter is not satisfying.

3. Enhancement by Averaging: When the time consumed by the averaging is acceptable, averaging is always the best way to eliminate the random signal noise.

Feature selection and classification are discussed in Chapter 4.

Chapter 4 Peak Detection

Whichever classification algorithm is adopted, the feature selected decides the performance of the classification algorithm. In this chapter, the most distinguishable features that can classify the peaks from non-peaks are selected. The result of the classification algorithm, Classification Tree, which is expatiated in chapter 2, is discussed.

4.1 Searching Local Maximum

From the signal shown in the figure 2-6 and figure 2-7, peaks come from the reflected ultrasonic wave. The target is to recognize the peaks that come from defects. As a recognition problem, normally it is needed to calculate the features of every sample point and then use some classification method to do the recognition task. Because it is known that the peaks that come from the defects are local maximum points, it is not needed to calculate the features of every data point. It is only need to calculate the features of every local maximum point, then use the classification method to recognition the target peaks.

The search procedure is described as below:

START:	Set segment length value as 40
	Set local maximum as the current data value
	Set increase counter as 0
	Do until no data available
	Move to next data
	If current data value larger than the previous data value
	Update the local maximum to the current data value;
	Increase counter increases by 1;
	Else
	If increase counter larger than or equal to segment length
	Record the current data position
	Reset increase counter as 0
END	

The value of segment length is experimental. If the segment length is small, more local maximum points will be selected. If the segment length is large, the less local maximum points will be selected. However, maybe the true defect is ignored also. This value of segment length is chosen using an artificial intelligence method, explained in the later paragraphs.

This step is necessary and significant. By searching local maximum, it is no need to calculate the features of every data point. This reduces much computation time in the real-time testing.

4.2 Feature Selection

Feature selection or extraction is a process of selecting a mapping of the form $x' = f(x)$ by which a sample $s = \{x_1, x_2, \dots, x_n\}$ in a n -dimension measurement space \mathfrak{R}^n is transformed into the form $s' = \{x'_1, x'_2, \dots, x'_m\}$ in a m -dimension measurement space \mathfrak{R}^m with $m < n$. The main objective of these processes is to retain the optimum salient characteristics necessary for the recognition process, and to reduce the dimensionality of the measurement space so that effective and easily computable algorithms can be devised for efficient categorization.

A judicious selection of features for building classifiers is a very crucial aspect of classifier design, and deserves careful consideration. Success of the classifier heavily depends on the set of features used to represent objects of interest [42]. On the one hand, there is certainly nothing to lose in using all available measurements in classifier design. On the other hand, too many features make the classifier increasingly complex unnecessarily.

In many problems, extracting 'features' from the raw data will result in significantly improved pattern recognition. Features capture characteristics of the inputs that are most relevant for the estimation or classification problem at hand. Typically, domain-specific knowledge is used to develop good features.

Based on the analysis on Chapter 3 and the shape of the peaks, the selected features are:

1. The AR Model Coefficients: 4th-order AR Model is used to express the peak signals: $H(z) = \frac{e}{1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3}}$. Although higher order for the AR model is possible for calculation, that will add more complexity and be not much effective because of the noise and the limitation of the sample data. The AR model coefficient a_1, a_2, a_3 are selected as three features.
2. Standard Deviation (σ): The measure of dispersion or scatter of values of a random variable about the mean. If the value tends to be concentrated near the mean (μ), the variance is small; while if the values tend to be distributed away from the mean, the variance is large.
3. Pearson Correlation (Pearson's "r"): The measure of the linearity of the data.
4. Dispersion Uniformity Degree: Dispersion Uniformity Degree is defined as: $DUD = \frac{Maximum - \mu}{\sigma}$

where: *Maximum* is the maximum value in the data set.

μ is the mean value of the data set.

σ is the standard deviation.

The next step is to choose the best set of features that discriminate the classes most efficiently. That is, enhance the separability among the different classes while increasing homogeneity within the respective class at the same time [31].

In Figure 4-1, almost all the peaks with 10% -100% full screen amplitude congregate in Zone 2. The peaks with amplitude low than 10% full screen amplitude scatter in Zone 1. Here the truncated signal is not considered in order to reduce the complexity.

In Figure 4-2 (also ref to Figure 4-4), the set of peaks and the set of non-peaks are more agglomerated respectively compared to Figure 4-1. And the boundary is more obvious than that in Figure 4-1. The mixed part is not as large as the mixed part in Figure 4-1.

So the three AR coefficients are eliminated, only taking standard deviation σ , Pearson's "r" and Dispersion Uniformity Degree are left as the features. This step reduces the complexity further.

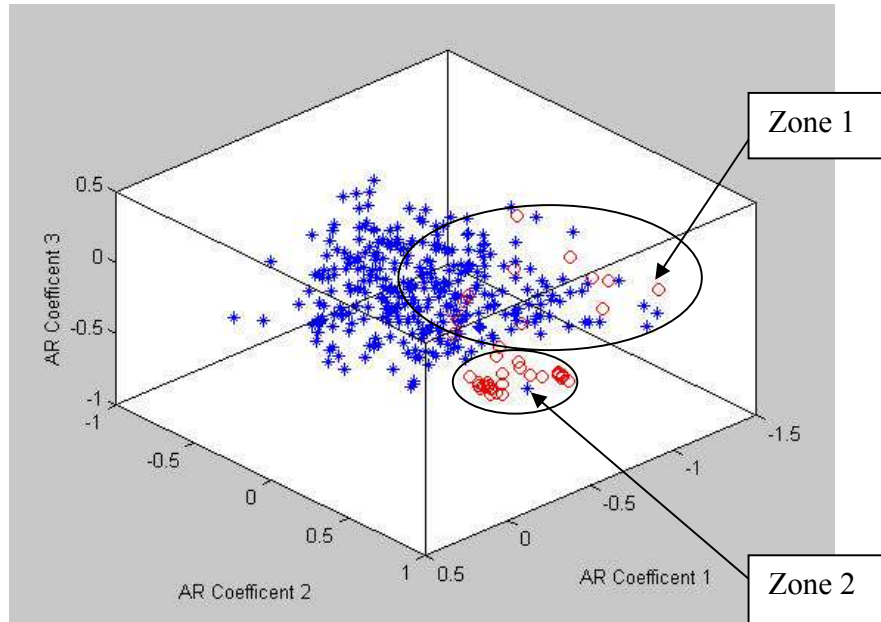


Figure 4-1: Feature a_1, a_2, a_3

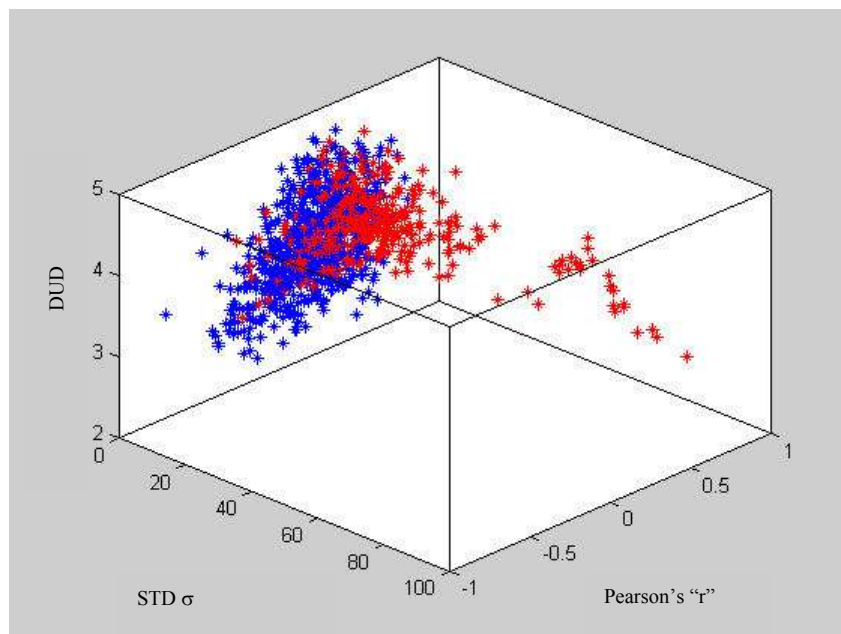


Figure 4-2: σ , Pearson's "r" and DUD

4.3 Using Classification Tree

After the feature space is obtained, the next task of a pattern recognition system is classification. The goal of a classifier is to classify objects of interest into one of the number of categories or classes [43].

There are many methods to do the classification task [44-50]. A neural network is the most often way mentioned in engineering papers. However, for the task to recognize the peaks, the classification tree is the better method.

A neural network classifier is suitable for the classification case shown in Figure 4-3(A). When the distance between class 1 and class 2 is short, in order to fulfill the classification task, more sample data is required to train the neural network, and of course more training time is also required to train the neural network [51][52]. However, for the classification task shown in Figure 4-3(B), no matter how many samples used, no matter how long the neural network is trained, it is not possible to fulfill the classification task perfectly for the common forward neural network [53][54]. The classification tree is not suitable for the classification task shown in Figure 4-3(E), because the classification tree will produce a classification rule with zigzag form finally. That will result in a complex classification rule, or else a classifier with larger error rate. The classification tree is very suitable for the classification task shown in Figure 4-3(C). As for the classification task shown in Figure 4-3(D), a satisfactory result can be obtained by using classification tree, even the class-1 and class-2 have a small overlap part [55].

The task to classify the peaks in the non-destructive testing signal is much similar to the case shown in Figure 4-3©, just with a small overlap part. See Figure 4-4 and Figure 4-5 for detail.

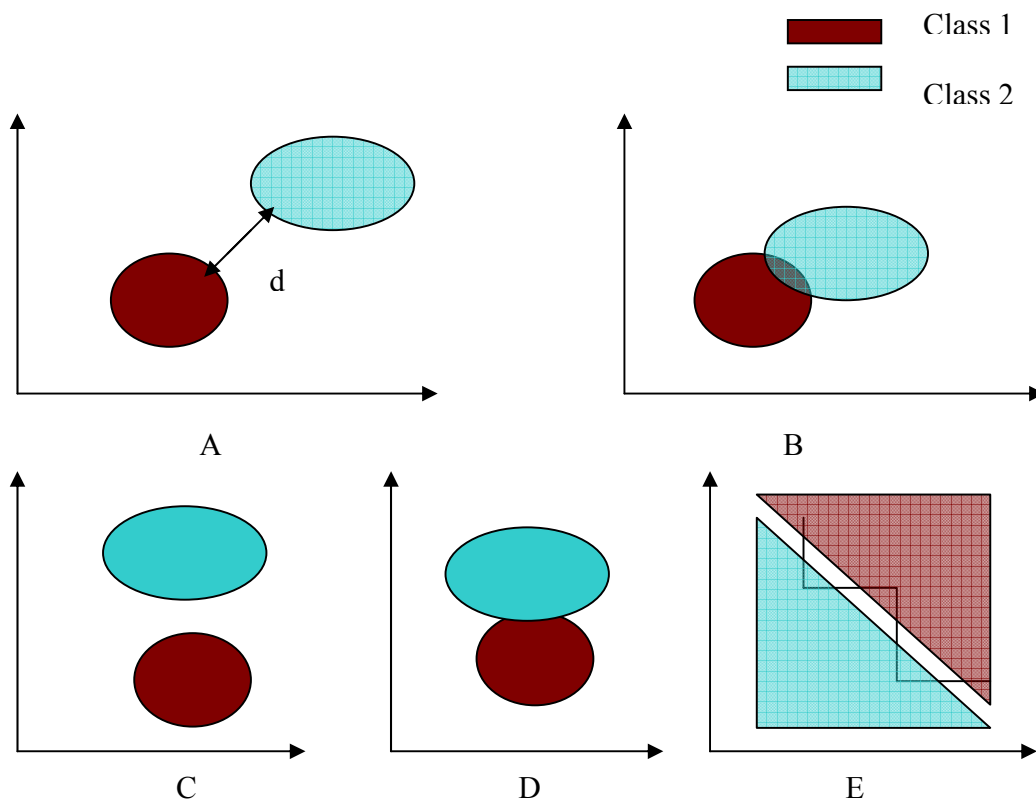


Figure 4-3: Comparing Neural Network and Classification Tree

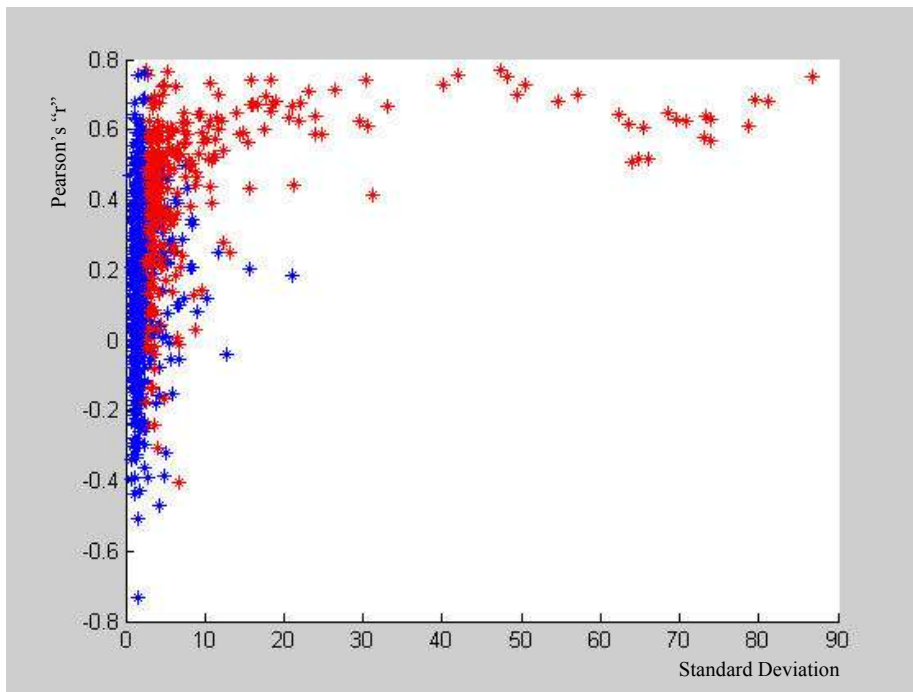


Figure 4-4: Peaks (red) and Non-peaks (blue) (1)

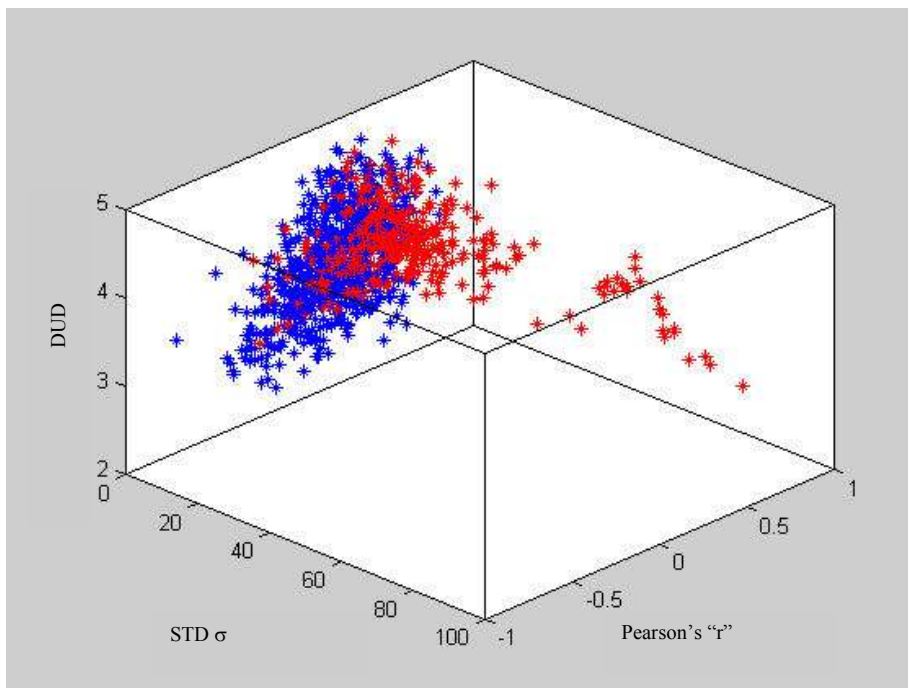


Figure 4-5: Peaks (red) and Non-peaks (blue) (2)

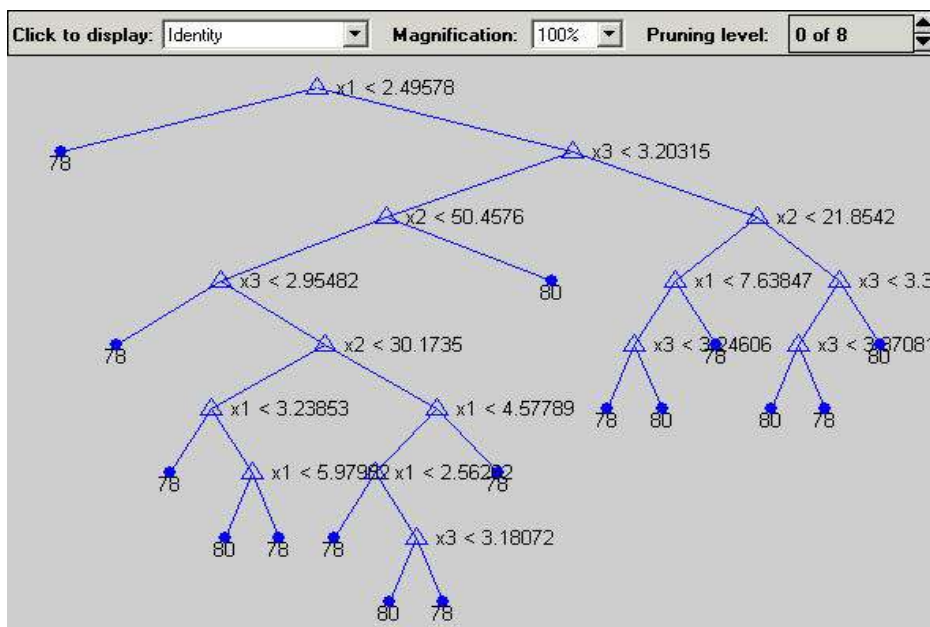


Figure 4-6: Whole Classification Tree

Figure 4-6 is the whole tree generated according the splitting algorithm expatiated in Chapter 2.6.1. The strings beside the nodes (triangle) are the condition to split the nodes. The leafs (dot) are the classified samples. "78" stands for the class "non-peak" and "80" stands for the class "peak". This is the whole tree, which means that all the data in the sample set can be classified successfully by the classification rule described.

Obviously, the rule is too complex and is hard to explain. And more, this is not the perfect classification rule because the sample set does not include the entire possible situation and there are error data that is misclassified. So pruning the tree is necessary.

Figure 4-7 shows that the whole is pruned to more simple form. The gray dash lines are the branches pruned from the whole tree based on the pruning algorithm expatiated in Chapter 2.6.3.

More details about the pruning procedure are shown in Figure 4-8.

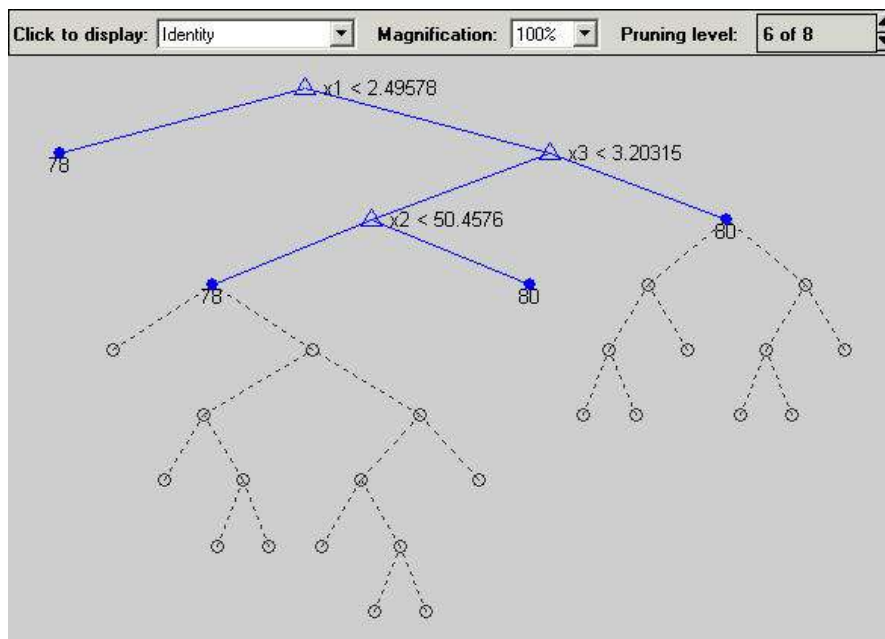


Figure 4-7: Pruning Classification Tree

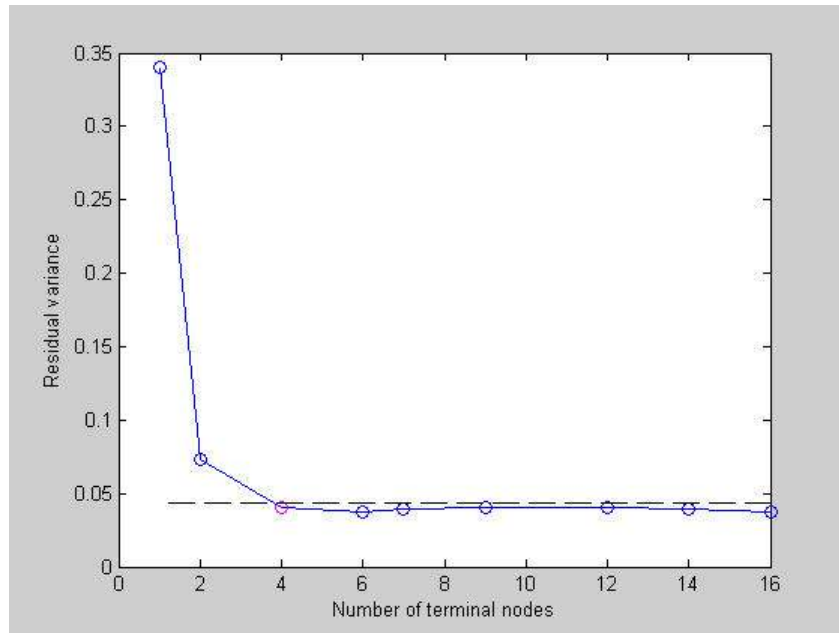


Figure 4-8: 1-SE Rule

The circles on the blue line are the respective trees with respective classification error rate. The horizontal dash line is 1-SE more than the minimum classification error rate. The shortest tree below the horizon slash line is picked. That is the tree represented by the red circle. The ultimate tree is shown in Figure 4-9.

“x1” stands for standard deviation. “x2” stands for the 100 times of Pearson’s “r”. “x3” stands for the dispersion uniformity degree. “78” stands for the class “non-peak” and “80” stands for the class “peak”.

It is easy to explain the classification rule shown in Figure 4-9. That is:

The samples are peak if they satisfy:

1. The standard deviation larger than 2.49578
2. And the dispersion uniformity degree larger than 3.20315 are peaks.

Or

1. The standard deviation larger than 2.49578
2. And the dispersion uniformity degree lower than 3.20315
3. And the Pearson’s “r” larger than 50.4576

These values depend on the training data. When the training data set is small, the values vary much. When the training data set is large enough, the values only vary in a small range. Those features describe the shape of the peaks. It is a kind of shape analysis. If the peak shape was complex, this method would be less effective.

The other form of the classification rule represented by C language is:

```
( (stdvalue_temp>STDTHRESHOLD) & ...
    ( abs(pearson_r_temp)>PEARSON_THRESHOLD)) ...
    |...
( (stdvalue_temp*STD_MEAN_MAX_THRESHOLD+meanvalue_temp)<maxvalu
e_temp &...
    (stdvalue_temp>STDTHRESHOLD)...
    )...
```

where:

1. STDTHRESHOLD is the standard deviation threshold 2.49578
2. PEARSON_THRESHOLD is the Pearson's r threshold 50.4576
3. STD_MEAN_MAX_THRESHOLD is the dispersion uniformity degree 3.20315

The ultimate effect of the classification rule can be seen from figure 5-1. The red rectangles marks are the peaks classified.

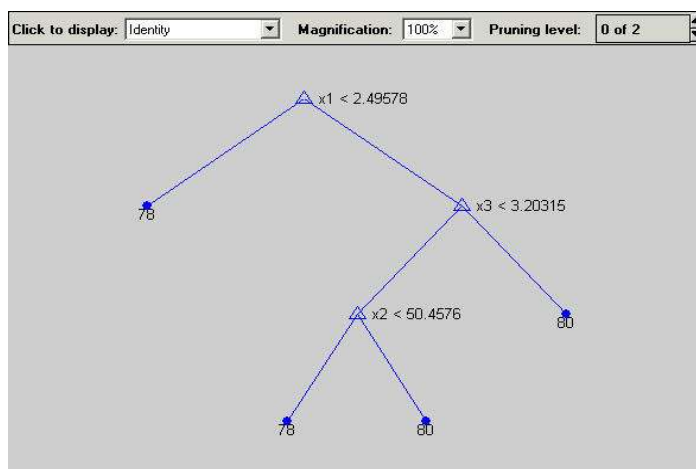


Figure 4-9: Ultimate Classification Tree

4.4 Experiment Classification Result

The table 4-1 shows experiment result. The automatic peak detection algorithm can detect the peaks with very little difference comparing with the human operator. When the gain is too high (higher than 45db), the signal will be much noisy. Even human operator cannot recognize all the peaks properly. In practice, almost all the test procedure defines the gain range.

Table 4-1: Experiment Result

Gain (db)	Peaks recognized by eye	Peaks recognized automatically	error
6	2	2	0
10	2	2	0
15	3	3	0
20	4	4	0
25	4	4	0
30	9	9	0
35	16	15	1
40	20	21	1

The peaks detected include the peaks that come from defects and the peaks that come from the structure of the work piece, such as corners, edges and surfaces. There are two methods to discriminate the two kinds of peaks.

One method is using ultrasonic simulation software, such as UTSIM. UTSIM is a user interface integrating a CAD model representing a part under inspection and an ultrasound beam model. The beam model can accurately predict fields for focused or planar beams, through curved or flat interfaces in a 3D space. The 3D model of the test piece is created and then the ultrasonic simulation software is use to get the simulation test result, that is

called “signature “. And then the practical test result is compared with the signature. The peaks in the practical test result that do not exist in the signature are the peaks that come from the defects.

The other method is putting the probe on a perfect sample. And then the test result is recorded as the signature. Then compare between the practical test result and the signature is carried out in order to discriminate the defect peaks and the structure peaks.

4.5 Chapter Summary

Four features are selected at first:

1. The AR Model Coefficients.
2. Standard Deviation.
3. Pearson Correlation (Pearson’s “r”).
4. Dispersion Uniformity Degree.

Although the first feature shows very good cluster property for the peaks with 10% - 100% full screen amplitude, the scatter is notable for the peaks with amplitude low than 10% full screen amplitude. So the first feature is ignored. This also simplified the classification problem.

The algorithm of Classification Tree is applied based on the 2nd, 3rd, and 4th feature. The experimental result is quite good.

The next step is to integrate the algorithm into the testing system and design the automatic procedure for the calibration and the testing. The automatic calibration procedure and the automatic shaft testing procedure are shown in next chapter. The implementation of both procedures of the SOFRA software is developed.

Chapter 5 Automatic Testing Procedure

Ultrasonic non-destructive testing is a strictly prescribed procedure. There are many national and international standards [1][56] to follow with Ultrasonic NDT. The automatic calibration procedure and the automatic testing procedure must be designed based on the standards. A SOFRA software is developed to realize the two automatic procedures. Matlab is not suitable to develop the real-time testing system because the Matlab program running speed is relatively slow. So the C++ is applied as the programming language. The 90° longitudinal wave probe is used, hence all the ultrasonic wave velocities mentioned in this chapter are the wave velocity of longitudinal wave.

5.1 Statflow of SOFRA Software

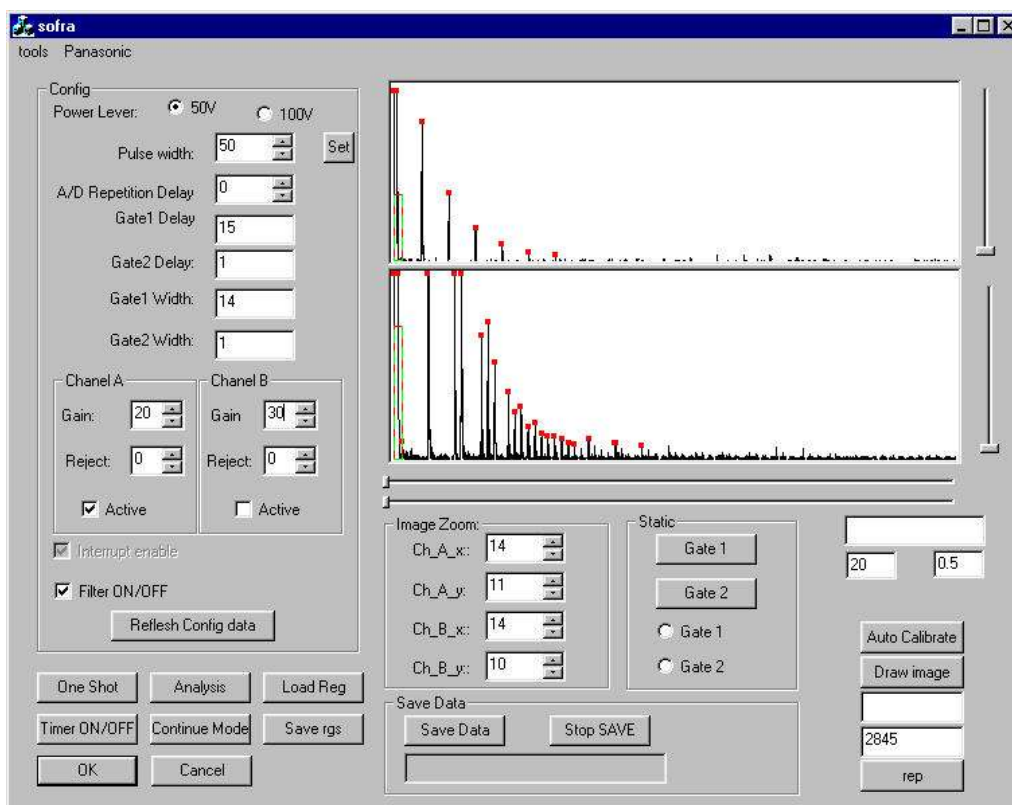


Figure 5-1: SOFRA Main Interface

Having developed the peak detection algorithm, the next step is to automate the non-destructive testing procedure. A SOFRA Software is developed to do the non-destructive testing automatically. Figure 5-1 is the main interface of SOFRA.

As a basic requirement to the automatic non-destructive testing system, the system must be able to finish most of the operations that the operator does when he use the manual testing device, such as adjusting the gain, adjusting the time delay, recognizing the peaks, calculating the defect position, and saving the data.

The functions in the main interface include (shown in Figure 5-2):

1. The signals of two channels are displayed simultaneously.
2. Zoom in or zoom out the signals.
3. For each channel, two gates (Gate 1 and Gate 2) can be set respectively. It is very convenient to operate, just select the gate needed set and then press and drag the mouse in the signal display area.
4. Set the impulse voltage height, 50 V or 100 V.
5. Set the impulse width, from 50 ns to 775 ns, step 25 ns.
6. Set the A/D convert delay, from 0 to 400 ns. It is a very important parameter. The start point and end point of testing range is decided by the delay value, ultrasonic wave velocity, buffer size and sample frequency:

$$TestingRangeStartPoint = \frac{DelayValue \times Velocity}{2}$$

$$TestingRangeEndPoint = TestingRangeStartPoint + \frac{BufferSize \times Velocity}{SampleFrequency \times 2}$$

When the test part is longer than the testing range covered by the buffer, the only way is to set a larger delay value. By such a way, the test range is extended to test the long part, such as shafts.

7. Set the gain and reject lever for each channel.
8. Open and save the card configuration.
9. Save the test data.

Those functions cover all the needed operation during the manual non-destructive testing. For an automatic non-destructive system, the system should not bother the user with too many abstruse terms. In the following two sections, automatic calibration procedure and automatic testing procedure are proposed. Very few and easy to understand parameters need to be set before the calibration procedure and testing procedure.

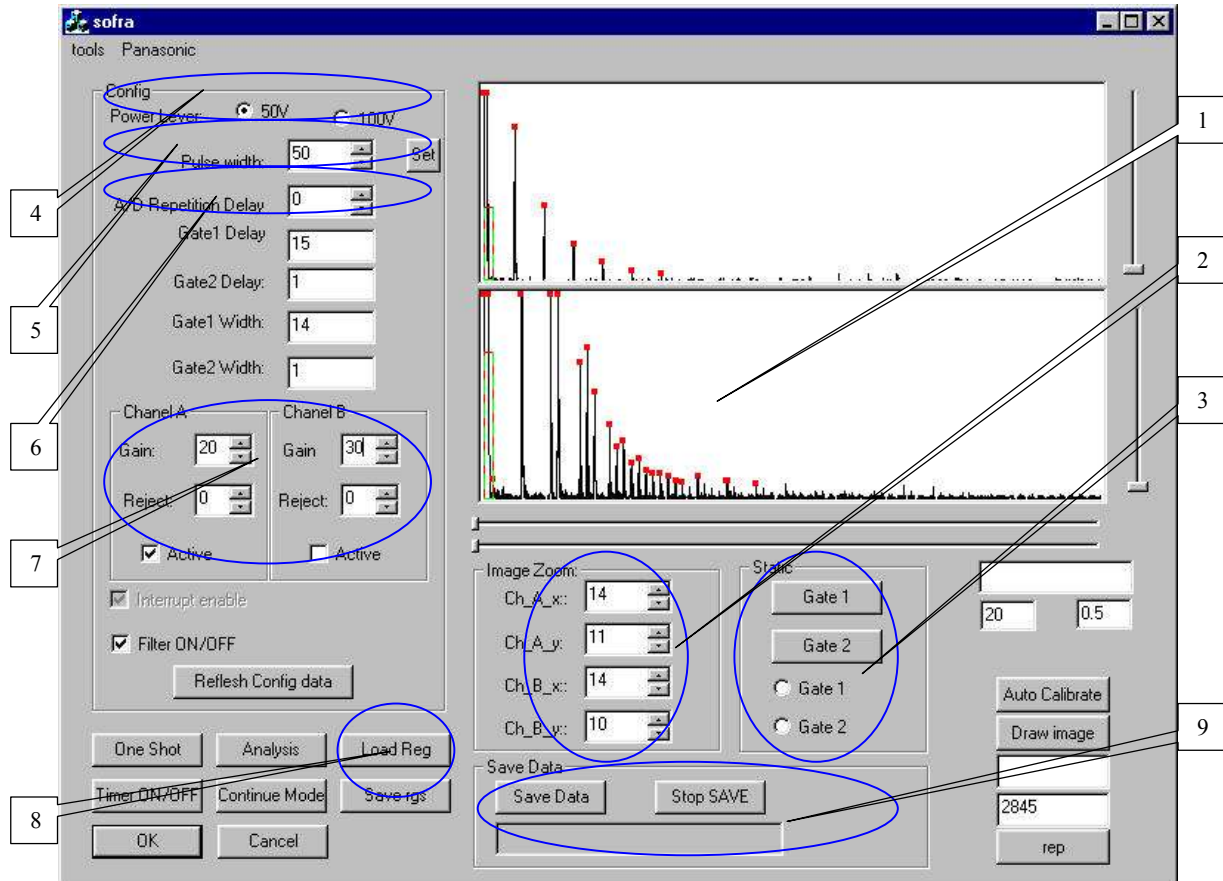


Figure 5-2: SOFRA Function

5.2 Automatic calibration

Calibration is required by national and international standards [1][56] and most require that a system of periodic calibration and maintenance must exist for any facility doing nondestructive testing. The first important parameter should be obtained by calibration is the ultrasonic wave velocity in the special material. In order to obtain the flaw position exactly, the ultrasonic wave velocity in the material should be precise enough.

The parameters should be set before automatic calibration include (Shown in Figure 5-4):

1. Sample frequency. For the SOFRA400H ultrasonic card, the sample frequency is fix to 40 MHz.
2. Calibrate block thickness. A cubic calibrate block is used to calibrate. The dimension precision of the calibrate block should be high enough, because this parameter has notable and direct effect on the calibration result: the ultrasonic velocity in the material.
3. Material Name. This is only for annotative purpose.
4. Precision: It also an important parameter, it is the number of echo peaks used to calculate the ultrasonic velocity in the calibrate block. This parameter is used for thinner calibrate block, a better measure value can be obtained by setting the parameter as 4 or 5. Shown in Figure 5-3

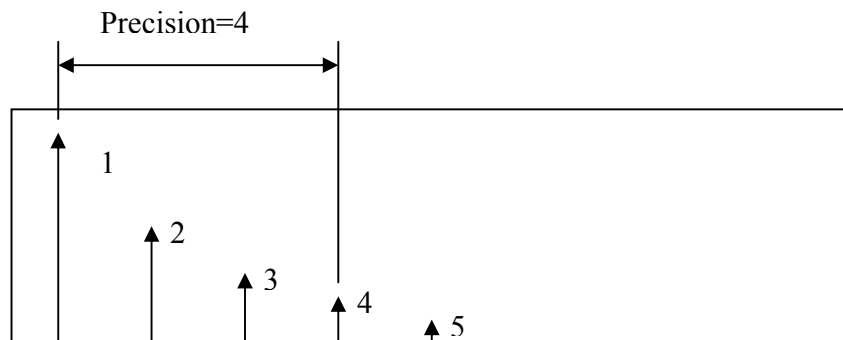


Figure 5-3: Meaning of “Precision”

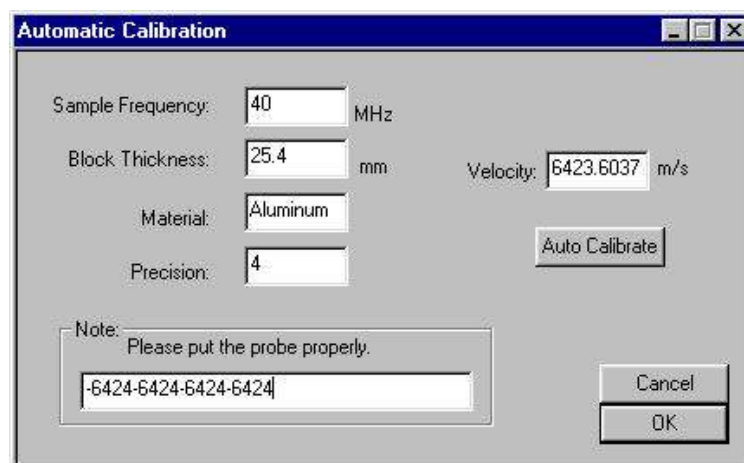


Figure 5-4: Automatic Calibration Parameters and Result

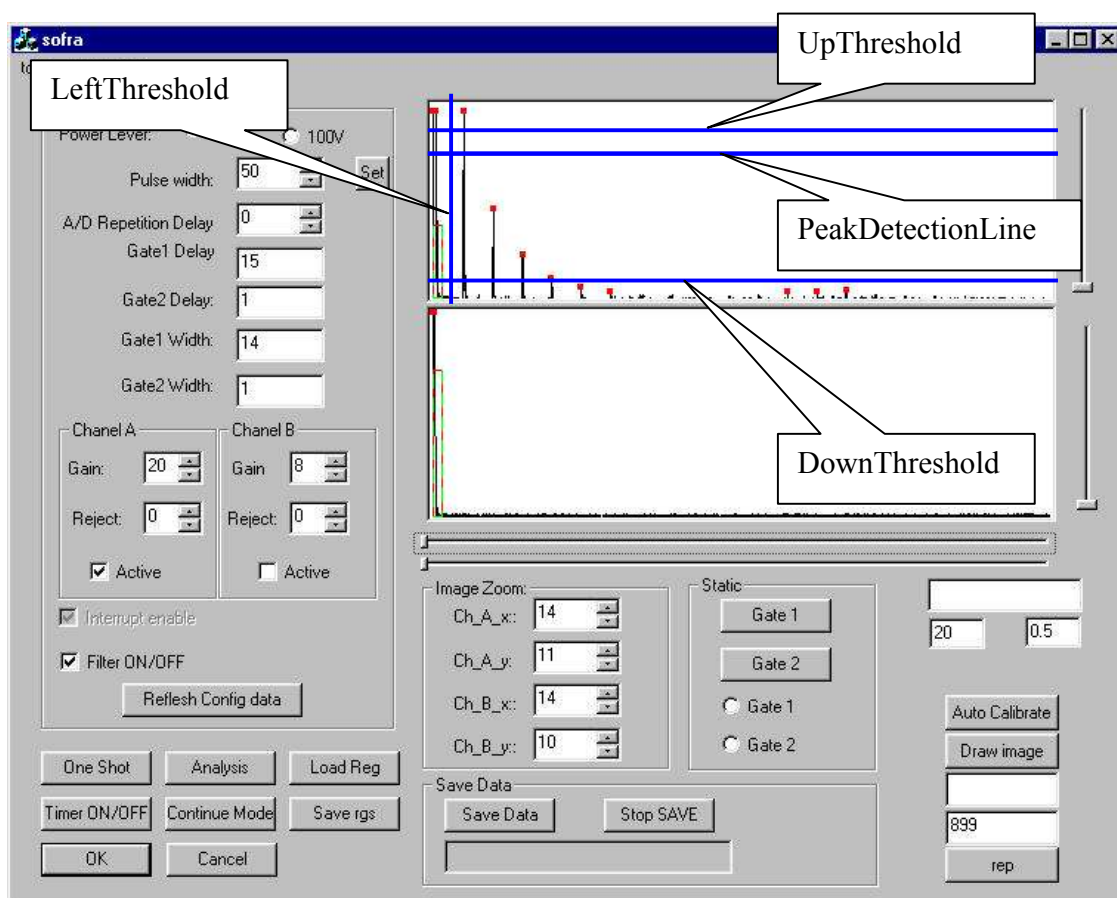


Figure 5-5: Automatic Calibration

First, the peak detection area should be set. Based on the analysis in the chapters before, the classification error rate is high for peaks with too high (truncated) or too low amplitude. So two thresholds are predisposed: UpThreshold and DownThreshold. The peaks with amplitude lower than the DownThreshold or higher than the UpThreshold are ignored.

Because of the echo signals form the interface between the probe and the couplant and the interface between the couplant and the test block, the initialized peaks [57] [58] have no significance. So another threshold is predisposed: LeftThreshold. The peaks that appear to the left of the LeftThreshold are ignored too.

Then the automatic calibration begins:

1. Initialize one structure array---PeakTimeOrder:

{ int GainValue int PeakValue int PeakPosition }

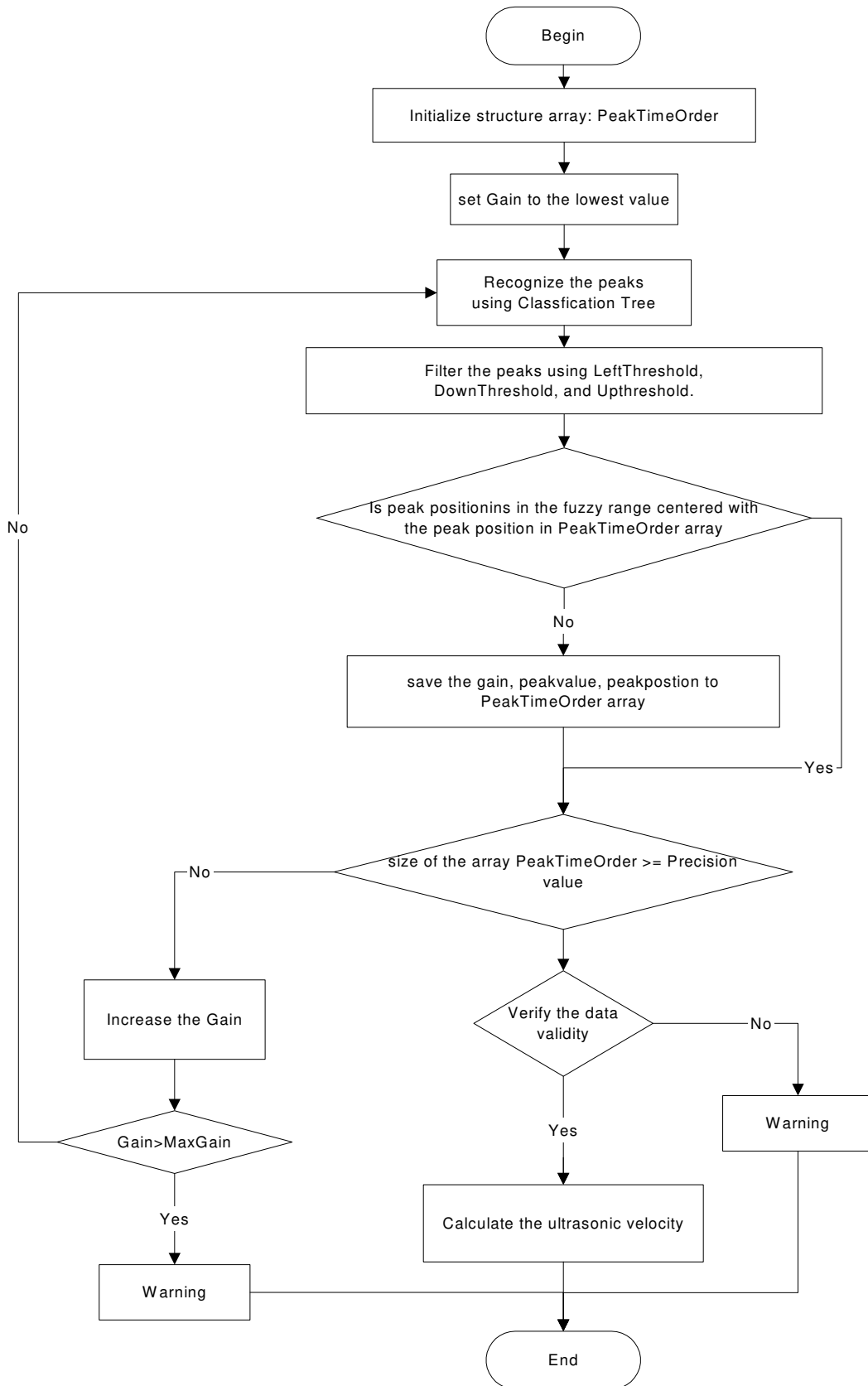
2. Set the gain to the lowest value.
3. Recognize all the peaks.
4. Ignore the peaks that appear to the left of LeftThreshold, lower than DownThreshold or higher than UpThreshold.
5. For the peaks locating between UpThreshold and PeakDetectionLine , compare the position of every peak with every PeakPosition value in structure array PeakTimeOrder,
 - if the difference is less than predisposed fuzzy factor (FuzzyFactor), that means that the two peaks are in the same position, the difference is caused just by noise.
 - if not, save the current gain value, peak amplitude and peak position to the structure array PeakTimeOrder.
6. Compare the size of array PeakTimeOrder
 - If the size of array PeakTimeOrder less than the Precision value set in Figure 5-4, increase the gain value, return to step 2.
 - For the size of array PeakTimeOrder equal to the Precision value set in Figure 5-4, the ultrasonic velocity is calculated by:

$$\frac{\text{Thickness} \times \text{SampleFrequency} \times 1000}{(\text{Distance Between 1st Peak and } (\text{precision} - 1)\text{th Peak}) \times (\text{Precision} - 1) \times 2}$$
7. Display the ultrasonic velocity.
8. Automatic calibration end.

The automatic calibration flow chart is shown in next page.

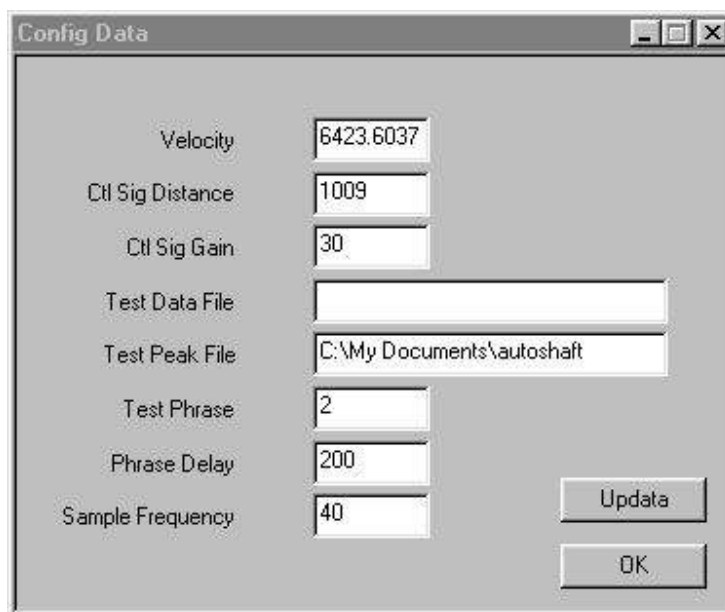
The measure velocity values of repeat test are displayed in the text box in the Figure 5-4, the repeatability of the calibration is relatively good.

Chapter 5: Automatic Testing Procedure



5.3 Automatic Shaft Testing

After automatic calibration, the next step is to do automatic shaft testing. Before testing procedure, several parameters need to be set. Figure 5-6 shows the configure data for automatic shaft testing.



Parameter	Value
Velocity	6423.6037
Ctrl Sig Distance	1009
Ctrl Sig Gain	30
Test Data File	
Test Peak File	C:\My Documents\autoshaft
Test Phrase	2
Phrase Delay	200
Sample Frequency	40

Figure 5-6: Automatic Shaft Testing Parameters

The details about the parameters:

1. Velocity: The ultrasonic velocity is the measured value obtained by the automatic calibration.
2. Control Signal Distance: This distance is the position with clearest echo signal. “Clearest” have two meanings: one is that the signal amplitude is high enough; the other is that there in no other signals near the signal. The distance value is required because the relative positions of all other peak signals are deduced based on the distance value.
3. Control Signal Gain: The control gain is required by following the SMRT shaft testing specification [21].
4. Test Data File: All the testing data are saved in this file. The saved testing data is used to reappear the whole testing procedure.

5. Test Peak File: All the details about the recognized peaks are saved in the file. The saved data is analyzed to find the defects and corresponding positions.
6. Test Phrase: Because the buffer in the ultrasonic card is not enough to store all the testing data for a long shaft part, the testing procedure have to be divided into several phrases to cover the whole shaft.

For example:

The buffer size is 8 K bytes, the sample frequency is 40 MHZ, the shaft is 1 m long, the ultrasonic wave velocity in the shaft is 6400 m/s, then the buffer can cover: $\frac{BufferSize \times Velocity}{SampleFrequency \times 2} = 0.64(m)$. In order to test the section farther than

0.64 m the A/D converter delay time need to be set to certain value. If the A/D converter delay time is set to 200 ms, the section form 0 to $Delaytime \times Velocity / 2 = 0.64(m)$ will be skipped. The buffer covers from 0.64 m to 12.8 m now.

7. Phrase Delay: This value is the incremental value of the A/D converter delay time.
8. Sample Frequency: For SOFRA4001H ultrasonic card, it is fixed at 40 MHZ.

The automatic shaft testing procedure is similar to the automatic calibration procedure:

1. Initialize one structure array---PeakTimeOrder:

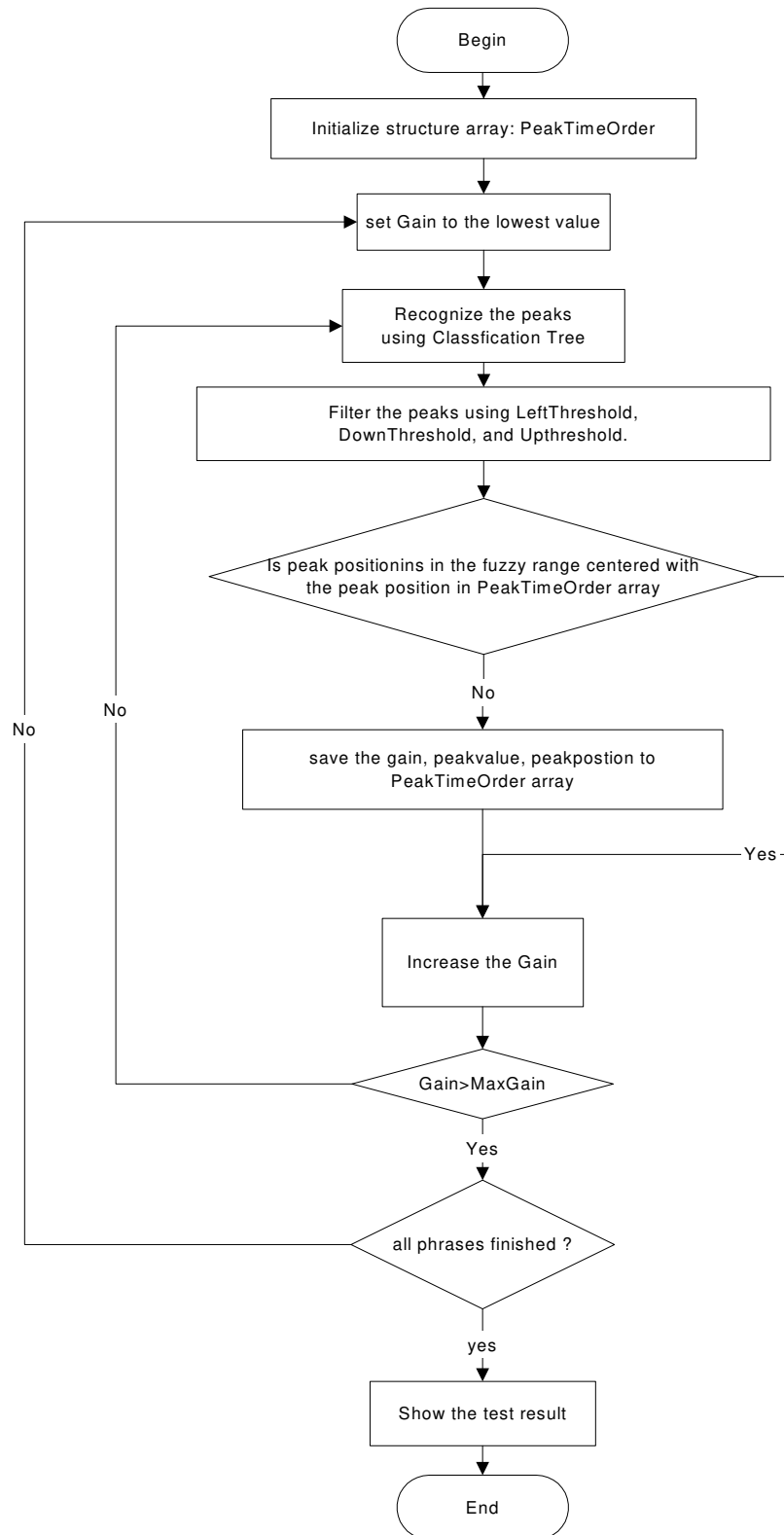
```
{ int GainValue    int PeakValue    int PeakPosition }
```

2. Set the gain to the lowest value.
3. Recognize all the peaks.
4. Ignore the peaks that appear to the left of LeftThrehold, lower than DownThreshodl or higher than UpThreshold.
5. For the peaks locating between UpThreshold and PeakDetectionLine , compare the position of every peak with every PeakPostion value in structure array PeakTimeOrder,

- If the difference is less than the predisposed fuzzy factor (FuzzyFactor), that means that the two peaks are in the same position, the difference is caused just by noise.
 - If not, save the current gain value, peak amplitude and peak position to the structure array PeakTimeOrder.
6. If the gain is not the maximum value, increase the gain, go back to step 3.
- If the gain is the maximum value, check out whether all phrases finished or not.
- If yes, the automatic shaft testing end
 - If no, go to the next phrase testing by increasing the A/D converter time delay time and Setting the gain to the lowest value, then return to step 3

The automatic shaft testing flow chart is shown on the next page.

Chapter 5: Automatic Testing Procedure



The peak information collected from the testing procedure is shown in Figure 5-7, Figure 5-8 (2D representation) and Figure 5-9, Figure 5-10 (3D representation).

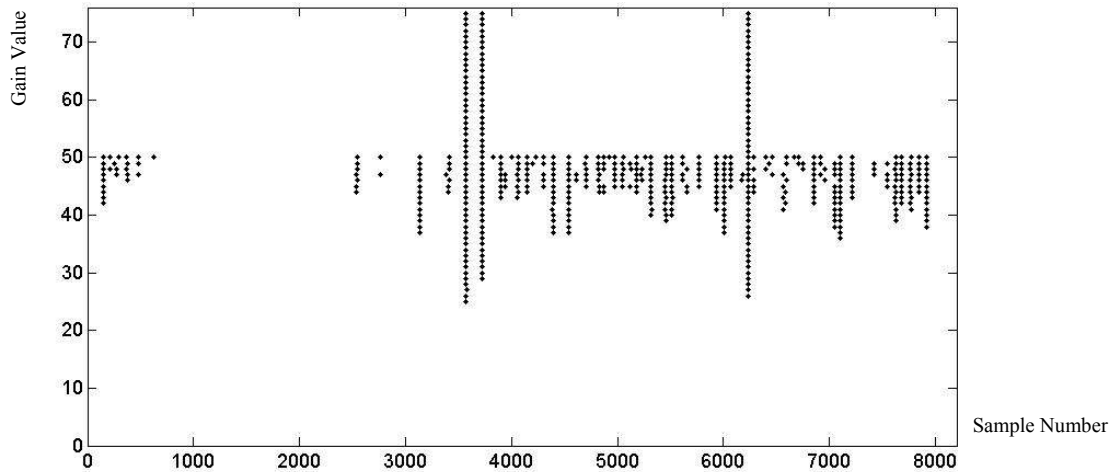


Figure 5-7: Peak Position (Shaft Testing Phase 1)

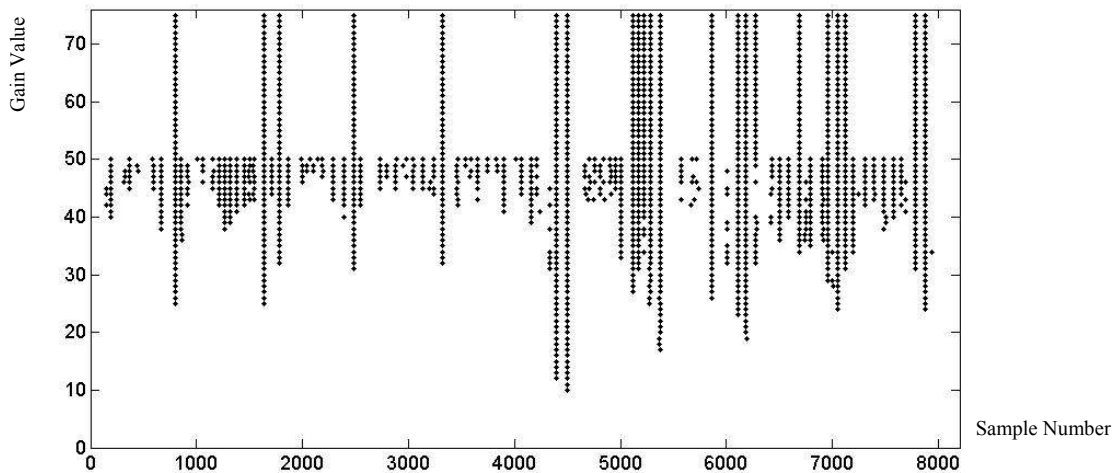


Figure 5-8: Peak Position (Shaft Testing Phase 2)

In Figure 5-7 and Figure 5-8, x-axis is the peak position value and y-axis is the gain value. With the gain grows up, there are more and more peaks. Of course the amplitude of the peaks is also grows up, as shown in Figure 5-9 and Figure 5-10 (z-axis is the amplitude value in Figure 5-9 and Figure 5-10). Because when the gain value is greater than 50, there are too many noise peaks, the maximum gain value is set to 50, although the gain

can be adjusted to 74. The dots appeared above the gain value 50 means that the peak at that position has passed the Peak Detection Line before the gain grows up to 50.

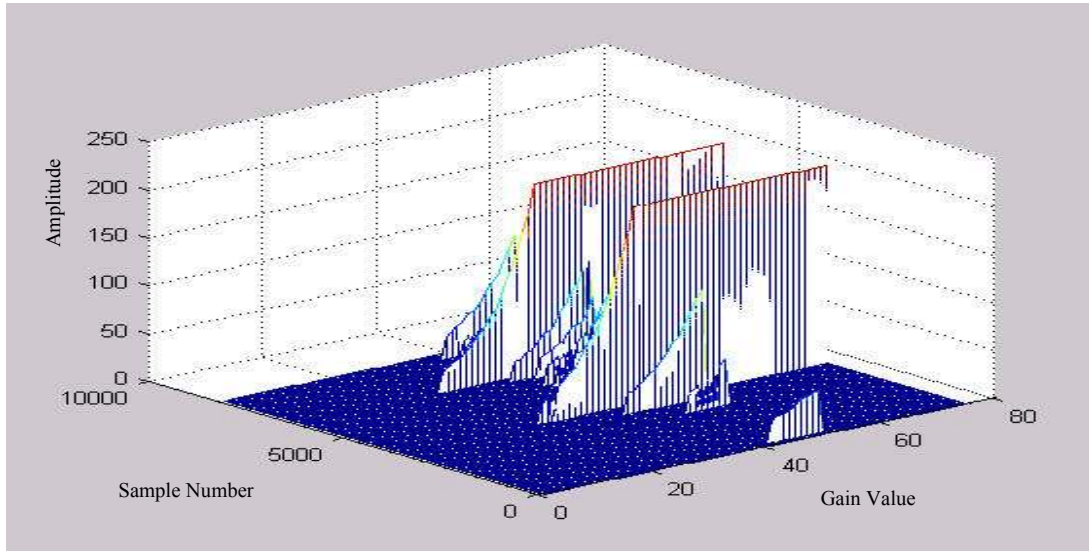


Figure 5-9: Peak Position (Shaft Testing Phrase 2) 3D

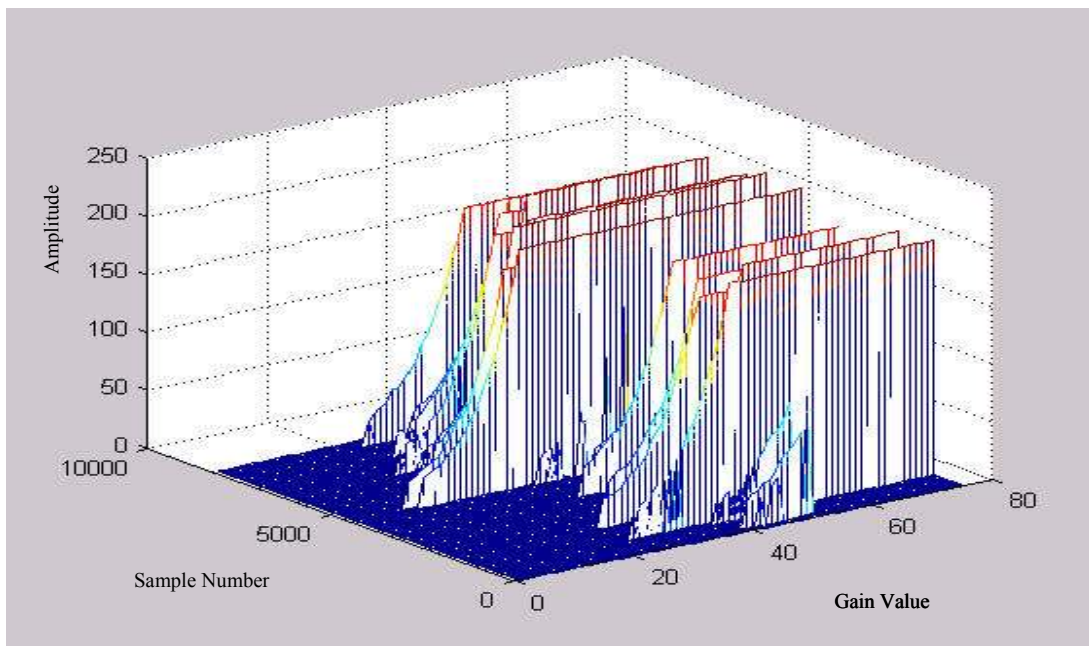


Figure 5-10: Peak Position (Shaft Testing Phrase 1) 3D

All the peak data at every gain value is saved in the peak test file specified in Figure 5-6. Then the manual method is followed to find the defect position by comparing the peaks with the signature of non-defect standard shaft according the value of control gain.

Figure 5-11 shows the testing result when control gain value is 36. From Figure 5-11, the defect signals and the corresponding positions can be easily identified.

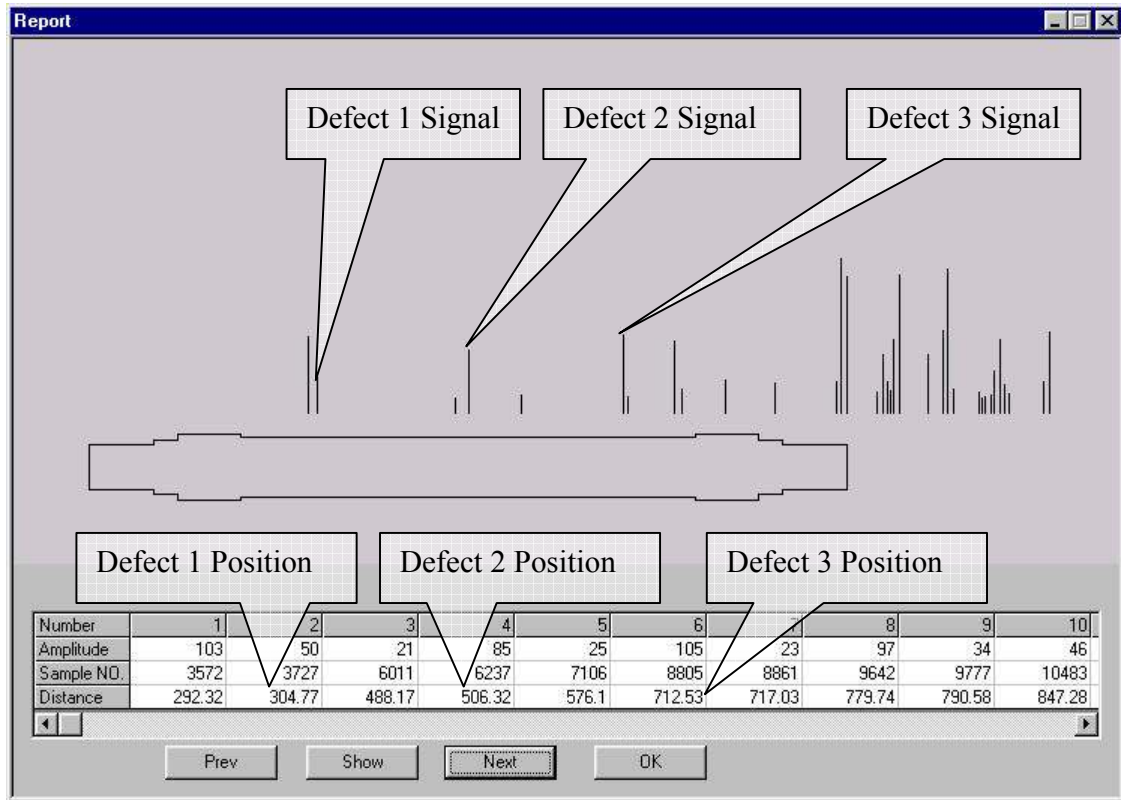


Figure 5-11: Report Result (Control Gain 36)

The real defects in the shaft are shown in Figure 5-12. Comparing the testing result to the real value, the testing result is relative good.

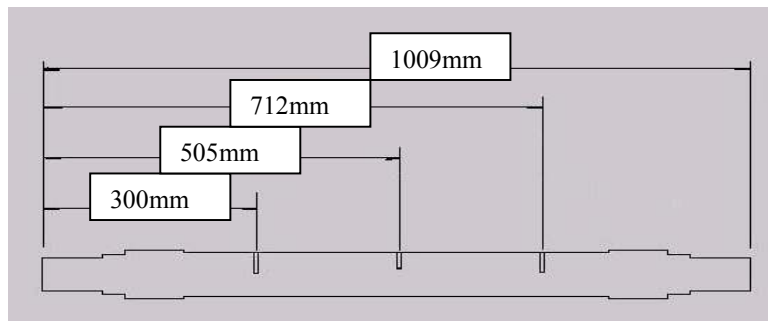


Figure 5-12: Real Defect Position

There are several reasons underline the difference between the measured value and the real value:

1. The defects are holds or cuts, not the ideal plane defect.
2. The lines connecting the probe and the defects are a little tilt to the shaft axis.
3. The defect positions are measured by hand, they are not precise.
4. The calibrate material used to get the ultrasonic velocity is not same as the shaft.

5.4 Mechanical Parts Organization

As the automatic testing system, Js-5 robot made by Kawasaki, Japan is included in this automatic system. A probe holder is designed as shown in figures below:

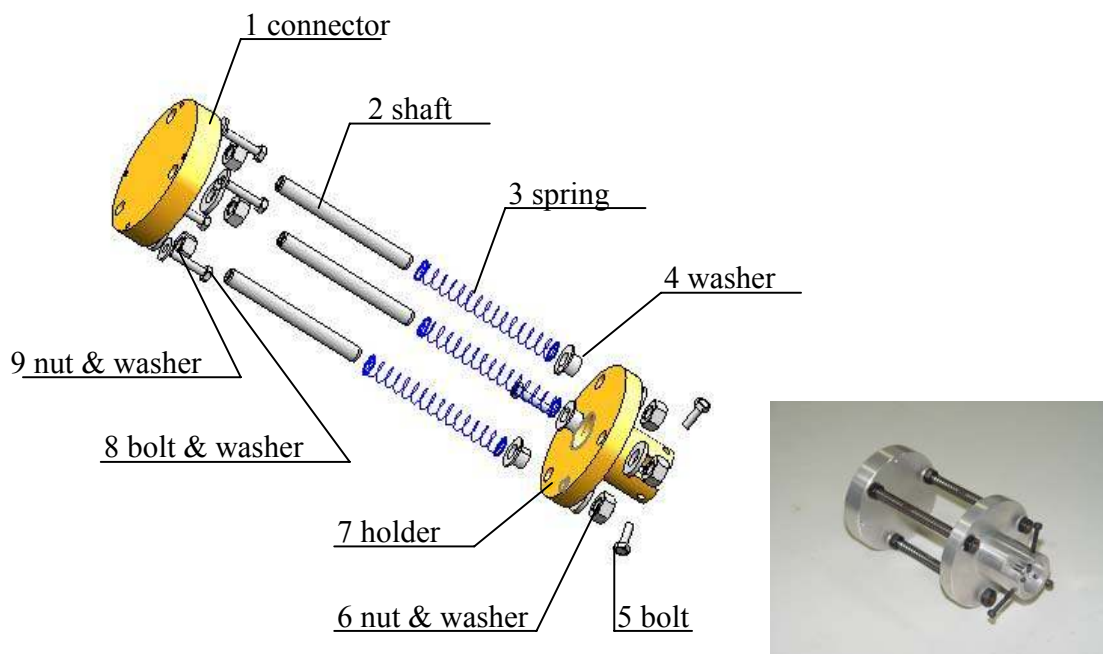


Figure 5-13: Probe Holder

The connector (1) is fixed to the end of the robot arm by the bolts (8). There are threads on both ends of the shaft (2). One end of the shaft is fixed to the connector; the other end is fixed to the nuts (6). The hold (7) can move along the shafts. The probe is put into the holder (7) and then three bolts (5) are used to fix the probe. The springs offer the force to

the holder (7), so that the probe fixed on the holder (7) can attach the test surface of the test piece with certain pressure. The spring modulus is 0.03kgf/mm. The spring rest length of the spring is 80mm. After assembly, the pre-form of the spring is 6mm. The movement range of the holder (7) is 0mm-20mm. So the minimum pressure force and the maximum pressure force are:

$$\text{Min} \quad 0.03\text{kgf/mm} \bullet 6\text{mm} \bullet 3 \bullet 9.8\text{N/kg} = 5.3\text{N}$$

$$\text{Max} \quad 0.03\text{kgf/mm} \bullet (6\text{mm} + 20\text{mm}) \bullet 3 \bullet 9.8\text{N/kg} = 22.9\text{N}$$

According to the investigation to the NDT operators and the NDT technical staff, the force in this range (5.3N – 22.9N) is similar to that applied to the probe by human operator in the practical testing normally.

The drawing of the connector and the holder are included in appendix B. The specification of the standard parts is also listed in appendix B.

The Kawasaki Js-5 is a 6-degree of freedom, 6-axis industrial robot. After installation of the probe holder and the probe, the probe can reach every point with any direction in the working range of the robot. The probe can also move along the desired path with correspondent program. See appendix C for the source code of the robot movement program. In the next section of this chapter, C-scan of welding point of the Panasonic compressor is carried out.

5.5 C-Scan Testing

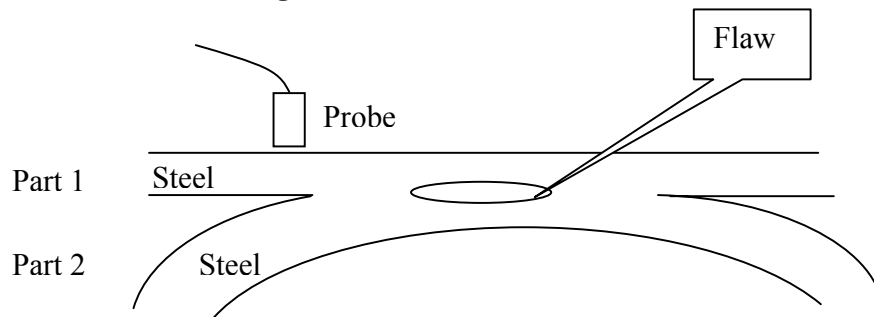


Figure 5-14: Welding Point

The C-Scan testing is to detect the flaw in the weld joints in the Panasonic refrigerator compressor casing. The weld joint cross-section is shown in Figure 5-14. The possible flaw location is in the welding surface between part 1 and part 2.

Before start C-Scan, the C-Scan range and the resolution need to be set. The resolution is the interval between every two A-Scan test point, shown in Figure 5-16.

Because the peak position can be found now, it is easy to do C-Scan. First, detect the position of the peak echoed from surface A. Then use robot to move the probe along the track shown in Figure 2-8. Save all the A-Scan data to file. After finishing scanning, the data can be express by a 3-D data array, shown in Figure 5-15. Every column data along z-axis is A-Scan Data. Because the probe is not strictly perpendicular to the actual weld plane, it is necessary to average all the date between plane B and plane C along the z-axis. Figure 5-17 shows the weld point C-Scan image. The flaw in the middle of the weld point is very clear.

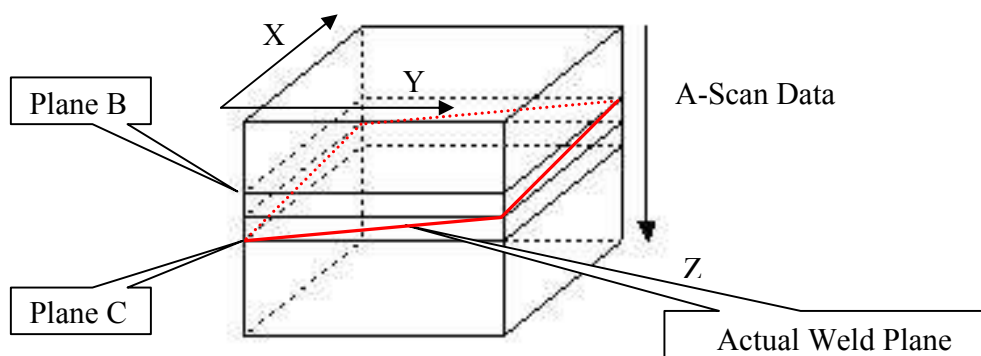


Figure 5-15: C-Scan Data Expression

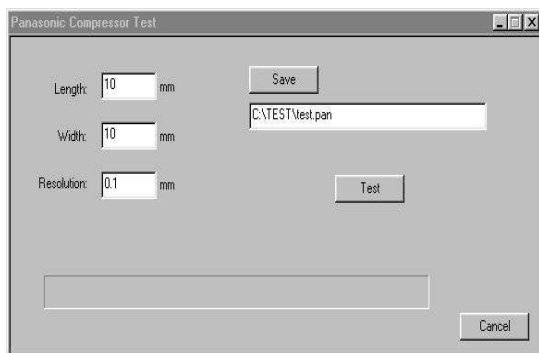


Figure 5-16: C-Scan Setting

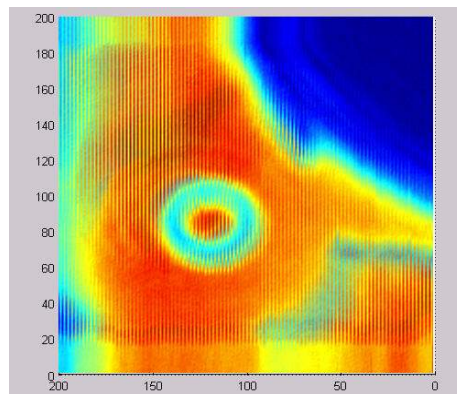


Figure 5-17: Weld Point C-Scan Image

Welds must be tested to ensure the safety. Currently, the common practice for testing is semi-destructive test, using a hammer hit the welds. The spot-weld quality and strength are primarily determined by the diameter of the weld nugget. The volume of steel melted during welding. If the melted region is greater than three-fourths of the whole weld point, the weld is judged to be good.

Roberts and Mason [61] analyzed the ultrasonic signal from the weld point. From the relationship of the amplitude of peak that comes from the gap and the amplitude of peak that comes from the back wall, they deduced a simple threshold. Welds with intermediate echo strength above the threshold will be classified as good.

With hardware development C-scan is applied in practice more frequently. The methods to analyze C-scan image automatically are image processing methods. Merazi Meksen et al [62] use high pass filter to get the edge of the C-scan image and then use randomized Hough transform to detect the elliptical contour of the defects in C-scan image and the parabola in the TOFD image.

As for the medical ultrasound B-mode image, active contour model (snake) [66], which is also used as motion tracking technology, is often applied [63][64]. Gradient Vector Flow

[65], an improved method of active contour mode, can provide better external force. This results in the advantages of the Gradient Vector Flow: insensitivity to initialization and the ability to move into boundary concavities. One of the disadvantages of active contour model is its inability to capture concavity. And more, tracking the merging and separation of many active contours is much difficult. Level Set approach can overcome the shortcomings of active contour because it possesses an inherent property of merging, splitting, preserving corners and cusps [67]. Markov Random Process should have its position in the automatic analysis of the C-scan image. Beside the approaches mentioned above, the traditional image processing methods, such as Shortest Cut and Watershed, can also be effective in some situation.

The content of this topic is relatively broad and is good to be a new project scope.

5.6 Weld Point Checking

For Panasonic compressor weld points, it is required to detect a gap within a given weld point. If there is a gap within the weld point, the ratio of gap area to the whole weld point area should be calculated.

Hough transform is one of the image processing techniques to isolate features of a given image. Since it could characterize the features in parametric form, it is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. The disadvantage of the Hough transform is the computational complexity. In terms of feature detections, the Hough space can be described in up to 4-dimensional space. Nevertheless, the Hough transform detection is only possible when the shape of the features is perfect. Although the Hough transform is relatively unaffected by additive noise (for example, pepper and salt noise), the distortion of curves could result in unacceptable transformed features. For the irregular boundary, the Hough transform is not applicable. Active contour models (snakes) which can be used to locate the object boundary serve as the alternative.

The snake moving to the region boundary is a function of internal and external forces. The key problem is to place the initial snake in proper position. The watershed algorithm will be used to locate the weld point. The initial snake can be derived from the watershed algorithm.

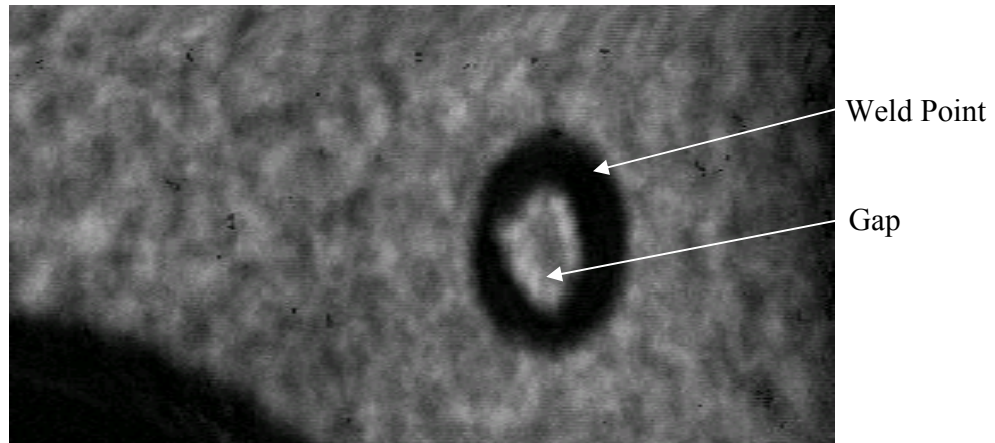


Figure 5-18: Original Weld Point Image

Figure 5-18 is the C-scan image of the weld point. For computation purpose, the original image is resized to 200×200 resolutions. It is not reliable to apply the watershed algorithm to the original image as the result might be “over-segmented”. The Gaussian smoothing filter is used to reduce the noise in the original image. Figure 5-19(A) is the filtered image after the convolution of original image with the Gaussian smoothing filter. The watershed image of filtered image is shown in Figure 5-19(C). It is still over segmented. The weld point position can not be located from Figure 5-19(C). The image needs to be smoother in order to remove the spurious watersheds and identify the weld point position. The morphological operations, gray level erosion and gray level dilution, are applied sequentially to achieve the objective. After erosion or dilution, the image needs to be reconstructed. Figure 5-19(B) is the image after erosion and dilution process. Figure 5-19(D) is the watershed image of the image after erosion and dilution process. The weld point position can be located easily. In some other cases, there are unwanted watersheds which might be connected with the image boundary. A recursive program is used to remove these unwanted watersheds.

A grid superimposes with the watershed where the intersection points of the grid and watershed form the watershed snake. The watershed snake is shown in Figure 5-20. Two snakes will be derived from the watershed snake: expanding snake and shrinking snake. The expanding snake expands and stops at the external boundary of the weld point. The shrinking snake shrinks and stops at the internal boundary of the weld point.

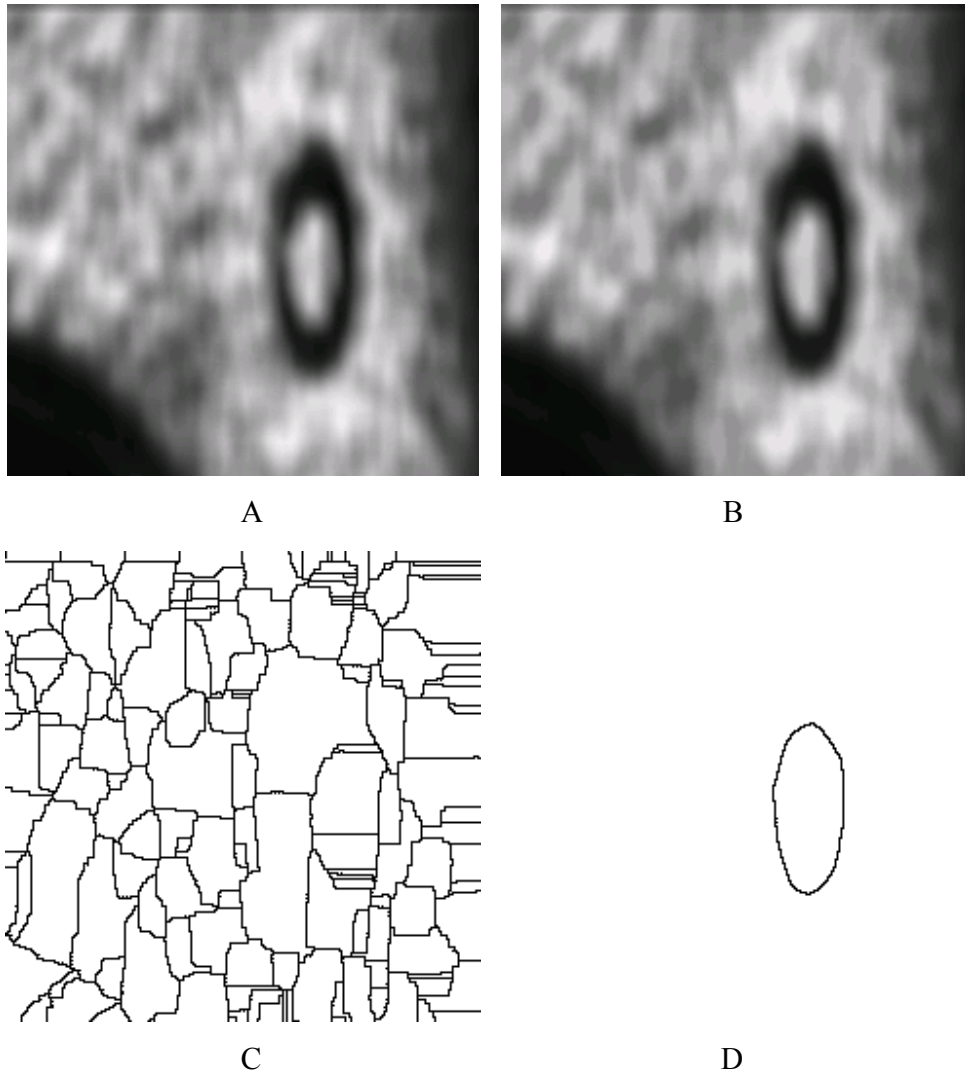


Figure 5-19: Detect weld point position using watershed: (A) Image filtered with Gaussian smoothing filter; (B) Image after eroding and diluting the filtered image; (C) Watershed of image A; (D) Watershed of image C;

The points of expanding snake can be located by moving the points of the watershed snake outside away from the watershed snake in the normal direction. The numerical calculation is show in the equation:

$$Expand \bar{P}_i = Watershed \bar{P}_i + D_{expand} * \bar{n} .$$

where:

$Expand \bar{P}_i$ is the i th point of the expanding snake;

$Watershed \bar{P}_i$ is the i th point of the watershed snake.

D_{expand} , is the expanding distance.

\bar{n} is a unit vector perpendicular to the vector $(Watershed \bar{P}_{i-1} - Watershed \bar{P}_{i+1})$

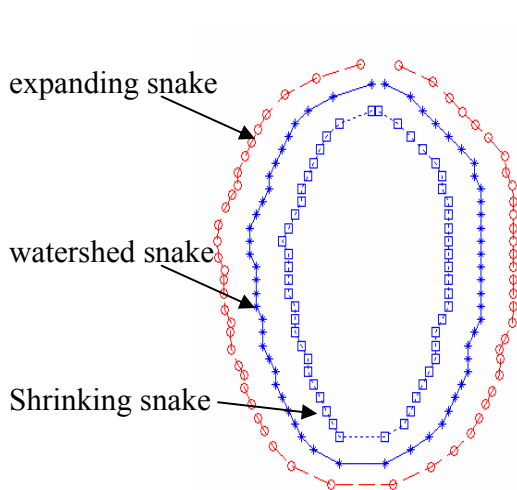


Figure 5-20: Watershed snake, expanding snake and shrinking snake

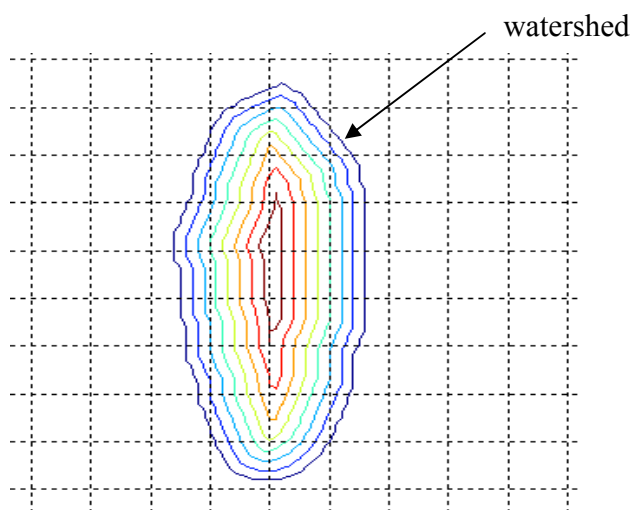


Figure 5-21: The contour of distance function

The shrinking snake can not be obtained from the same algorithm, because it might cause “swallow tail” in the shrinking snake. Instead, a distance function could be created from the watershed snake. Every pixel is labeled with the shortest distance to the watershed snake. The region outside the watershed snake is ignored because the shrinking snake must be encircled by the watershed snake. The result is the numerical solution of Eikonal equation. The shrinking distance, D_{shrink} , is introduced and a grid is used. The shrinking snake is formed by all the pixels in the grid that are labeled as D_{shrink} .

From previous paragraphs, the initial expanding and shrinking snake are formed. The internal force applied is the elastic and bending forces that represent the traditional active contours. The external force is basically formed by the gradient vector flow. The Canny edge detector is applied to the smoothed image. The Canny operator works in a multi-stage process. First of all, the image is smoothed by Gaussian smooth filter. Then a simple 2-D first order derivative operator is applied to the smoothed image to highlight regions of the image with high first order spatial derivatives. The third step is non-maximal suppression. The gradient magnitude of all non-local-maximal pixels will be set to zero. Figure 5-22 shows the final edge map. Subsequently, the gradient vector flow is calculated according to the algorithm proposed by Xu and Prince[65]. Since the task of the initial expanding snake is to locate the external boundary of the weld point, the expanding force is applied to the expanding snake. The expanding force will force the expanding snake to expand until it reaches the external boundary of the weld point. Similarly, the shrinking force is applied to the shrinking snake for the same reason.



Figure 5-22: Edge Map

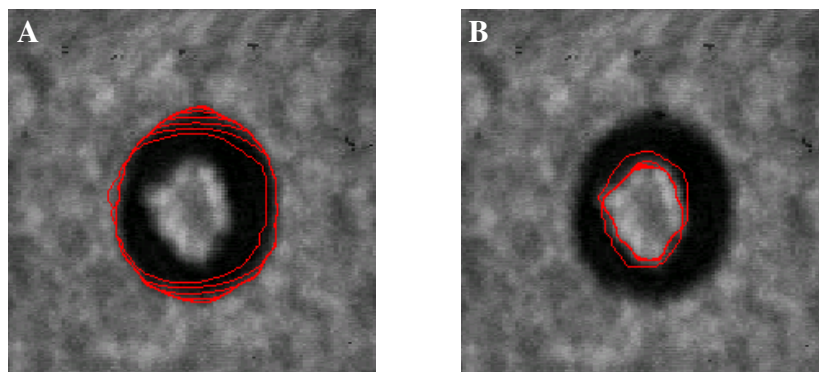


Figure 5-23: Convergence of expanding (A) and shrinking (B) snake

Figure 5-23(A) shows the convergence of the expanding snake and Figure 5-23(B) shows the convergence of the shrinking snake. Figure 5-24 shows the ultimate outcome. The area ratio of the gap to the whole weld point is 0.21.

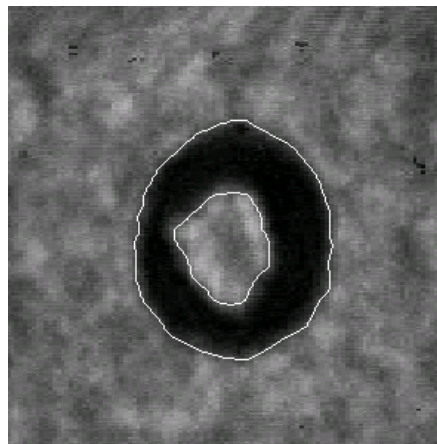


Figure 5-24: Weld point checking result

The working principle of this hybrid method using the watershed and active contours is described. However, there are still numerous parametric considerations in the implementation of this hybrid method, such as:

1. The width of the Gaussian smoothing filter can not to be too large. The smoothing should not break the feature of the weld point. Thus, the filter might not remove all the noises. Hence, the erosion and dilation techniques are adopted. The size of the mask used in the erosion and dilation can be adjusted to remove the residual noise.
2. The parameters (the width of the Gaussian smoothing filter and the threshold) of the Canny edge detector should be selected carefully to reduce the undesirable edges and keep the weld point boundary simultaneously.
3. The expanding and shrinking distance (D_{expand} and D_{shrink}) represent the initial expanding and shrinking snake. In order to make sure that the two initial snakes do not exceed the true weld point boundary, they should be small enough.
4. The elastic and bending forces should be adjusted. So the snakes can converge to the boundary with corners. Besides, the snakes should be prevented from leaking from the gaps of the edge map at the same time.

5. The expanding and shrinking force can accelerate the convergence speed of the expanding and shrinking snakes and force the snakes to pass through the spurious edges that might appear between the external and internal weld point boundary. However, the expanding and shrinking forces should not be too large. Otherwise, the expanding snake will pass the external boundary and expand to the whole image. The shrinking snake will shrink to a point without convergence to the internal boundary.

5.7 Software Structure Improvement

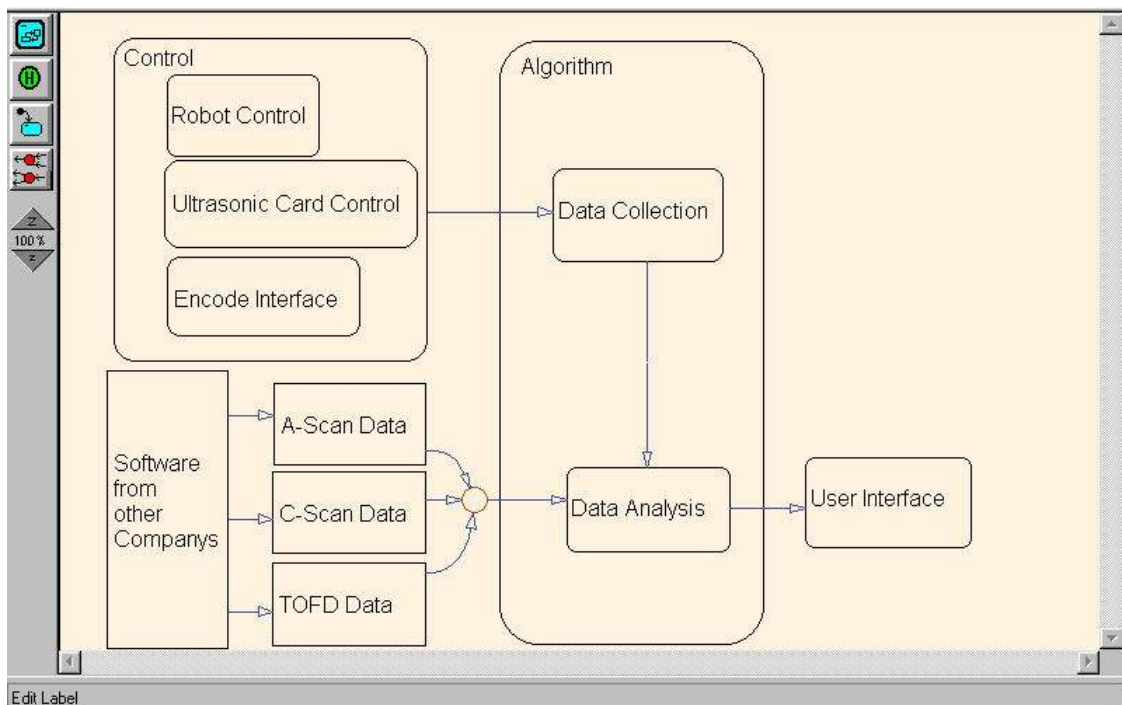


Figure 5-25: Software Structure Improvement

The software structure is shown in Figure 5-25. The automatic detection algorithm and automatic testing procedure should be isolated from other modules.

There are many kinds of ultrasonic testing hardware in the market, such as ultrasonic cards, robots and encoders. Separating the algorithm from those modules can save much

time when it is necessary to change certain hardware to fulfill some specific testing requirement.

And more, there are also many types of non-destructive testing software available in the market. The software can collect the testing data, such as A-Scan data, B-Scan data, C-Scan data and TOFD data. The algorithm orientates only the data. So it is possible to use the data directly without the work to collect the testing data.

In summary, the algorithm should be independent on all other modules. Especially the algorithm should be independent on the system platform, because Microsoft Window system is not a real time system. A real-time system is a system that must satisfy explicit (bounded) response-time constraints or risk severe consequences, including failure [59]. Although "software watch dog" can be used to check whether the software meets the time deadline or not, It is not guaranteed that it works in all situation. ("Software watch dog" is a background process that checks the state of other processes. If some process does not responses for a certain period, the "dog" will send an alert.) For some application situation, such as underwater NDT Testing, more reliability is desirable. The Windows System is not the proper application plat. The best solution is using a real Real-Time System (such as QNX [60]) to guarantee that all the requirements are met.

5.8 Chapter Summary

Two automatic procedures, the automatic calibration procedure and the automatic shaft testing procedure, are designed and realized in the SOFRA software. The experiment result of the automatic calibration is good. The experimental result of the testing on the aluminum shaft is very good (ref. Figure 5-11). SOFRA also shows the capability to do the C-Scan. This shows the possibility to automate other ultrasonic non-destructive testing procedures.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

There are many parameters that decide the output signal of the ultrasonic card. Some of the important parameters are:

1. **Sample Frequency:** From only the sense of the signal, the sample frequency is two times of the probe frequency at least. In practical, the sample frequency is 4 to 5 times of the probe frequency. However, when the buffer size is fixed, the one-shot testing range will be shorter with higher sample frequency.
2. **Buffer Size:** The larger, the better. Buffer size and sample frequency decide the one-shot testing range.
3. **Impulse Voltage and Width:** Because of the attenuation of the ultrasonic wave, higher impulse voltage and wider impulse width are necessary in order to get output signal strong enough.
4. **Averaging Capacity:** Averaging is the best way to eliminate the random noise. With high average capacity (1 million points per second), the output signal is clear, at the same time, the time to get such a signal is short.
5. **Rectification:** It is convenient to recognize the peaks in visualization after rectification. However, rectification makes much trouble to the filter.
6. **Filter:** Hardware filter is always much faster than software filter.

Three methods to enhance the signal are:

1. **Filter:** Because normal filter will cause phase shift, zero-phase filter is required in order to keep the peak position at the same place. The outcome of filtering a non-rectified signal should be better than that of filtering a rectified signal. The presence of the profile function means that there are frequency components besides the frequency labeled on the probe. The filter eliminates the frequency components in the stop band, hence the signal energy is lower and the peak is lower.

2. Correlation: Because of the presence of the profile function, the perfect reference signal for all peaks cannot be obtained. However, if the target is only to enhance some peaks, not for all peaks, a specific template signal can be chosen to reduce the noise.
3. Averaging: Averaging does not distort the original signal definitely. It is always the first choice if the average speed is fast enough.

In order to recognize the peak, feature selection is the first step:

1. The features decided by the AR model coefficients are very suitable to classify the non-truncated peaks and the peaks which amplitude is not too low. This can be seen from the cluster in Figure 4-1.
2. The other set of features is the statistics parameters. It can recognize the peaks with relative low amplitude.

Then, the statistical parameters are used as the features to do the classification task. The outcome of using the classification tree to classify is much better than the outcome of using neural network to classify. The classification tree is used to decide the value of the features, and create the classification rule according to the ultimate optimized classification tree. The classification rule does the classification task very well.

Software named SOFRA is developed. Two automatic procedures, automatic calibration and automatic shaft testing, are described. Both procedures can fulfill the corresponding task automatically.

6.2 Future Work

The method adopted to do the classification task is a kind of supervised learning method. The best way to do is to use an unsupervised learning method. Unsupervised learning method, will make the automatic non-destructive testing system independent on the hardware.

This automatic non-destructive system is the first and necessary step to develop a more complex automatic non-destructive testing system. The automatic detection of the size and shape of the flaw and the mode conversion of the ultrasonic wave will be the future work.

Mode conversion occurs when a wave encounters an interface between materials of different acoustic impedance and the incident angle is not normal to the interface. Shear waves travel more slowly than longitudinal waves so that it is possible to collect an electrical signal containing waves representing both shear waves and longitudinal waves by using a receiver. In accordance to the collected electrical signal, these two type of waves arrive at the receiver at different times. The time separation can be measured and then the distance to the defect can be calculated provided the longitudinal wave speed and Poisson's ratio or the shear wave speed are known.

The cylindrical shear waves are a feature of crack tip diffraction. A transducer with greater sensitivity to longitudinal waves is used as a transmitter, and a transducer with greater sensitivity to shear waves is used as receiver. The defect position can be located by putting the receiver to 2 or 3 different positions.

Many tedious but important tasks now handled by hand can be fulfilled by the automatic non-destructive testing system. For example, the automatic non-destructive testing system can be used to construct the distance amplitude correction (DAC) curves. DAC curves are constructed from the peak amplitude responses from reflectors of equal area at different distances in the same material. Such curves are plotted specifically for a flat-bottom hole target and engraved on a transparent plastic sheet for attachment to the CRT screen. Disk-shaped reflectors, side drilled holes and hemispherical bottom holes are used as equivalent reflectors when using contact probes. This technique is important because of the amplitude of ultrasonic pulses varies with distance from the probe, and this causes the echo from a constant reflector to vary with distance. DAC curves are needed for all kinds of probes to evaluate echoes of reflectors. With the DAC curves created and

saved to database by the automatic non-destructive system, the system should have the capability to detect both the defect position and the defect size and the defect form.

Other technologies used in deciding the defect size and the defect type include:

1. Maximum amplitude technology.
2. 20 db drop technology
3. 6 db drop technology
4. Defect time character analysis
5. Frequency analysis

There are many kinds of different defects with different character. As for the weld ports, 5 kinds welding defects exist mainly.

1. Gas pore
2. Group porosity
3. Isolated slag inclusion
4. Fine linear slag inclusion
5. Cracks

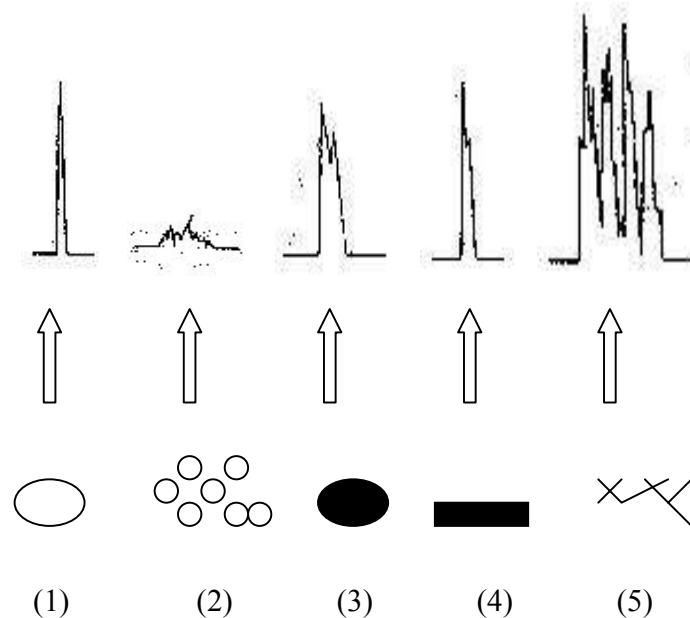


Figure 6-1: Weld Defect Types

Currently, these techniques used to decide the defect size and the defect type are applied manually in the industry. It is common that testing to the same part by different analysts produces different outcomes. The automatic non-destructive systems integrated with all the techniques mentioned would get more consistent outcomes and reduce the testing error at the same time.

The final automatic non-destructive system should have the capability to analyze the TOFD image, the working signal, and the signals from different defects.

Also expert knowledge will be added as much as possible to the automatic non-destructive testing system and Artificial Intelligence methods adopted to improve the performance of the automatic non-destructive testing system.

Bibliography

- [1] J. Golis, "ASNT Level 3 Study Guide Ultrasonic Method", The American Society for Nondestructive testing, Inc., ISBN 0-931403-29-4.
- [2] GS, Passi, Y Shoef, MV Kritsky, "Reducing the influence of human factors on the reliability of manual ultrasonic weld inspection", *Insight* 1995; 37 (10); 788-791
- [3] M. S. Obaidat, D. S. Abu-saymeh, "Performance comparison of neural networks and pattern recognition techniques for classifying ultrasonic transducers", *ACM* 1992 0-89791-502-X/92/0002/1234
- [4] M. S. Obaidat, H. Khalid, B. Sadoun, "Ultrasonic transducer characterization by neural network", *Journal of Information Sciences* 107 (1998) 195-215
- [5] M. Miller, B. Mi, A. Kita, I. Ume, "Development of automated real-time data acquisition system for robotic weld quality monitoring", *Mechatronics* 12 (2002) 1259-1269
- [6] K. M. Abd El-Ghany, M. M. Farag, "New software for remote ultrasonic scanning via the Internet", *NDT&E International*, 35 (2002) 1-8
- [7] P. Calmon, A. Lhemery, I. Lecoeur-Taibi, R. Raillon, L. Paradis, "Models for the computation of ultrasonic fields and their interaction with defects in realistic NDT configurations", *Nuclear Engineering and Design* 180 (1998) 271-283
- [8] W. E. Bond, D. C. St. Clair, M. M. Amirfathi, C. J. Mera, S. Aylward, "Neural network analysis of nondestructive evaluation patterns", *ACM* 1992 0-89791-502-X/92/0002/0643
- [9] J. Tang, Q. Ni, Y. Wang, "Multi-channel digital automatic ultrasonic detecting system", *Proceedings of the 3rd World Congress on Intelligent Control and Automation, 2000. Vol 4*, page 2572-2574
- [10] T. Masuda, K. Ito, "Review of application of automatic ultrasonic", *Insight: Non-Destructive Testing and Condition Monitoring*, v 40, n 4, Apr, 1998, page 260-264
- [11] A. McNab, I. Dunlop. "AI techniques applied to the classification of welding defects from automated NDT data", *Br J Nondestructive Testing* 1991;33; page11-18

- [12] A. R. Baker, C. G. Windsor, "The classification of defects form ultrasonic data using neural networks: the Hopfield method", *NDT Int* 1989;22(1);97-105
- [13] J. B. Santos, J. Perdigao, "Automatic defects classification system using spectral analysis and neural networks", *Ultrasonic International 93 Conference Proceedings*, 1993; page 763-766
- [14] R. Draï, F. Sellidj, M. Khelil, A. Benchaala, "Elaboration of some signal processing algorithms in ultrasonic techniques: application to materials NDT", *Ultrasonics* 38 (2000) 503-507
- [15] K. K. Yau, "Split-Spectrum Processing for Nondestructive Testing", *NDTnet August 1997, Vol.2 No.08* [http:// www.ndt.net/article/splitspec/splitspec.htm#fig26](http://www.ndt.net/article/splitspec/splitspec.htm#fig26)
- [16] J. Minker, "Logic-based artificial intelligence", By KLUWER ACADEMIC PUBLISHERS, ISBN 0-7923-7224
- [17] I. Cornwell, A. McNab, "Towards automated interpretation of ultrasonic NDT data", *NDT&E International* 32 (1999) 101-107
- [18] J. Jarmulak, E. J. H. Kerckhoffs, P. P. van't Veen, "Hybrid knowledge based system for automatic classification of B-scan images from ultrasonic rail inspection", *Tenth Conference on Innovative Applications of Artificial Intelligence*, 1998 page1121-1126
- [19] S. W. Lawson, G. A. Parker, "Automatic detection of defects in industrial ultrasound images using a neural network", *Proceedings of SPIE, Vol 2786, 1996*, page 37-47.
- [20] S. Cawson, "Ultrasonic testing and image processing for in-progress weld inspection", *NDTnet - April 1996, Vol.1 No.04*
- [21] Singapore Mass Rapid Transit, Job Specification.
- [22] 2003 http://www.ndt-ed.org/EducationResources/CommunityCollege/Ultrasonics/cc_ut_index.htm/
- [23] 2003 http://www.ndt.net/article/az/ut_idx.htm
- [24] D. D. Reynolds, "Engineering Principles of Acoustics Noise and Vibration Control", By Allyn and Bacon, Inc. ISBN(International) 0-205-07283-6
- [25] L. D. Rozenberg, "Physical Principles of Ultrasonic Technology", By Plenum Press, ISBN 0-306-35042-4

- [26] J. Kinney, "Statistics for science and engineering", By Addison Wesley, ISBN 0-201-43720-1
- [27] F. J. Taylor, "Principles of Signals and Systems", By McGraw-Hall, Inc., ISBN 0-07-911171-8
- [28] F. Taylor, J. Mellott, "Hands-on Digital Signal Processing", By McGraw-Hill, ISBN 0-07-912965-X
- [29] A. Cohen, "Biomedical Signal Processing", By CRC Press,
- [30] H. Demuth, M. Beale, MATLAB Neural Network Toolbox User's Guide, MathWorks, Inc. March 2001.
- [31] S. K. Pal, A. Pal, "Pattern Recognition from Classical to Modern Approaches", By World Scientific, ISBN 981-02-4684-6
- [32] D. F. Mix, "Random Signal Processing", By Prentice-Hall, Inc., ISBN 0-02-381852-2
- [33] G. J. Miao, M. A. Clements, "Digital Signal Processing and Statistical Classification", By Artech House, ISBN 1-58053-135
- [34] S. Raudgs, "Statistical and Neural Classifiers an integrated Approach to Design", By Springer, ISBN 1852332972
- [35] I. K. Sethi, A. K. Jain, "Artificial Neural Networks and Statistical Pattern Recognition", By North-Holland, ISBN 0 444 887410 7
- [36] A. V. Oppenheim, A. S. Willsky, S. H. Nawab, "Signals & System(Second Edition)", By Prentice Hall. ISBN 0-13-814757-4.
- [37] K. C. Chung , "Signals", By Prentice Hall, ISBN 981-4096-70-9.
- [38] A. V. Oppenheim, R. W. Schafer, "Discrete-Time Signal Processing", By Prentice-Hall International, Inc., ISBN 0-13-21677L-9
- [39] L. S. Kino, "Acoustic Waves: Devices, Imaging, and Analog Signal Processing", By Prentice-Hall, Inc. ISBN 0-13-003047-3
- [40] G. Zelniker, F. J. Taylor, "Advanced Digital Signal Processing", By Marcel Dekker, Inc., ISBN 0-8247-9145-2
- [41] J. Jan, "Digital Signal Filtering, Analysis and Restoration", By The institution of Electrical Engineers, ISBN 0 85296 760 8.

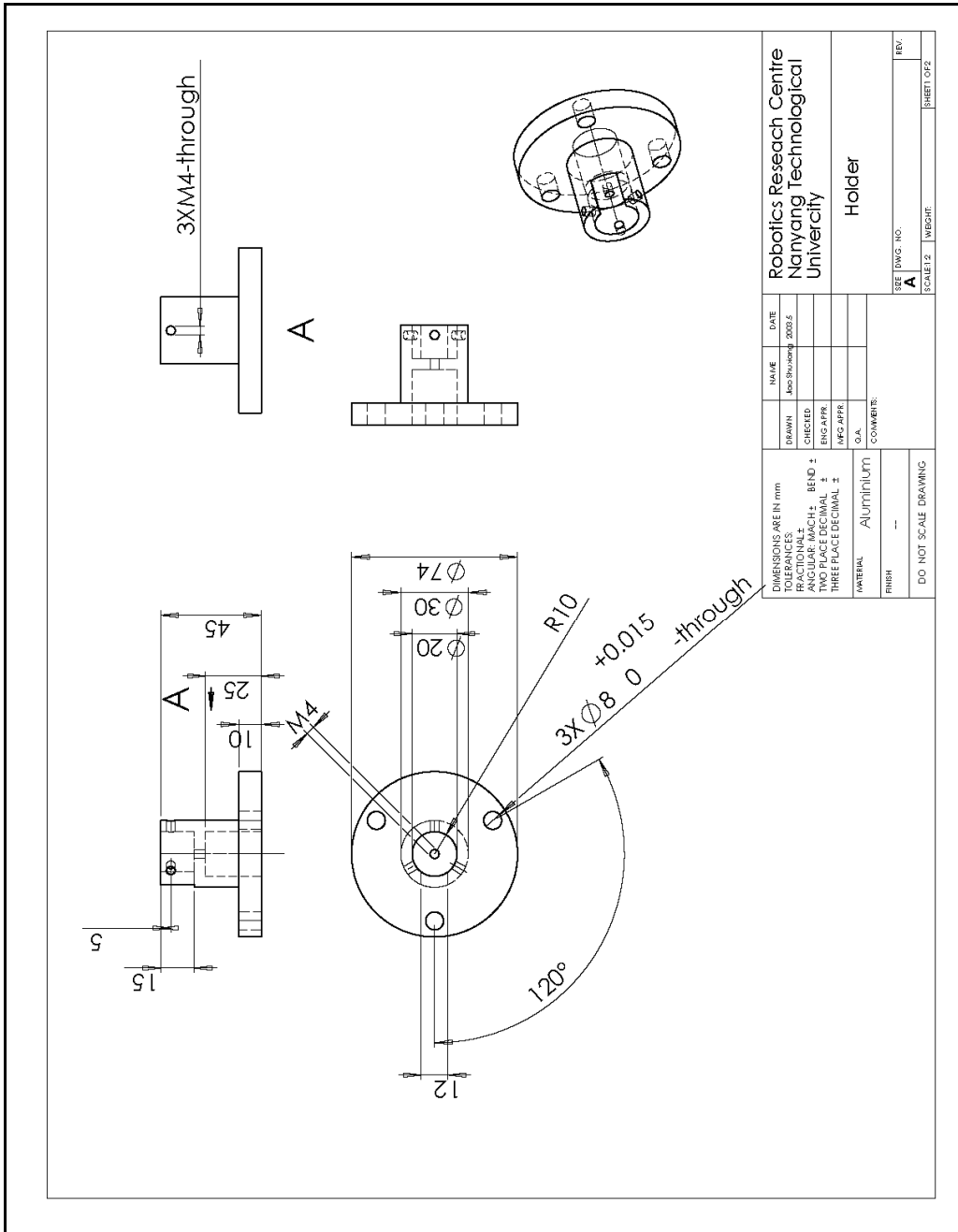
- [42] N. R. Pal, "Pattern Recognition in Soft Computing Paradigm", By World Scientific, ISBN 981-02-4491-6
- [43] C. W. Therrien, "Decision Estimation and Classification", By John Wiley & Sons, ISBN 0-471-83102-6
- [44] M. Friedman, A. Kandel, "Introduction to Pattern Recognition-Statistical, Structural, Neural and Fuzzy Logic Approaches", In *World Scientific*, ISBN 9810233124.
- [45] M. Smith, "Neural Networks For Statistical Modeling" By Van Nostrand Reinhold ISBN 0-442-01310-8.
- [46] S. Haykin, B. Kosko, "Intelligent Signal Processing", By IEEE Press, ISBN 0-7803-6010-6.
- [47] J. C. Principe, N. R. Euliano, W. C. Lefebvre, "Neural and adaptive Systems: Fundamentals Through Simulations", By John Wiley & Sons, Inc. ISBN 0-471-35167-9.
- [48] E. Micheli-Tzanakou, "Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence" By CRC Press. ISBN 0- 8493-2278-2.
- [49] H. Bunker, A. Kandel, "Hybrid Methods in Pattern Recognition", Series in Machine Perception and Artificial Intelligence, By World Scientific, ISBN 981-02-4832
- [50] R. Nevatia, "Machine Perception", By Prentice-Hall, Inc., ISBN 0-13-541904-2
- [51] B. D. Ripley , "Pattern Recognition and Neural Networks", In Cambridge University Press, ISBN 0 521 46086 7.
- [52] M. Anthony, P. L. Bartlett, "Neural Network Learning: Theoretical Foundations", By Cambridge University Press. ISBN 0 521 57353.
- [53] S. V. Kartalopoulos, "Understanding Neural Networks and Fuzzy Logic", By IEEE Press, ISBN 0-7803-1128,1996.
- [54] S. Haykin, "Neural Networks: A Comprehensive Foundation", By Macmillan College Publishing Company, Inc., ISBN 0-02-352761-7.
- [55] R. L. Kennedy, Y. Lee, B. V. Roy, C. D. Reed, R. P. Lippmann, "Solving Data Mining problems through Pattern Recognition", By prentice Hall, ISBN 0-13-095083-1
- [56] ASME Code Section V (Nondestructive Examination), *American Society of Mechanical Engineers*.

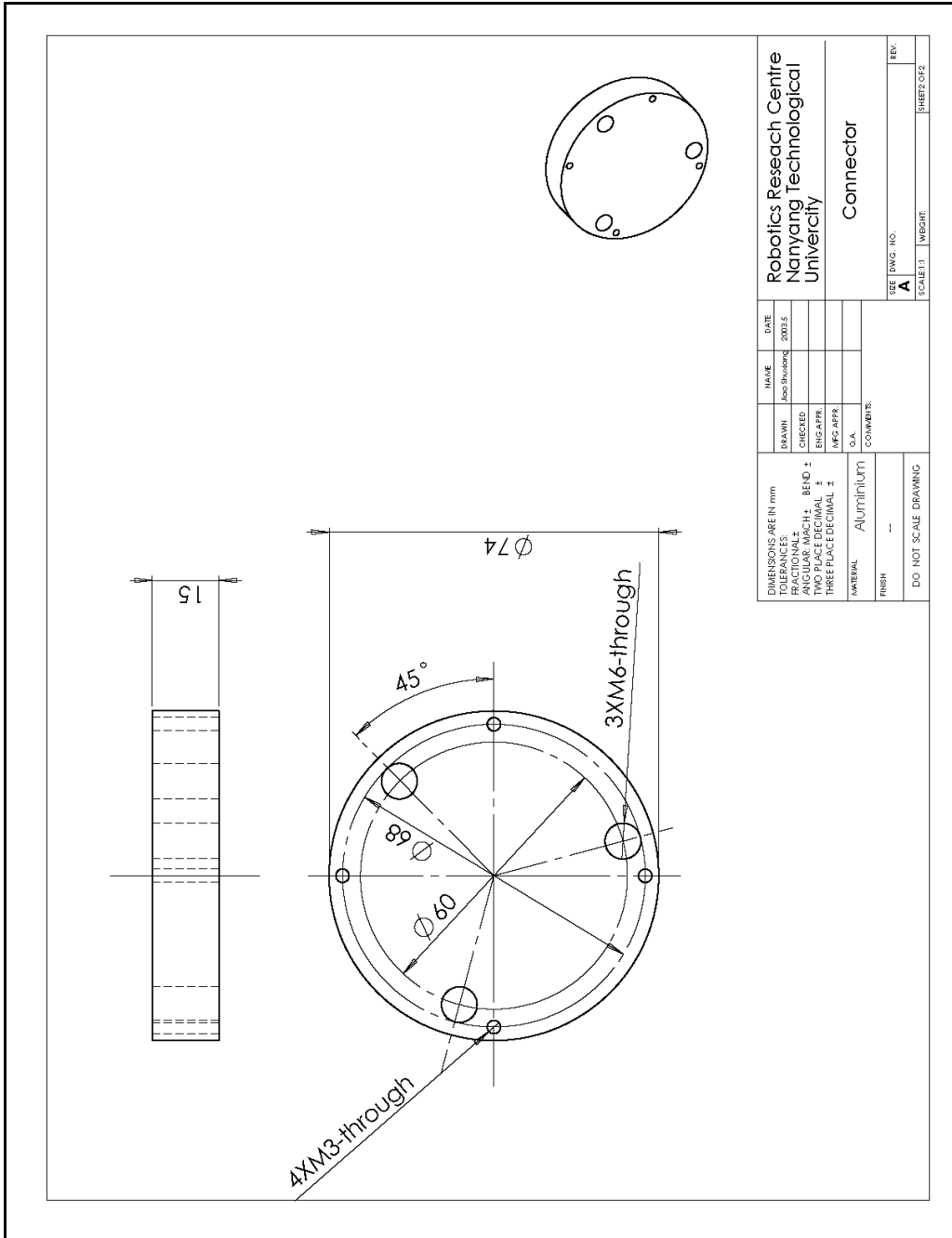
-
- [57] M. P. Norton, "Fundamental of Noise and Vibration Analysis for Engineers", By Cambridge University Press, ISBN 0-521-341485
- [58] O. V. Abramov, "High-Intensity Ultrasonics Theory and Industrial Applications", By Gordon and Breach Science publishers, ISBN 90-5699-041-1
- [59] P. A. Laplante, "Real-Time Systems Design and Analysis: An Engineer's Handbook. (Second Edition)", By IEEE Press. ISBN 0-7803-3400-0
- [60] R. Krten, "Getting Started with QNX4: a Guide for Realtime Programmers", By PARSE Software Devices. ISBN 0-9682501-0-6.
- [61] D. Roberts, K. Mason, C. Levis, "Ultrasonic Spot Weld Testing with Automatic Classification", *Science and Technology of Welding and Joining*, 2002 Vol. 7 No. 1 Page 47-50
- [62] T. M. Meksen, R. Draï, F. Sellidk, "Pattern Recognition in Ultrasonic Imagery Using the Hough Transform", *WCU 2003, Paris, September 7-10, 2003*, Page 753-756
- [63] F. Lefebvre, G. Gerger, P. Laugier, "Automatic Detection of the Boundary of the Calcaneus from Ultrasound Parametric Images Using an Active Contour Model; Clinical Assessment", *IEEE Transaction on Medical Imaging*, Vol. 17, No. 1, February 1988, Page 45-52
- [64] D. Cheng, S. T. Arno, K. Cheng, M. Sandrock, Q. Pu, H. Burkhardt, "Automatic Detection of the Intimal and the Adventitial Layers of the Common Carotid Artery Wall in Ultrasound B-Mode Images Using Sankes", *Journal of Computer Methods and Programs in Biomedicine*, Vol. 67, No. 1, 2002, page 27-37
- [65] C. Xu, J. L. Prince, "Snakes, Shapes, and Gradient Vector Flow", *IEEE Transactions on Image processing*, Vol. 7, No. 3, March 1998, Page 359-369
- [66] M. Kass, A. Witkin, D. Terzopoulos, "Sankes: Active Contour Models", *International Journal of Computer Vision*, Vol. 1, No. 4, 1987, Page 321-331
- [67] S. Kulkarni, B. N. Chatterji, "Shape Segmentation Based on Curve Evolution Using Level Sets", *National Conference on Communications*, Jan. 2002.

Appendix A: Source Code of SOFRA

The author had written the source code from scratch. However, if it were to be included here it would take more than 200 pages. It is included in the appendix CD. You can also email Dr. Stephen Brian Wong for the softcopy of the source code.

Appendix B: Drawing of Probe Holder





DIMENSIONS ARE IN mm		NAME	DATE
TOLERANCES:		Ang Shuohong	2003.5
FRACTIONAL	±0.1	DRAWN	
DECIMAL	±0.05	CHECKED	
TWO PLACE DECIMAL	±0.02	ENG APPR	
THREE PLACE DECIMAL	±0.01	MFG APPR	
		Q.A.	
MATERIAL: Aluminium		COMMENTS:	
FINISH: ---		SEE DWG. NO.:	
DO NOT SCALE DRAWING		A	
		SCALE: 1:1	
		INCHES:	
		REV:	
		SHEET: 012	

Robotics Research Centre
Nanyang Technological University

Connector

Table: Specification of standard parts in the probe holder:

Item No.	Description	Qty
1	Shaft, PSFRM6-105-F15-B9-P6-T5-Q6-SC25	3
2	Spring, WR-8-80	3
3	Oil free bushing, flange type, LFZF 6-8	3
4	Aluminum socket head cap, CBAL 4-15	4
4	Aluminum socket head cap, CBAL 3-15	4
5	Aluminum nuts, CBALN 4	6
6	Washer, SSWA 3-1.0	4
5	Washer, SSWA 6-1.0	6

Appendix C: Source Code of Robot Control

This target is achieved by two programs: the program running in robot and the program running in the computer that communicate with each other.

1. Running in Robot:

```

*****
.* ZROBOT.TYPE JS005F-E001_514 ( 2000-00-00 00:00 )
.* SOFTWARE VERSION 1504H6-5SE
.* SERVO VERSION
.* E03-S0D-JY10 : SID=0055 1997-03-07 JS005-E001 AD-HQ 1GE Servo software
.* E03-S0D-JY10 : SID=0055 1997-03-07 JS005-E001 AD-HQ 1GE Servo software
*****
PROGRAM excom()
  step = 15
  d = 75
;   TOOL wittprch
  SPEED 20
  PROTRESET
  PROTOCOL 2,0.02,2,10,10,1
  SETSIO 7:9600,1
  ACCURACY 0.01 ALWAYS
  ACCEL 5 ALWAYS
  DECEL 75 ALWAYS
  SPEED 5 ALWAYS
  CP OFF
  n = 1
; initialiation ok!
  WHILE (1) DO
    RECEIVE $command
    SEND "command received"
    i = 1;
100  RECEIVE $paratype
    SEND "paratype received"
    IF $paratype == "finished" GOTO 200
    RECEIVE $temp
    $para[i] = $temp
    SEND "PARA RECEIVED"
    IF $paratype == "VAL" THEN
      para[i] = VAL($para[i])
      $temp = "null"
    END

```

```
i = i+1
GOTO 100
200 IF ($command == "tdraw") THEN
TDRAW para[1],para[2],para[3]
$temp = "null"
END
IF ($command == "draw") THEN
DRAW para[1],para[2],para[3]
$temp = "null"
END
IF ($command == "align") THEN
ALIGN
$temp = "null"
END
IF ($command == "DRIVE") THEN
DRIVE para[1],para[2],para[3]
$temp = "null"
END
IF ($command == "ACCURACY") THEN
ACCURACY PARA[1],PARA[2]
$temp = "null"
END
IF ($command == "SPEED") THEN
SPEED PARA[1],PARA[2]
$temp = "null"
END
IF ($command == "ACCEL") THEN
ACCEL PARA[1],PARA[2]
$temp = "null"
END
IF ($command == "DECEL") THEN
DECEL PARA[1],PARA[2]
$temp = "null"
END
IF ($command == "HOME") THEN
HOME
$temp = "null"
END
IF ($command == "HOME2") THEN
HOME2
$temp = "null"
END
IF ($command == "SETHOME") THEN
HERE loc
SETHOME PARA[1]
$temp = "null"
```

```
END
IF ($command == "SET2HOME") THEN
SET2HOME PARA[1]
$temp = "null"
END
IF ($command == "TOOL") THEN
HERE loc
TOOL loc
$temp = "null"
END
IF ($command == "BASE") THEN
HERE loc
BASE loc
$temp = "null"
END
IF ($command == "GETCOORDINATE") THEN
HERE loc
DECOMPOSE location[1]=loc
$temp = $ENCODE(/F7.2,location[1])
SEND $temp
$temp = $ENCODE(/F7.2,location[2])
SEND $temp
$temp = $ENCODE(/F7.2,location[3])
SEND $temp
END
SEND "abcd"
IF ($command == "FINISH") THEN
BREAK
END
END
.END
```

2. Running in the computer:

```

/* Six files are included:
    robot1.h
    robot1.cpp
    decompose.h
    decompose.cpp
    Robot_Execute_Command.h
    Robot_Execute_Command.cpp
*/

/////////////////////////////////////////////////////////////////
//
// robot1.h: interface for the robot class.
//
/////////////////////////////////////////////////////////////////

#ifndef AFX_ROBOT1_H__E1A67546_1022_449E_A944_D1C1E617B3E7__INC
LUDED_
#define
AFX_ROBOT1_H__E1A67546_1022_449E_A944_D1C1E617B3E7__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif

class robot
{
public:
    robot();
    virtual ~robot();
    HANDLE hCom;
    HANDLE hfile;
    DCB dcb;
    COMMTIMEOUTS timeouts;
    CString response;

// Implementation
    DWORD Num;
    char ResponseData[20];
    char InputData[20];
    void robot::SendString(CString sendstring) ;
    void robot::ReceiveString() ;
    void robot::readchar(char ref);
    void robot::MoveXYZ(int axis, CString distance);
    void robot::MoveX(CString x_distance) ;

```

```

void robot::MoveY(CString y_distance) ;
void robot::MoveZ(CString z_distance) ;
void writefile_loop( HANDLE hCom, char * c, int write_char_number,
DWORD * Num, int id);
void robot::OpenNewPort();
};

#endif

/////////////////////////////////////////////////////////////////
//
// robot1.cpp: implementation of the robot class.
//
/////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "robot.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

/////////////////////////////////////////////////////////////////
// Construction/Destruction
/////////////////////////////////////////////////////////////////

robot::robot()
{
    hCom=CreateFile("COM1",
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        0,
        NULL
    );

    GetCommState(hCom, &dcb);

    dcb.BaudRate=9600;
    dcb.ByteSize=8;
    dcb.Parity=NOPARITY;
    dcb.StopBits=ONESTOPBIT;

```



```

dcb.fDtrControl=1;
dcb.fRtsControl=1;
dcb.XonLim=2048;
dcb.XoffLim=512;

SetCommState(hCom, &dcb);

timeouts.ReadIntervalTimeout=10;
timeouts.ReadTotalTimeoutMultiplier=10;
timeouts.ReadTotalTimeoutConstant=10;
timeouts.WriteTotalTimeoutMultiplier=10; //no
timeouts.WriteTotalTimeoutConstant=10;

SetCommTimeouts(hCom, &timeouts);
response="";

}

robot::~robot()
{
}
//send
void robot::SendString(CString sendstring)
{

    int i=0;
    char c,bcc=0;
// protocol for send data to Robotic
    c=5;
    robot::writefile_loop(hCom, &c, 1, &Num, NULL);
    c=6;
    readchar(c);
    c=2;
    robot::writefile_loop(hCom, &c, 1, &Num, NULL);

    for(i=0;i<sendstring.GetLength();++i)
    {
        c=sendstring[i];
        robot::writefile_loop(hCom, &c, 1, &Num, NULL);
        bcc^=c;
    }

    bcc^=0x03;
    c=0x03;
    robot::writefile_loop(hCom, &c, 1, &Num, NULL);

```

```

robot::writefile_loop(hCom, &bcc, 1, &Num, NULL);

c=6;
readchar(c);
c=0x04;
robot::writefile_loop(hCom, &c, 1, &Num, NULL);
}

//receive
void robot::ReceiveString()
{
    // TODO: Add your control notification handler code here
    char c;
    response="";
    c=5;
    readchar(c);

    c=0x06;
    robot::writefile_loop(hCom, &c, 1, &Num, NULL);
    c=2;
    readchar(c);
    //Sleep(300);
    c=0;
    while (c!=3)
    {
        if ((c!=0) && (Num==1)){
            response=response+c;
        }
        Num=0;
        ReadFile(hCom, &c, 1, &Num, NULL);
    }
    c=6;
    writefile_loop(hCom, &c, 1, &Num, NULL);
    c=4;
    readchar(c);
}

void robot::readchar(char ref)
{
    char c;
    int i=0;
    c=0;
    int re;

```

```
while (c!=ref) {
    re=ReadFile(hCom, &c, 1, &Num, NULL);
}
}

void robot::MoveXYZ(int axis, CString distance)
{
    switch (axis)
    {
    case 1:
        robot::SendString("xmove");
        robot::ReceiveString();
        if (robot::response.Find("xmove")!==-1)
        {
            robot::SendString(distance);
            robot::ReceiveString();

            if (robot::response.Find("xmove ok")!==-1){
                ::MessageBox(NULL, "Move x ok." ,"Success",MB_OK);
            }
        }
        break;
    case 2:
        robot::SendString("ymove");
        robot::ReceiveString();
        if (robot::response.Find("ymove")!==-1)
        {
            robot::SendString(distance);
            robot::ReceiveString();

            if (robot::response.Find("ymove ok")!==-1){
                ::MessageBox(NULL, "Move y ok." ,"Success",MB_OK);
            }
        }
        break;
    case 3:
        robot::SendString("zmove");
        robot::ReceiveString();
        if (robot::response.Find("zmove")!==-1)
        {
            robot::SendString(distance);
            robot::ReceiveString();

            if (robot::response.Find("zmove ok")!==-1){
                ::MessageBox(NULL, "Move z ok." ,"Success",MB_OK);
            }
        }
    }
}
```

```

        }
    }
    break;
default:
    MessageBox(NULL,"Please give the correct axle
number. ","Error",MB_OK);
}
}

void robot::MoveX(CString x_distance)
{
    robot::MoveXYZ(1,x_distance);
}

void robot::MoveY(CString y_distance)
{
    robot::MoveXYZ(2,y_distance);
}

void robot::MoveZ(CString z_distance)
{
    robot::MoveXYZ(3,z_distance);
}

void robot::writefile_loop( HANDLE hCom, char * c, int write_char_number,
    DWORD* Num, int id)
{
    int re;
    re=false;
    while (re==false){
        re=WriteFile(hCom, c, 1, Num, NULL);
    }
}

void robot::OpenNewPort()
{
    /*
        hCom=CreateFile("COM2",
            GENERIC_READ | GENERIC_WRITE,
            0,
            NULL,
            OPEN_EXISTING,
            0,
            NULL
        );
    */
}

```

```

*/
    GetCommState(hCom, &dcb);
    dcb.BaudRate=9600;
    dcb.ByteSize=8;
    dcb.Parity=NOPARITY;
    dcb.StopBits=ONESTOPBIT;
    dcb.fDtrControl=1;
    dcb.fRtsControl=1;
    dcb.XonLim=2048;
    dcb.XoffLim=512;

    SetCommState(hCom, &dcb);

    timeouts.ReadIntervalTimeout=10;
    timeouts.ReadTotalTimeoutMultiplier=10;
    timeouts.ReadTotalTimeoutConstant=10;
    timeouts.WriteTotalTimeoutMultiplier=10; //no
    timeouts.WriteTotalTimeoutConstant=10;

    SetCommTimeouts(hCom, &timeouts);
    response="";

}

////////////////////////////////////
//
// decompose.h: interface for the decompose class.
//
////////////////////////////////////

#ifndef AFX_DECOMPOSE_H__981CD692_4B07_11D7_B82F_00032D00B6B5_
_INCLUDED_
#define AFX_DECOMPOSE_H__981CD692_4B07_11D7_B82F_00032D00B6B5_
INCLUDED_

#ifdef _MSC_VER > 1000
#pragma once
#endif

class decompose
{
public:
    int PartNumber;
    CString Part[10];

```

```

    CString PartType[10];
    decompose();
    virtual ~decompose();
    void decompose::init();
    bool decompose::GetPart(CString CommandString);
};

#endif

////////////////////////////////////
//
// decompose.cpp: implementation of the decompose class.
//
////////////////////////////////////

#include "stdafx.h"
#include "decompose.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

decompose::decompose()
{
    PartNumber=0;
    for (int i=0;i<=9;i++){
        Part[i]="";
    }
}

decompose::~decompose()
{
}

void decompose::init()
{
    PartNumber=0;
    for (int i=0;i<=9;i++){
        Part[i]="";
        PartType[i]="";
    }
}

```

```

    }
}
bool decompose::GetPart(CString CommandString)
{
    int length;
    length=CommandString.GetLength();
    decompose::init();
    if (length==0){
        return true;
    }
    while ((Part[PartNumber]=="") && (length>0)){
        Part[PartNumber]=CommandString.SpanExcluding(" ");
        if (Part[PartNumber]==""){
            CommandString.TrimLeft(" ");
            if (CommandString.GetLength()==length){
                CommandString.TrimLeft(",");
            }
            length=CommandString.GetLength();
        }
    }
    }else{

        length=length-Part[PartNumber].GetLength();
        CommandString=CommandString.Right(length);
        PartNumber++;

    }

}

int i;
for (i=0;i<PartNumber;i++)
{

    //Part[i].FindOneOf("1234567890.")
    CString temp;
    temp=Part[i];
    temp.Replace("1","");
    temp.Replace("2","");
    temp.Replace("3","");
    temp.Replace("4","");
    temp.Replace("5","");
    temp.Replace("6","");
    temp.Replace("7","");
    temp.Replace("8","");
    temp.Replace("9","");
    temp.Replace("0","");
    temp.Replace(".", "");
    temp.Replace("-", "");

```

```

        if (temp==""){
            PartType[i]="val";
        }else{
            PartType[i]="str";
        }
    }
    return true;
}

/////////////////////////////////////////////////////////////////
//
// Robot_Execute_Command.h: interface for the Robot_Execute_Command class.
//
/////////////////////////////////////////////////////////////////

#ifndef AFX_ROBOT_EXECUTE_COMMAND_H__8DBB9D0B_3685_4E8D_B399_DC5CCBE9A11C__INCLUDED_
#define AFX_ROBOT_EXECUTE_COMMAND_H__8DBB9D0B_3685_4E8D_B399_DC5CCBE9A11C__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "robot.h"
#include "decompose.h"

class Robot_Execute_Command
{
public:
    Robot_Execute_Command();
    virtual ~Robot_Execute_Command();
    robot robot_operator;
    decompose decomposer;
    void Robot_Execute_Command_Do(CString command);
    CString Robot_Execute_Command::GetCoordinate();
};

#endif
// #ifndef AFX_ROBOT_EXECUTE_COMMAND_H__8DBB9D0B_3685_4E8D_B399_DC5CCBE9A11C__INCLUDED_
/////////////////////////////////////////////////////////////////

```

```

//
// Robot_Execute_Command.cpp: implementation of the Robot_Execute_Command class.
//
////////////////////////////////////

#include "stdafx.h"
#include "Robot_Execute_Command.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

Robot_Execute_Command::Robot_Execute_Command()
{
}

Robot_Execute_Command::~Robot_Execute_Command()
{
}

void Robot_Execute_Command::Robot_Execute_Command_Do(CString command)
{
    int i;
    decomposer.GetPart(command);
    robot_operator.SendString(decomposer.Part[0]);
    robot_operator.ReceiveString();
    if (robot_operator.response=="command received"){
        for (i=1; i<=(decomposer.PartNumber -1) ;i++){
            robot_operator.SendString(decomposer.PartType [i]);
            robot_operator.ReceiveString();
            if (robot_operator.response=="paratype received"){
                robot_operator.SendString(decomposer.Part[i]);
                robot_operator.ReceiveString();
            }
        }

        robot_operator.SendString("finished");
        robot_operator.ReceiveString();
    }
    robot_operator.ReceiveString();
}

```

```
}  
  
CString Robot_Execute_Command::GetCoordinate()  
{  
    CString CoordinateString;  
    CoordinateString="";  
    robot_operator.SendString("GETCOORDINATE");  
    robot_operator.ReceiveString();  
  
    robot_operator.SendString("finished");  
    robot_operator.ReceiveString();  
  
    robot_operator.ReceiveString();  
    CoordinateString=CoordinateString+","+robot_operator.response;  
    robot_operator.ReceiveString();  
    CoordinateString=CoordinateString+","+robot_operator.response;  
    robot_operator.ReceiveString();  
    CoordinateString=CoordinateString+","+robot_operator.response;  
    return (CoordinateString);  
}
```

Appendix D: Matlab Programs

File List:

My2.m	125
MatrixView1.m	136
corr.m	142
Newfunction.m	147
PerfectSignal.m	148
ClassificationTree.m	150
analysis_test_data.m.....	152
weld_point_checking.m.....	153
gvf_snake.m.....	162
circlar_hough_transform.m.....	165
cube_local_maximum.m.....	167
get_circle_x_y_coordinate.m.....	170
command_set.m.....	172

File Name: My2.m

Description: This file is the source code of Matlab program shown in figure 3-6 to figure 3-14.

Source Code:

```
function varargout = my2(varargin)
% MY2 Application M-file for my2.fig
%   FIG = MY2 launch my2 GUI.
%   MY2('callback_name', ...) invoke the named callback.

global i ;
global init_id;

if (init_id==1)

else
    i=100;
    init_id=1;
end
if nargin == 0 % LAUNCH GUI
    fig = openfig(mfilename,'reuse');
    set(fig,'Color', get(0,'defaultUiControlBackgroundColor'));
    initial_dir = pwd;
    handles = guihandles(fig);
    guidata(fig, handles);
    load_listbox(initial_dir,handles);
    if nargin > 0
        varargout{1} = fig;
    end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    initial_dir = varargin{1};
    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end
end

% -----
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
global i;
x=linspace(1,i);
y=linspace(1,i);
```

```

i =i+100;
newfig=corr;

% -----
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
global i;
x=linspace(1,i);
y=linspace(1,i);
i =i+100;

ff=findobj(' tag', ' axes3');
axes(ff);
plot(x,y);

% -----
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition DataFile;
global OriginalSignal FFTSignalMagnitude FFTSignalPhase;
global SampleFrequency;

SampleFrequency=40; %40M Hz

h=findobj(' tag', ' SampleNumber');
iSampleNumber = str2double(get(h, 'string'));
if isnan(iSampleNumber)
    errordlg('You must enter a numeric value', 'Bad Input', 'modal')
    return;
end

h=findobj(' tag', ' StartPostion');
iStartPosition = str2double(get(h, 'string'));
if isnan(iStartPosition)
    errordlg('You must enter a numeric value', 'Bad Input', 'modal')
    return;
end

h=findobj(' tag', ' EndPosition');
iEndPosition = str2double(get(h, 'string'));
if isnan(iEndPosition)
    errordlg('You must enter a numeric value', 'Bad Input', 'modal')
    return;
end

if (iSampleNumber<1 | ...
    iStartPosition<1 | ...
    iEndPosition<1 | ...
    iSampleNumber<iEndPosition | ...
    iStartPosition>iEndPosition )

```

```

        errordlg('You must enter correct number','Bad Input','on');
        return;
    end

    h=findobj('tag','DataFileList');
    temp=get(h,'string');
    temp1=get(h,'value');
    DataFile=temp(temp1);
    h=findobj('tag','Path');
    temp=get(h,'string');
    temp=char(temp);
    if temp(length(temp))~= '\ '
        temp=strcat(temp, '\ ');
    end
    DataFile=strcat(temp,DataFile);

    x=[1:iSampleNumber];
    %the red part is the part which will be FFT transformed.
    axes(handles.axesOriginalSignal);
    cla;
    hold on;
    plot(x(1:iStartPosition),OriginalSignal(1:iStartPosition),'color','B');
    plot(x(iStartPosition:iEndPosition),OriginalSignal(iStartPosition:iEndPosition),'color','R');
    plot(x(iEndPosition:iSampleNumber),OriginalSignal(iEndPosition:iSampleNumber),'color','B');
    hold off;
    set(handles.axesOriginalSignal,'box','on');

    %FFT the original signal, get the magnitude and the phase.
    SampleNumber=iEndPosition-iStartPosition+1;
    FFTSignal = fft(OriginalSignal(iStartPosition:iEndPosition),SampleNumber);
    FFTSignalMagnitude = abs(FFTSignal)/SampleNumber; % Magnitude
    FFTSignalPhase = (angle(roundn(FFTSignal,-6))); % phase
    %
    %plot the magnitude
    axes(handles.axesFFTMagnitude);
    cla;
    hold on;
    x=(0:SampleNumber-1)/SampleNumber*SampleFrequency;
    plot(x,FFTSignalMagnitude);
    ylabel('magnitude');
    xlabel('Frequency (MHz)');
    set(handles.axesFFTMagnitude,'box','on');
    hold off;

    %plot the phase
    axes(handles.axesFFTPhase);
    cla;

```

```

hold on;
stem(x, FFTSignalPhase);
set(gca, 'YTick', [-pi: pi/2 : pi]);
set(gca, 'YTickLabel', {-1:0.5:1});
xlabel(' Frequency (MHz)');
ylabel(' phase (\pi)');
%set(handles.axesFFTSignalPhase, 'box', 'on');
hold off;

% -----
% Read the current directory and sort the names
% -----
function load_listbox(dir_path, handles)
cd (dir_path)
dir_struct = dir(fullfile(dir_path, '*'));
temp=[dir_struct.isdir]';
i1=1;
i2=1;
for i=(1:size(temp, 1))
    if dir_struct(i).isdir==1
        directroy_list(i1)=dir_struct(i);
        i1=i1+1;
    else
        file_list(i2)=dir_struct(i);
        i2=i2+1;
    end
end

sorted_struct=[directroy_list file_list];
handles.file_names = {sorted_struct.name};
handles.is_dir = [sorted_struct.isdir];
handles.sorted_index = [sorted_struct.isdir];
guidata(handles.figure1, handles)

set(handles.DataFileList, 'String', handles.file_names, ...
    'Value', 1)
set(handles.Path, 'String', dir_path);

% -----
function varargout = DataFileList_Callback(h, eventdata, handles, varargin)
    index_selected = get(handles.DataFileList, 'Value');
    file_list = get(handles.DataFileList, 'String');
    filename = file_list{index_selected};
    if handles.is_dir(handles.sorted_index(index_selected))
        cd (filename)
        load_listbox(pwd, handles)
    end

% -----

```

```

function varargout = axesFFTMagnitude_ButtonDownFcn(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition DataFile;
global FFTSignalMagnitude;
global SampleFrequency;
ZoomDisplayFigure=figure;
axes;
cla;
SampleNumber=iEndPosition-iStartPosition+1;
x=(1:SampleNumber)/SampleNumber*SampleFrequency;
stem(x, FFTSignalMagnitude, 'fill');
ylabel('magnitude');
xlabel('Frequency (MHz)');

% -----
function varargout = axesFFTPhase_ButtonDownFcn(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition DataFile;
global FFTSignalPhase;
ZoomDisplayFigure=figure;
axes;
cla;
SampleNumber=iEndPosition-iStartPosition+1;
x=(1:SampleNumber);
stem(x, FFTSignalPhase, 'fill');
set(gca, 'YTick', [-pi: pi/2 : pi]);
set(gca, 'YTickLabel', {-1:0.5:1});
xlabel('Frequency (MHz)');
ylabel('phase (\pi)');

% -----
function varargout = axesOriginalSignal_ButtonDownFcn(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition DataFile;
global OriginalSignal;
ZoomDisplayFigure=figure;
axes;
cla;
hold on;
x=[1:iSampleNumber];
plot(x(1:iStartPosition), OriginalSignal(1:iStartPosition),...
      '-o',...
      'color', 'B',...
      'MarkerEdgeColor', 'k',...
      'MarkerFaceColor', 'm',...
      'MarkerSize', 2);
plot(x(iStartPosition:iEndPosition), OriginalSignal(iStartPosition:iEndPosition),...
      '-o',...
      'color', 'B',...
      'MarkerEdgeColor', 'k',...
      'MarkerFaceColor', 'm',...
      'MarkerSize', 2);

```

```

plot(x(iEndPosition:iSampleNumber),OriginalSignal(iEndPosition:iSampleNumber),...
      '-o',...
      'color','B',...
      'MarkerEdgeColor','k',...
      'MarkerFaceColor','m',...
      'MarkerSize',2);

hold off;
%-----
% LowPass Fir Fs=40MHz Fc=10MHz Order=8
% Num=[
%   -0.0000
%   -0.0990
%    0.0000
%    0.3335
%    0.5312
%    0.3335
%    0.0000
%   -0.0990
%   -0.0000
% ]';
% Den=1;
%-----
% LowPass Fir Fs=40MHz Fc=15MHz Order=8
% Num=[
%    0.0000
%    0.0643
%   -0.1467
%    0.2166
%    0.7318
%    0.2166
%   -0.1467
%    0.0643
%    0.0000]';
% Den=1;

%-----
% LowPass Fir Fs=40MHz Fc=15MHz Order=20
Num=[
  -0.0257
    0.0212
   -0.0000
  -0.0294
    0.0500
  -0.0435
    0.0000
    0.0753
   -0.1614

```

```

    0.2299
    0.7676
    0.2299
   -0.1614
    0.0753
    0.0000
   -0.0435
    0.0500
   -0.0294
   -0.0000
    0.0212
   -0.0257
];
Den=1;
%-----
%
% Num=[
%    0.125
%    0.125
%    0.125
%    0.125
%    0.125
%    0.125
%    0.125
%    0.125
%    ]';
% Den=1;
%-----
% Num=[
%   -0.0000
%   -0.0990
%    0.0000
%    0.3335
%    0.5312
%    0.3335
%    0.0000
%   -0.0990
%   -0.0000
% ]';
% Den=1;

%1
FiltFilteredSignal=filtfilt (Num, Den, OriginalSignal);
%2
%FiltFilteredSignal=filter (Num, Den, OriginalSignal);
%3
% FiltFilteredSignal=filter (Num, Den, OriginalSignal);
% FiltFilteredSignal=fliplr (FiltFilteredSignal);
% FiltFilteredSignal=filter (Num, Den, FiltFilteredSignal);

```

```

% FiltFiltedSignal=fliplr(FiltFiltedSignal);
hold on;
plot(1:length(OriginalSignal),FiltFiltedSignal,'m');
hold off;

% -----
function varargout = ZoomSignal_Callback(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition DataFile;
global OriginalSignal FFTSignalMagnitude FFTSignalPhase;
plot(x(iStartPosition:iEndPosition), OriginalSignal(iStartPosition:iEndPosition), 'color', 'r');
%
plot(x(iEndPosition:iSampleNumber), OriginalSignal(iEndPosition:iSampleNumber), 'color', 'b');
% hold off;
step=1;
temp=1;
while (iEndPosition<=length(OriginalSignal))
    %pushbutton3_Callback(h, eventdata, handles, varargin);
    %pushbutton_getdata_Callback(h, eventdata, handles, varargin);
    pushbutton3_Callback(h, eventdata, handles, varargin)
    iEndPosition=iEndPosition+step;
    iStartPosition=iStartPosition+step;
    set(handles.StartPostion, 'String', num2str(iStartPosition, '%4i'));
    set(handles.EndPosition, 'String', num2str(iEndPosition, '%4i'));
    pause(0.02);
    MatrixView1(1, FFTSignalMagnitude', temp-1);
    temp=temp+1;
end

% -----
% find the AR coefficients

function varargout = ZoomMagnitude_Callback(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition OriginalSignal;
global FFTSignalMagnitude;
SampleNumber=iEndPosition-iStartPosition+1;

SignalNeedAnalysis=OriginalSignal(iStartPosition:iEndPosition);

%AR_Coefficients=arcov(SignalNeedAnalysis, 3);
[b, a]=prony(SignalNeedAnalysis, 1, 3);

%a=exp(1)^0.5;
fvtool(b, a);

% -----
function varargout = ZoomPhase_Callback(h, eventdata, handles, varargin)
global iSampleNumber iStartPosition iEndPosition DataFile;

```

```

global FFTSignalPhase;
ZoomDisplayFigure=figure;
axes;
cla;
SampleNumber=iEndPosition-iStartPosition+1;
x=(1:SampleNumber);
stem(x,FFTSignalPhase,'fill');
set(gca,'YTick',[-pi: pi/2 : pi]);
set(gca,'YTickLabel',{-1:0.5:1});
xlabel('sample number');
ylabel('phase (\pi)');

% -----
function varargout = pushbutton_getdata_Callback(h, eventdata, handles, varargin)

global iSampleNumber iStartPosition iEndPosition DataFile;
global OriginalSignal FFTSignalMagnitude FFTSignalPhase;
global DataSegmentNumber AverageSegmentNumber;
global SampleFrequency;

%DataSegmentNumber=5;
AverageSegmentNumber=1;

h=findobj('tag','SegmentNumber');
DataSegmentNumber = str2double(get(h,'string'));
if isnan(DataSegmentNumber)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

h=findobj('tag','SampleNumber');
iSampleNumber = str2double(get(h,'string'));
if isnan(iSampleNumber)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

h=findobj('tag','StartPostion');
iStartPosition = str2double(get(h,'string'));
if isnan(iStartPosition)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

h=findobj('tag','EndPosition');
iEndPosition = str2double(get(h,'string'));
if isnan(iEndPosition)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

```

```

end
if (iSampleNumber<1 | ...
    iStartPosition<1 | ...
    iEndPosition<1 | ...
    iSampleNumber>iEndPosition | ...
    iStartPosition>iEndPosition )

    errordlg('You must enter correct number','Bad Input','on');
    return;
end

h=findobj('tag','DataFileList');
temp=get(h,'string');
temp1=get(h,'value');
DataFile=temp(temp1);
h=findobj('tag','Path');
temp=get(h,'string');
temp=char(temp);
if temp(length(temp))~= '\ '
    temp=strcat(temp, '\ ');
end
DataFile=strcat(temp,DataFile);

%open data file and read the original signal
try
    fp=fopen(char(DataFile),'r');

    %status = fseek(fp,8224*(DataSegmentNumber-1)+16,'bof');

    %OriginalSignal=fread(fp,iSampleNumber,'uint8');
    OriginalSignal=zeros(iSampleNumber,1);

    for ii=(1:AverageSegmentNumber)
        status = fseek(fp,8224*(DataSegmentNumber-1+ii-1)+16,'bof');
        TempSignal=fread(fp,iSampleNumber,'uint8');
        OriginalSignal=OriginalSignal+TempSignal;
    end
    OriginalSignal=OriginalSignal./AverageSegmentNumber;

    x=[1:iSampleNumber];
    err=fclose(fp);
    if (size(OriginalSignal)<iSampleNumber)
        iSampleNumber=size(OriginalSignal);
        if (iStartPosition>iSampleNumber | iEndPosition>iSampleNumber)
            errordlg('the data is less than your request','open data','on');
            return;
        end
    end
end
catch

```

```

        errordlg('open data file error','open data','on');
        return;
    end
    %plot the original signal
    %the red part is the part which will be FFT transformed.
    axes(handles.axesOriginalSignal);
    cla;
    hold on;
    plot(x(1:iStartPosition),OriginalSignal(1:iStartPosition),'color','B');
    plot(x(iStartPosition:iEndPosition),OriginalSignal(iStartPosition:iEndPosition),'color',
    ', 'R');
    plot(x(iEndPosition:iSampleNumber),OriginalSignal(iEndPosition:iSampleNumber),'color',
    'B');
    hold off;
    set(handles.axesOriginalSignal,'box','on');

    %FFT the original signal, get the magnitude and the phase.
    SampleNumber=iEndPosition-iStartPosition+1;
    FFTSignal = fft(OriginalSignal(iStartPosition:iEndPosition),SampleNumber);
    FFTSignalMagnitude = abs(FFTSignal)/SampleNumber; % Magnitude
    FFTSignalPhase = (angle(roundn(FFTSignal,-6))); % phase
    %
    %plot the magnitude
    axes(handles.axesFFTMagnitude);
    cla;
    hold on;
    x=(0:SampleNumber-1)/SampleNumber*SampleFrequency;
    plot(x,FFTSignalMagnitude);
    ylabel('magnitude');
    xlabel('Frequency (MHz)');
    set(handles.axesFFTMagnitude,'box','on');
    hold off;

    %plot the phase
    axes(handles.axesFFTPhase);
    cla;
    hold on;
    stem(x,FFTSignalPhase);
    set(gca,'YTick',[-pi: pi/2 : pi]);
    set(gca,'YTickLabel',{'-1:0.5:1'});
    xlabel('sample number');
    ylabel('phase (\pi)');
    %set(handles.axesFFTSignalPhase,'box','on');
    hold off;

    % -----
    function varargout = SegmentNumber_Callback(h, eventdata, handles, varargin)

```

File Name: MatrixView1.m

Description: The file is the source code of Matlab program shown in figure 3-16. this program is called when the button “Short Time FFT” (shown in figure 3-6) is clicked.

Source Code:

```
% -----
function fig_data = MatrixView1(block_name, NewData, TimeOrder)

global hfig;
if (TimeOrder==0)
    % CREATE_SCOPE Create new scope GUI
    % Initialize empty settings:hcolorbar
    fig_data.main = []; % until we move things here
    fig_data.menu = [];

    hfig = figure('numbertitle', 'off', ...
        'name', 'MatrixView', ... % 'menubar', 'none',
        'nextplot', 'add', ...
        'integerhandle', 'off', ...
        'renderer', 'painters', ...
        'doublebuffer', 'off', ...
        'HandleVisibility', 'callback');

    % Axis for the image:
    hax = axes('Parent', hfig, ...
        'DrawMode', 'fast', ...
        'Box', 'on', 'ticklength', [0 0]);

    % Axis for the colorbar:
    haxcbar = axes('Parent', hfig, ...
        'xtick', [], ... % turn off x ticks
        'xlim', [0 1], ...
        'yaxislocation', 'right', ...
        'Box', 'on', 'ticklength', [0 0]);

    % Set up image:
    himage = image('parent', hax, 'cdata', []);

    % Set up colorbar image:
    hcolorbar = image('parent', haxcbar, 'cdata', [], 'xdata', [0 1]);

    h = uicontrol('Parent', hfig, 'Style', 'pushbutton', 'String', 'New', ...
        'Position', [0 0 50 20], 'Callback', 'newfunction([]);');

    % Create a context menu:
```

```

mContext = uicontextmenu('parent',hfig);

% Establish settings for all structure fields:
fig_data.hfig = hfig;

% Store major settings:
fig_data.main.haxis = hax;
fig_data.main.himage = himage;
fig_data.menu.context = mContext;

% Store settings for axis zoom:
p = {'Units','Position'};
fig_data.main.axiszoom.off = {p, {'Normalized',[.1 .1 .8 .8]}};
fig_data.main.axiszoom.on = {p, {'Normalized',[ 0 0 1 1]}};
fig_data.main.axiszoom.cbar = {p, {'Normalized',[.13 .145 .645 .8]}};

% Copy colorbar data:
fig_data.main.colorbar.pos = {p, {'Normalized',[.8 .145 .075 .8]}};
fig_data.main.colorbar.h = hcolorbar;
fig_data.main.colorbar.hax = haxcbar;

set(hax, 'Position', fig_data.main.axiszoom.cbar{2}{2});
set(haxcbar, 'Position', fig_data.main.colorbar.pos{2}{2});

fig_data.params.YMin=0;
fig_data.params.YMax=32;
fig_data.params.CMap=jet(256);
fig_data.params.NumCols=length(NewData);
fig_data.params.NumRows=0;
fig_data.params.XLabel='40 MHz';
fig_data.params.YLabel='y';
fig_data.params.ZLabel='z';
fig_data.haxcolorbar=haxcbar;
fig_data.params.AxisColorbar=haxcbar;
fig_data.params.AxisZoom='on';
fig_data.hcolorbar=hcolorbar;

set(haxcbar, 'YLim', [5 ,50] );
N = size(fig_data.params.CMap,1);

set(fig_data.hcolorbar, 'CDataMapping','scaled', 'CData', (1:N), 'vis', 'on');

fig_data.params.AxisColorbar='on';
fig_data.params.AxisZoom='on';

% Record figure data:
set(hfig, 'UserData', fig_data);

% Assign context menu to the axis, lines, and grid:

```

```

    set([fig_data.main.haxis fig_data.main.himage], ...
        'ContextMenu', mContext);

    setup_axes(hfig, NewData);

    fig_data=get(hfig, 'UserData');
    update_image(fig_data, NewData); % one frame of data

else
    %hfig=findobj('name','MatrixView')
    fig_data=get(hfig, 'UserData');
    tempmax=max(NewData);
    tempmin=min(NewData);
    if fig_data.params.YMax<tempmax
        fig_data.params.YMax=tempmax;
    end

    if fig_data.params.YMin>tempmin;
        fig_data.params.YMin=tempmin;
    end

    update_image(fig_data, NewData); % one frame of data

set(fig_data.main.colorbar.hax,'YLim',[fig_data.params.YMin ,fig_data.params.YMax] );
set(fig_data.hcolorbar,'YData',[fig_data.params.YMin ,fig_data.params.YMax]);

end

% -----
function setup_axes(block_name, u)
% Setup viewer x- and y-axes

% Does not alter block_data
% u = input data (one matrix)

block_data = get(block_name,'UserData');
hfig      = block_data.hfig;
hax       = block_data.main.haxis;
himage    = block_data.main.himage;

haxclr    = block_data.haxcolorbar;
ylim      = block_data.params.YMin;

% Setup X-axis label:
% -----
xlabel = block_data.params.XLabel;
if ~isstr(xLabel), xLabel = 'X-Axis'; end
hxLabel = get(hax, 'XLabel');
set(hxLabel, 'String', xLabel);

```

```

% Setup Y-axis label:
% -----
yLabel = block_data.params.YLabel;
if ~isstr(yLabel), yLabel='Y-Axis'; end
hyLabel = get(hax,'YLabel');
set(hyLabel, 'String', yLabel);

% Setup Colorbar label:
% -----
cLabel = block_data.params.ZLabel;
if ~isstr(yLabel), cLabel='Z-Axis'; end
hyLabel = get(haxclr,'YLabel');
set(hyLabel, 'String', cLabel);
% Setup image data:
% -----
cmap = block_data.params.CMap;
set(hfig,'colormap', cmap);

if isa(u,'double'),
    % block_data.params.YMin
    % block_data.params.YMax
    if ~isempty(block_data.params.YMin) & ...
        ~isempty(block_data.params.YMax),
        set(hax, ...
            'clim',[block_data.params.YMin block_data.params.YMax], ...
            'climmode','auto');
        set(himage,'CDataMapping','Scaled');
    else
        set(hax, 'climmode','auto');
        set(himage,'CDataMapping','Direct');
    end
end

% Update colorbar image:
% -----
cbar      = block_data.params.AxisColorbar;
axiszoom  = block_data.params.AxisZoom;

useColorbar  = strcmp(cbar,'on');
noZoom       = strcmp(axiszoom,'off');
cbarANDnozoom = useColorbar & noZoom;

% Modify colorbar vertical axis limits to match
% the current scaling method in force:
N = size(cmap,1);
ylim = [1 N];
%set(haxcbar,'YLim',[fig_data.params.YMin ,fig_data.params.YMax] );
if isa(u,'double'),

```

```

    if ~isempty(block_data.params.YMin) & ...
        ~isempty(block_data.params.YMax),
        ylim = [block_data.params.YMin block_data.params.YMax];
    end
end

set(block_data.hcolorbar, ...
    'CDataMapping','scaled', ...
    'CData',(1:N)', ...
    'YData',ylim, ...
    'vis','on');

% Perform AxisZoom:
% -----
fig_data = get(hfig,'UserData');
if noZoom,
    % Turn off AxisZoom:

    % - turn on menus
    set(fig_data.menu.top,'vis','on');
    set(hfig,'menu','figure');

    % - reset axis position
    if useColorbar,
        set(hax, fig_data.main.axiszoom.cbar{:});
    else
        set(hax, fig_data.main.axiszoom.off{:});
    end
end
end

%-----
function update_image(block_data, NewData)
% Reshape for matrix input:
new_nrows = length(NewData) ./ block_data.params.NumCols;
u = reshape(NewData, new_nrows, block_data.params.NumCols);

ncols = block_data.params.NumCols;
nrows = block_data.params.NumRows+new_nrows;
block_data.params.NumRows=nrows;

set(block_data.main.haxis, ...
    'xlim',[-0.5 39.5], ...
    'ylim',[0.5 nrows+0.5], ...
    'zlimmode','manual');

xold = get(block_data.main.himage,'XData');
OldData=get(block_data.main.himage,'CData');
NewData=vertcat(OldData, NewData);

```

```
set(block_data.main.himage, ...
    'CData', NewData,...
    'XData', [0,39]);%, ...
    %' YData', [1,32]);

set(block_data.hfig, 'UserData', block_data);
```

File Name: corr.m

Description: The file is used to calculate the correlation of two signals. The program is called when the button “AutoCrossCorrelation” (shown in figure 3-6) is clicked.

Source Code:

```
%The orange signal is the signal (red section)
%got in my2.m

function varargout = corr(varargin)
% CORR Application M-file for corr.fig
% FIG = CORR launch corr GUI.
% CORR('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 24-Jun-2003 22:24:18

if nargin == 0 % LAUNCH GUI
    fig = openfig('corr.fig','reuse');
    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);
    if nargin > 0
        varargout{1} = fig;
    end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end
end

% -----
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
global i;
i=i+3;
axes;
plot(x,y);

% -----
function varargout = pushbutton_Correlation_Callback(h, eventdata, handles, varargin)
```

```

%compute(handles);

global OriginalSignal FFTSignalMagnitude;
global iSampleNumber iStartPosition iEndPosition ;

%overlay_number=50;
%CorrelationSegmentLength=100;

h=findobj('tag','editOverlay');
overlay_number = str2double(get(h,'string'));
if isnan(overlay_number)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

h=findobj('tag','editCorrelationLength');
CorrelationSegmentLength = str2double(get(h,'string'));
if isnan(CorrelationSegmentLength)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

axes(handles.axesCorrelation);
cla;

CorrelationStartPosition=iStartPosition;
CorrelationEndPosition=CorrelationStartPosition+CorrelationSegmentLength-1;
while (CorrelationEndPosition<=iEndPosition)

CorrelationSegment=OriginalSignal( CorrelationStartPosition:CorrelationEndPosition,1);
CorrelationResult=xcorr(CorrelationSegment');
hold on;
plot( (1-CorrelationSegmentLength:CorrelationSegmentLength-1),CorrelationResult);
hold off;
if (CorrelationStartPosition==iStartPosition)
    CorrelationResultMatrix=CorrelationResult;
else
    CorrelationResultMatrix=vertcat(CorrelationResultMatrix,CorrelationResult);
end
CorrelationStartPosition=CorrelationEndPosition-overlay_number+1;
CorrelationEndPosition=CorrelationStartPosition+CorrelationSegmentLength-1;
end
CorrelationMeanResult=mean(CorrelationResultMatrix);
CorrelationMeanResult=CorrelationMeanResult(size(CorrelationMeanResult,1),:);
hold on;
plot( (1-CorrelationSegmentLength:CorrelationSegmentLength-1),CorrelationMeanResult,'color','red');
hold off;

```

```

% -----
function varargout = pushbuttonCrossCorrelation_Callback(h, eventdata, handles,
varargin)
global OriginalSignal FFTSignalMagnitude;
global iSampleNumber iStartPosition iEndPosition ;

% ReferenceSignalStartPosition=75;
% ReferenceSignalEndPostion=89;

h=findobj('tag','edit_RefSignalStartPosition');
ReferenceSignalStartPosition = str2double(get(h,'string'));
if isnan(ReferenceSignalStartPosition)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

h=findobj('tag','edit_RefSignalEndPosition');
ReferenceSignalEndPostion = str2double(get(h,'string'));
if isnan(ReferenceSignalEndPostion)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

axes(handles.axesCorrelation);
cla;

OriginalSignalStartPosition=iStartPosition;
OriginalSignalEndPosition=iEndPosition;

OriginalSignal=OriginalSignal( OriginalSignalStartPosition:OriginalSignalEndPosition,1
);
OriginalSignalLength=length(OriginalSignal);

ReferenceSignal=OriginalSignal( ReferenceSignalStartPosition:ReferenceSignalEndPost
ion,1);

%below 3 lines
%create a reference signal template
% f=4M Hz Sample frequency=40M Hz Length= 1 circle
% if comment the 3 lines, you will get a reference signal template
% from the original signal, the start and end postion can be set
% on the figure.
h=findobj('tag','editCycles');
Cycles = str2double(get(h,'string'));
if isnan(Cycles)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

```

```

h=findobj('tag','checkboxRectify');
RectifyOrNot =get(h,'Selected');

SampleFrequency=40*10^6; % Sample frequency=40M Hz
SignalFrequency=5*10^6; % Signal frequency=5M Hz
%Cycles=3;
x_temp=[0:1/SampleFrequency:(1/SignalFrequency)*Cycles];
if strcmp(RectifyOrNot,'on')
    y=abs(sin(2*pi*SignalFrequency*x_temp));
else
    y=(sin(2*pi*SignalFrequency*x_temp))+1;
end
y=abs(sin(2*pi*SignalFrequency*x_temp));
ReferenceSignal=y;

%use profile function: uni distribution
%ProfileX=[-2:4/length(y):2-0.000000000001];
%ProfileY = normpdf(ProfileX,0,0.4);
%ReferenceSignal=ReferenceSignal.*ProfileY;

CorrelationResult=xcorr(OriginalSignal',ReferenceSignal');

hold on;
plot( (1:OriginalSignalLength),OriginalSignal,'b');
MaxValue=max(CorrelationResult);
CorrelationResult=CorrelationResult/MaxValue*255;
%plot( (1-OriginalSignalLength:OriginalSignalLength-1),CorrelationResult,'r');
plot( (1:OriginalSignalLength),CorrelationResult(OriginalSignalLength:OriginalSignalLength*2-1),'r');
hold off;

% -----
function varargout = pushbutton_WhiteNoiseCorr_Callback(h, eventdata, handles,
varargin)

global OriginalSignal FFTSignalMagnitude;
global iSampleNumber iStartPosition iEndPosition ;

%overlay_number=50;
%CorrelationSegmentLength=100;

h=findobj('tag','editOverlay');
overlay_number = str2double(get(h,'string'));
if isnan(overlay_number)
    errordlg('You must enter a numeric value','Bad Input','modal')
    return;
end

h=findobj('tag','editCorrelationLength');
```

```

CorrelationSegmentLength = str2double(get(h, 'string'));
if isnan(CorrelationSegmentLength)
    errordlg('You must enter a numeric value', 'Bad Input', 'modal')
    return;
end

axes(handles.axesCorrelation);
cla;

WhiteNoiseSignal=wgn(300,1,20);
CorrelationStartPosition=1;
CorrelationEndPosition=CorrelationStartPosition+CorrelationSegmentLength-1;
while (CorrelationEndPosition<=300)

CorrelationSegment=WhiteNoiseSignal( CorrelationStartPosition:CorrelationEndPosition,1
);
    CorrelationResult=xcorr(CorrelationSegment');
    hold on;
    plot( (1-CorrelationSegmentLength:CorrelationSegmentLength-1),CorrelationResult);
    hold off;
    if (CorrelationStartPosition==1)
        CorrelationResultMatrix=CorrelationResult;
    else
        CorrelationResultMatrix=vertcat(CorrelationResultMatrix,CorrelationResult);
    end
    CorrelationStartPosition=CorrelationEndPosition-overlay_number+1;
    CorrelationEndPosition=CorrelationStartPosition+CorrelationSegmentLength-1;
end
CorrelationMeanResult=mean(CorrelationResultMatrix);
CorrelationMeanResult=CorrelationMeanResult(size(CorrelationMeanResult,1),:);
hold on;
plot( (1-CorrelationSegmentLength:CorrelationSegmentLength-
1),CorrelationMeanResult,'color','red');
hold off;

```

File Name: Newfunction.m

Description: The code is called by Metrixview1.m

Source Code:

```
function newfunction(param1)

global hfig;
global filtereddata matrixdata;

fig_data=get(hfig,'Userdata');
matrixdata=get(fig_data.main.himage,'Cdata');

figure;
%mesh(matrixdata);
matrixdata=matrixdata(:,10);
matrixdata=matrixdata';
%filtereddata=[matrixdata
%             linspace(1, length(matrixdata), length(matrixdata))];
filtereddata=matrixdata;

%save('c:\temp\filtereddata', 'filtereddata');
figure;
plot(linspace(1, length(matrixdata), length(matrixdata)), matrixdata, ...
      '-o', ...
      'color', 'blue', ...
      'MarkerEdgeColor', 'k', ...
      'MarkerFaceColor', 'm', ...
      'MarkerSize', 2);
```

File Name: PerfectSignal.m

Description: This file is the source code of Matlab program shown in figure 3-3 and figure 3-4.

Source Code:

```
% 1. change
%     SampleFrequency : the more the better
% 2. non-rectification signal [y=(...)] better than the rectified signal [y=abs(...)]
%     (from frequency domain)
%
%
%

SubplotRows=2;
SubplotColumns=3;

Cycles=6;
SignalFrequency=5*1000000;
SampleFrequency=80*1000000;
SamplePoint=Cycles/SignalFrequency*SampleFrequency;
x=[0:SamplePoint];
y=abs(cos(x*2*pi*SignalFrequency/SampleFrequency));
figure(931);

subplot(SubplotRows,SubplotColumns,1);
plot(x,y);
title('5MHz Signal');

%need:
%     SamplePoint
ProfileX=[-2:4/SamplePoint:2];
ProfileY = normpdf(ProfileX,0,0.3);
subplot(SubplotRows,SubplotColumns,2);
plot(ProfileX,ProfileY);
title('Profile Function');

y=y.*ProfileY;
subplot(SubplotRows,SubplotColumns,3);
plot(x,y);
title('Simulated Signal')

%need
%     Signal : y
%     Sample Point : x
%     SamplePoint
%     SampleFrequency

FFTy = fft(y); % Compute DFT of x
m = abs(FFTy); p = (angle(FFTy)); % Magnitude and phase [unwrap(angle(FFTy));]
```

```

f=x/SamplePoint*SampleFrequency/1000000;
%figure(931);
subplot(SubplotRows,SubplotColumns,4);
plot(f,m); title(' Magnitude');
%set(gca,'XTick',[15 40 60 85]);
xlabel(' Frequency (MHz)');
ylabel(' Magnitude');

%figure(932);
subplot(SubplotRows,SubplotColumns,5);
plot(f,p/pi); title(' Phase');
xlabel(' Frequency (MHz)');
ylabel(' phase (\pi)'); %'phase (\pi)'
%set(gca,'XTick',[15 40 60 85]);

% Num: [1 10]
%      0 -0.0000 -0.0366  0.0000  0.2872  0.4987  0.2872  0.0000 -
0.0366 -0.0000
%
% Den: 1
%
%

%NoiseSignal=y*10+wgn(1,length(y),0);
NoiseSignal=awgn(y,10);
b=[ 0 -0.0000 -0.0366  0.0000  0.2872  0.4987  0.2872  0.0000 -0.0366
-0.0000];
FiltedSignal = (filtfilt(b,1,NoiseSignal));

figure(933)
plot([1:length(NoiseSignal)],NoiseSignal, [1:length(NoiseSignal)], FiltedSignal,'r')

```

File Name: ClassificationTree.m

Description: Using training data to create the classification tree (figure 4-6 to 4-8).

Source Code:

```
tempP=[];
tempT=[];

load('D:\transfer\mat_data\CTData\NN_NNP_NNT7.mat');
tempP=horzcat(tempP, NNP);
tempT=horzcat(tempT, NNT);
load('D:\transfer\mat_data\CTData\NN_NNP_NNT8.mat');
tempP=horzcat(tempP, NNP);
tempT=horzcat(tempT, NNT);
load('D:\transfer\mat_data\CTData\NN_NNP_NNT16.mat');
tempP=horzcat(tempP, NNP);
tempT=horzcat(tempT, NNT);
load('D:\transfer\mat_data\CTData\NN_NNP_NNT17.mat');
tempP=horzcat(tempP, NNP);
tempT=horzcat(tempT, NNT);
load('D:\transfer\mat_data\CTData\NN_NNP_NNT18.mat');
tempP=horzcat(tempP, NNP);
tempT=horzcat(tempT, NNT);

NNP=tempP;
NNT=tempT;
%NN_NNP_NNT8.mat
%NN_NNP_NNT7.mat
%NN_NNP_NNT16.mat
%NN_NNP_NNT17.mat
%NN_NNP_NNT18.mat

%plot3(NNP(1,:), NNP(2,:), NNP(3,:));
figure(10);
hold on;
for ii=1:(size(NNP,2))
    if (NNT(1,ii)==1);
        plot3(NNP(1,ii), NNP(2,ii), NNP(3,ii), 'r*');
        %_plot3(NNP(1,ii), NNP(2,ii), NNP(3,ii), 'LineStyle', 'or');
        %NOTE: For classification, the target should be char or string, or else
        treetest will fail.
        NNT(1,ii)='P';
    else
        plot3(NNP(1,ii), NNP(2,ii), NNP(3,ii), 'b*');
        NNT(1,ii)='N';
    end
    if (NNP(2,ii)<0)
        NNP(2,ii)=-NNP(2,ii);
    end
end
```

```
    NNP(2, ii)=NNP(2, ii)*100;
end
hold off;
NNP=NNP';
NNT=NNT';
T=treesfit(NNP,NNT,'method','classification','prune','on')
treedisp(T);

[c, s, n, best] = treetest(T,'crossvalidate',NNP,NNT);
figure(11);
%plot( n, c, 'r-', n(best+1), c(best+1), 'mo');
[mincost,minloc] = min(c);
plot(n, c, 'b-o', n, c+s, 'r:', n(best+1), c(best+1), 'mo', ...
     n, (mincost+s(minloc))*ones(size(n)), 'k--');

xlabel('Number of terminal nodes');
ylabel('Residual variance');

tmin = treeprune(T,'level',best);
treedisp(tmin);
```

File Name: analysis_test_data.m

Description: Reading peak data from .tst file created by SOFRA and then display the data (figure 5-7 to 5-10).

Source Code:

```
% format : analysis_test_data('007.tst')
% analysis_test_data('c:\007.tst')
function analysis_test_data(filename)
test_data=zeros(8200,75);
fp=fopen(filename,'r');
fseek(fp,0,'bof');

while(~feof(fp))
    point_number=fread(fp,1,'uint32');
    if (point_number==1023)
        point_number=fread(fp,1,'uint32');
        if (point_number~=0)
            for ii=(1:point_number)
                gain=fread(fp,1,'uint32');
                value=fread(fp,1,'uint32');
                position=fread(fp,1,'uint32');
                test_data(position,gain)=value;
                for iil=(gain:75)
                    test_data(position,iil)=value;
                end
            end
        end
    elseif (point_number==0)
        fseek(fp,4,'cof');
    else
        gain=fread(fp,1,'uint32')
        for ii=(1:point_number)
            value=fread(fp,1,'uint32');
            position=fread(fp,1,'uint32');
            test_data(position,gain)=value;
        end
    end
end

fclose(fp);

%-----
figure;
mesh(test_data);
%-----
figure;
spy(test_data');
```

File Name: weld_point_checking.m

Description: Check weld points using the watershed and the active contours

Source Code:

```
function check_weld()
global DL;

oima=['F:\weldpoints\size_fit_bracket1.jpg' ]
% oima=['F:\weldpoints\size_fit_bracket2.jpg' ]
% oima=['F:\weldpoints\size_fit_bracket3.jpg' ]
% oima=['F:\weldpoints\size_fit_bracket4.jpg' ]
% oima=['F:\weldpoints\size_fit_bracket5.jpg' ]
% oima=['F:\weldpoints\size_fit_bracket6.jpg' ]

% oima=['F:\weldpoints\size_fit_bracket7.jpg' ]
% oima=['F:\weldpoints\size_fit_bracket8.jpg' ]

    for file_number=1:size(oima,1)    % 5 no circle
        panasonic_weld_check(oima(file_number,:));
        %pause;
    End

%-----

function panasonic_weld_check(filename)
global DL;
close all;

%parameters
sizeX=250; sizeY=250;
se=strel('disk',3,0); %
histograph_eq=1;    % 0 no histogram equation
                    % 1 histogramy equation
LowPassfilter=1;    % 0 no low pass filter
                    % 1 low pass filter
LowPassFilterSize=15;%

%Read in the image and convert it to grayscale
[I,map] = imread(filename);
I=imresize(I,[sizeX sizeY]);
I=rgb2gray(I);

%inverse the image
%I=255-double(I);
%I=uint8(I);
s=I;
```

```

oimg=I;
figure(1), imshow(I), title('Original Image');
% histeq and low pass filter
if histogram_eq==1
    I=histeq(I);
end
%I=wiener2(I, [5 5]);
if LowPassfilter==1
    h = ones(LowPassFilterSize,LowPassFilterSize) / (LowPassFilterSize^2);
    I = imfilter(I,h,'replicate');
end
figure(2), imshow(I), title('after histeq or low pass filter');

DL = watershed(I);
figure(), imshow(DL), title('watershed');

%Use the gradient magnitude as the segmentation function
% [FX,FY] = gradient(double(I), 1);
% gradient_map=(FX.^2+FY.^2).^0.5;
% %gradient_map=histeq(gradient_map);
% figure(3), imshow(gradient_map, []), title('gradient map');

%Mark the foreground objects
% Io=imopen(I, se);
% %figure(7), imshow(Io), title('Image open');
%
% Ioc = imclose(Io, se);
% figure(8), imshow(Ioc), title('Image open close')
%

Ie=imerode(I, se);
figure(4), imshow(Ie), title('Ie imerode ');

Iobr=imreconstruct(Ie, I);
figure(5), imshow(Iobr), title('Iobr imreconstruct ');

Iobrd = imdilate(Iobr, se);
figure(6), imshow(Iobrd), title('Iobrd imdilate ');

Iobrcbr = imreconstruct(imcomplement(Iobrd), imcomplement(Iobr));
Iobrcbr = imcomplement(Iobrcbr);
figure(7), imshow(Iobrcbr), title('Iobrcbr')

%Get the weld point watershed
bw = im2bw(Iobrcbr, graythresh(Iobrcbr));
figure(11), imshow(bw), title('bw');
%close all;
D = bwdist(bw);
figure(110), imshow(D), title('D bwdist im2bw');

```

```

DL = watershed(D);
figure(111), imshow(DL), title('DL watershed bwdist im2bw');

% X = bwlabel(BW, 4)
%   pad_image=padarray(DL, [1 1], 0, 'both');
%   label_image=bwlabel(pad_image, 8);
%   imshow(label2rgb(pad_image));
%   label_image=label_image-label_image(1, 1);
%   imshow(label_image);
DL=im2bw(DL);
[x y]=size(DL);
%clear the border
DL(1, :)=1;
DL(x, :)=1;
DL(:, y)=1;
DL(:, 1)=1;
    for (jj=2:1:y-1)
        if DL(2, jj)==0
            temp=[2 jj];
            mark_delete(temp);
        end
        if DL(x-1, jj)==0
            temp=[x-1 jj];
            mark_delete(temp);
        end
    end
    for (ii=2:1:x-1)
        if DL(ii, 2)==0
            temp=[ii 2];
            mark_delete( temp);
        end
        if DL(ii, y-1)==0
            temp=[ii y-1];
            mark_delete(temp);
        end
    end
end
%
%   DL=~DL;
%   imshow(DL);
%   DL=bwdist(DL);
figure(); imshow(DL);
    if (size(DL, 1)*size(DL, 2)-sum(sum(double(DL))))<10
        disp(' No hole.   Program finished. ');
        return;
    end
%
%
snake_data=DL;
snake_point_left=[0;0];

```

```

snake_point_right=[0;0];
[x y]=size(DL);
for (ii=1:3:x)
    last=0;
    for (jj=1:1:y)
        if DL(ii, jj)==0;
            if (last==0)
                snake_point_left= horzcat(snake_point_left, [ii;jj]);

                end
                last_point=[ii;jj];
                last=last+1;
            end
            %DL(ii, jj)=0;
            %imshow(DL);
        end
        if (last>1)
            snake_point_right=horzcat(snake_point_right, last_point);
        end
    end
end

snake_point_left(:,1)=[];
snake_point_right(:,1)=[];
snake_point=horzcat(snake_point_left, fliplr(snake_point_right));

figure;
plot((snake_point(1, :))', (snake_point(2, :))', 'b*-');

%for every point point_i,
%caclulate the gradient of point_i,
%with point_i+1 and point_i-1
%then move the point_i along the gradient

expand_distance=5;
shrink_distance=5;
expand_snake=[0;0];
shrink_snake=[0;0];

point_number=size(snake_point, 2);
for ii=(1:1:point_number);
    point_i=snake_point(:, ii);
    switch ii
        case 1
            point_i_before=snake_point(:, point_number)
            point_i_next=snake_point(:, ii+1);
        case point_number
            point_i_before=snake_point(:, ii-1);
            point_i_next=snake_point(:, 1)
        otherwise

```

```

        point_i_before=snake_point(:, ii-1)
        point_i_next=snake_point(:, ii+1);
    end

    differential_i=(point_i_next - point_i_before);
    gradient_vect_i=[0-differential_i(2,1);differential_i(1,1)];
    unit_gradient_vect_i=gradient_vect_i ./
((gradient_vect_i(1,1)^2+gradient_vect_i(2,1)^2)^0.5)
    expand_snake=horzcat(expand_snake, point_i - (unit_gradient_vect_i .*
expand_distance));
    shrink_snake=horzcat(shrink_snake, point_i + (unit_gradient_vect_i .*
shrink_distance));

end
expand_snake(:,1)=[];
shrink_snake(:,1)=[];
hold on;
plot((expand_snake(1,:))', (expand_snake(2,:))', 'o--', 'color', 'red');
% plot((shrink_snake(1,:))', (shrink_snake(2,:))', 'color', 'blue');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%DL_shrink=imfill(~DL, [1 1], 8);
%figure(); imshow(DL_shrink);

DL_shrink=expand_line_image(size(DL), snake_point');
DL_shrink=im2bw(DL_shrink);
DL_shrink=imfill(DL_shrink, [1 1]);

DL_shrink=bwdist(DL_shrink);
figure(); [c h]=imcontour(DL_shrink);%imshow(DL_shrink);

%%<<
snake_data=DL_shrink;
snake_point_left=[0;0];
snake_point_right=[0;0];
[x y]=size(DL);
for (ii=1:3:x)
    last=0;
    for (jj=1:1:y)
        if
(DL_shrink(ii, jj)>=shrink_distance)&&(DL_shrink(ii, jj)<=(shrink_distance+1)) ;
            if (last==0)
                snake_point_left= horzcat(snake_point_left, [ii;jj]);

            end
            last_point=[ii;jj];
            last=last+1;
        end
    end
end
end

```

```

        %DL(ii, jj)=0;
        %imshow(DL);
    end
    if (last>1)
        snake_point_right=horzcat(snake_point_right, last_point);
    end
end
snake_point_left(:,1)=[];
snake_point_right(:,1)=[];
shrink_snake=horzcat(snake_point_left, fliplr(snake_point_right));
figure(9);
hold on;
%plot((expand_snake(1, :))', (expand_snake(2, :))', 'color', 'red');
plot((shrink_snake(1, :))', (shrink_snake(2, :))', 's:', 'color', 'blue');
%%>>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%the range of snake
max_xy=max(snake_point, [], 2);
min_xy=min(snake_point, [], 2);
max_x=max_xy(1, 1);
max_y=max_xy(2, 1);
min_x=min_xy(1, 1);
min_y=min_xy(2, 1);

%the part of image
image_max_x=max_x+(max_x-min_x);
image_min_x=min_x-(max_x-min_x);
image_max_y=max_y+(max_y-min_y);
image_min_y=min_y-(max_y-min_y);

image_size=size(oimg);
if image_min_x<1
    image_min_x=1;
end
if image_max_x>image_size(1,1)
    image_max_x=image_size(1,1);
end
if image_min_y<1
    image_min_y=1;
end
if image_max_y>image_size(1,2)
    image_max_y=image_size(1,2);
end

%chop the image
oimg=oimg(image_min_x:1:image_max_x, image_min_y:1:image_max_y);
imshow(oimg);

```

```

snake_point=[snake_point(1,:)-image_min_x;snake_point(2,:)-image_min_y];
expand_snake=[expand_snake(1,:)-image_min_x;expand_snake(2,:)-image_min_y];
shrink_snake=[shrink_snake(1,:)-image_min_x;shrink_snake(2,:)-image_min_y];
hold on;
plot((expand_snake(1,:))', (expand_snake(2,:))', 'color', 'red');
plot((shrink_snake(1,:))', (shrink_snake(2,:))', 'color', 'blue');

outer_snake =gvf_snake(oimg, expand_snake, 'expand');
outer_zone = roipoly(oimg, outer_snake(1,:), outer_snake(2,:));
outer_area=sum(sum(outer_zone));

inner_snake=gvf_snake(oimg, shrink_snake, 'shrink');
inner_zone = roipoly(oimg, inner_snake(1,:), inner_snake(2,:));
inner_area=sum(sum(inner_zone));

figure, imshow(oimg);
snakedisp(inner_snake(1,:), inner_snake(2,:), 'w')
snakedisp(outer_snake(1,:), outer_snake(2,:), 'w')
title(['Ratio : ' num2str(inner_area/outer_area)]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mark_delete(point_list)
global DL;
[x y]=size(point_list);

neighbours_all=[0 0];
for(ii=1:1:x)
    temp=point_list(ii,:);
    DL(temp(1,1),temp(1,2))=1;
    neighbours_all=vertcat(neighbours_all,get_neighbours(point_list(ii,:)));
end
neighbours_all(1,:)=[];
if size(neighbours_all,1)==0
    return
end
mark_delete( neighbours_all);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [neighbours]=get_neighbours( point)
global DL;
neighbours=[0 0];
neighbour=[-1 0 ;...
            -1 1 ;...
            0 1 ;...
            1 1 ;...
            1 0 ;...
            1 -1 ;...
            0 -1 ;...

```

```

        -1 -1 ];
    for (ii=1:1:8)
        temp=point+neighbour(ii,:);
        if DL(temp(1,1),temp(1,2))==0
            DL(temp(1,1),temp(1,2))=1;
            neighbours=vertcat(neighbours, temp);
        end
    end
    neighbours(1,:)=[];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%windows = windowy and bothe should be odd.
function [std_normalized]=std_normalize(img,windowx, windowy)
std_normalized=ones(size(img));
x_rad=(windowx-1)/2;
y_rad=(windowy-1)/2;

xstart=(windowx+1)/2;
ystart=(windowy+1)/2;
window = strel(' disk',windowx,0);
window_area=sum(sum(window));
[img_x,img_y]=size(img);

for (ii=xstart:1:(img_x-x_rad))
    for (jj=ystart:1:(img_y-y_rad))
        temp=img([(ii-x_rad):1:(ii+x_rad)], [(jj-y_rad):1:(jj+y_rad)]);
        avg=sum(sum(temp.*window))/window_area;
        std=((sum(sum(((temp-avg). *window).^2)))/window_area)^0.5;
        std_normalized(ii,jj)=std;
    end
end

max_std=max(max(std_normalized));
std_normalized=std_normalized./max_std;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [new_img]=expand_line_image(img_size, point_list)
new_img=zeros(img_size);
point_list=double(uint8(point_list));
point_list=vertcat(point_list, point_list(1,:));
point_number=size(point_list,1);
for (ii=1:1:(point_number-1))
    if (abs(point_list(ii,1)-point_list(ii+1,1))>abs(point_list(ii,2)-
point_list(ii+1,2)))
        if point_list(ii,1)<point_list(ii+1,1)
            xi=point_list(ii,1):1:point_list(ii+1,1)
        else
            xi=point_list(ii,1):-1:point_list(ii+1,1)
        end
    end
end

```

```
yi=interp1([point_list(ii,1),point_list(ii+1,1)], [point_list(ii,2),point_list(ii+1,2)]
,xi);
    for (jj=1:1:size(xi,2))
        new_img(uint8(xi(1,jj)),uint8(yi(1,jj)))=1;
    end

else
    if point_list(ii,2)<point_list(ii+1,2)
        yi=point_list(ii,2):1:point_list(ii+1,2)
    else
        yi=point_list(ii,2):-1:point_list(ii+1,2)
    end

xi=interp1([point_list(ii,2),point_list(ii+1,2)], [point_list(ii,1),point_list(ii+1,1)]
,yi);
    for (jj=1:1:size(xi,2))
        new_img(uint8(xi(1,jj)),uint8(yi(1,jj)))=1;
    end
end

end
new_img;
```


File Name: gvf_snake.m

Description: This file is the source code of Matlab program shown in figure 5-23.

Source Code:

```
function [snake]=gvf_snake(image_data, init_snake, action)

I=image_data;
imshow(I);

%ImageFiltered_Med =medfilt2(I, [10 10]);

ImageFiltered_Med=wiener2(I, [10 10]);
f = edge(ImageFiltered_Med, 'canny', 0.03);

%f=double(ImageFiltered_Med)/255;

% Compute its edge map
disp(' Compute edge map ...');

%f = 1-double(I)/255 ;

% Compute the GVF of the edge map f
disp(' Compute GVF ...');
[u,v] = GVF(f, 0.2, 80);
disp(' Nomalizing the GVF external force ...');
mag = sqrt(u.*u+v.*v);
%temp=std_normalize(image_data, 5, 5);
%mag=mag.*temp;
px = u./(mag+1e-10); py = v./(mag+1e-10);

% display the results
figure(1);
subplot(221); imshow(I); title(' test image');
subplot(222); imshow(f); title(' edge map');

% display the gradient of the edge map
f=double(f);
[fx, fy] = gradient(f);
subplot(223); quiver(fx, fy);
axis off; axis equal; axis 'ij'; % fix the axis
title(' edge map gradient');

% display the GVF
subplot(224); quiver(px, py);
axis off; axis equal; axis 'ij'; % fix the axis
title(' normalized GVF field');
```

```

% snake deformation
disp(' Press any key to start GVF snake deformation');
%pause;
subplot(221);
%image(((1-f)+1)*40);
axis('square', 'off');
%colormap(gray(64));

%-----
%get snake points
%-----

x=init_snake(2,:);
y=init_snake(1,:);

%[x,y] = snakeinterp(x,y,3,1); % this is for student version
% for professional version, use
[x,y] = snakeinterpl(x,y,1);

snakedisp(x,y,'r')
pause(1);

for i=1:30,
    %[x,y] = snakedeform(x,y,0.5,0.5,1,0.6,px,py,5);
    % snakedeform(x,y,2,1,1,0.8,-0.2,px,py,5)
    % snakedeform(x,y,2,1,1,0.8,0.2,px,py,5);
    if (action=='expand')
        [x,y] = snakedeform(x,y,2,1,1,0.8,-0.2,px,py,5);
    end
    if (action=='shrink')
        [x,y] = snakedeform(x,y,2,1,1,0.8,0.2,px,py,5);
    end

    %[x,y] = snakedeform2(x,y,2,1,1,0.8,10,px,py,5);

    %[x,y] = snakeinterp(x,y,3,1); % this is for student version
    % for professional version, use
    [x,y] = snakeinterpl(x,y,1);
    snakedisp(x,y,'r')
    title(['Deformation in progress, iter = ' num2str(i*1)])
    pause(0.1);
end

snake=[x';y'];

disp(' Press any key to display the final result');
pause;
cla;

```

```

colormap(gray(64)); image(((1-f)+1)*40); axis('square', 'off');

snakedisp(x, y, 'r')

title(['Final result, iter = ' num2str(i*5)]);
disp(' ');
disp(' Press any key to run the next example');
%pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%windows = windowx and windowy and both should be odd.
function [std_normalized]=std_normalize(img,windowx, windowy)
std_normalized=ones(size(img));
x_rad=(windowx-1)/2;
y_rad=(windowy-1)/2;

xstart=(windowx+1)/2;
ystart=(windowy+1)/2;
>window = strel('disk',x_rad,0);
>window=ones(windowx,windowy);
>window_area=sum(sum>window));
>[img_x, img_y]=size(img);

for (ii=xstart:1:(img_x-x_rad))
    for (jj=ystart:1:(img_y-y_rad))
        temp=img([(ii-x_rad):1:(ii+x_rad)], [(jj-y_rad):1:(jj+y_rad)]);
        avg=sum(sum((double(temp)).*>window))/window_area;
        std=((sum(sum(((double(temp)-avg)).*>window).^2))/window_area)^0.5;
        std_normalized(ii, jj)=std;
    end
end

max_std=max(max(std_normalized));
std_normalized=std_normalized./max_std;

```

File Name: `circlar_hough_transform.m`

Description: **Circlar Hough Transform.**

Source Code:

```
function [increase_cells]=circlar_hough_transform(oima, radius)

% radius=[8 4];
oima=padarray(oima, [max(radius) max(radius)], 0, 'both');

size_of_oima=size(oima);
oima_rows=size(oima, 1);
oima_rols=size(oima, 2);

[r_radius, c_radius]=size(radius);

radius_number=c_radius;

%h = waitbar(0, 'Circlar Hough Transform Please wait...');
increase_cells=zeros(oima_rows, oima_rols, radius_number);
total_compute_amount=oima_rows*oima_rols*radius_number;

for ii=1:oima_rows
    for jj=1:oima_rols

        %only for progress ID
        temp='circlar_hough x y';
        disp(temp);
        temp=[ii jj];
        disp(temp);

        for zz=1:radius_number

            if oima(ii, jj)==1
                [x, y]=get_circle_x_y_coordinate(ii, jj, radius(1, zz));
                %
                x=uint16(x);
                y=uint16(y);
                pt=size(x, 2);
                pii=1;
                while pii<pt
                    pjj=pii+1;
                    while pjj<=pt
                        if (x(1, pii)==x(1, pjj) & y(1, pii)==y(1, pjj))
                            x(:, pjj)=[];
                            y(:, pjj)=[];
                            pt=pt-1;
                        end
                    end
                end
            end
        end
    end
end
```

```
        else
            pjj=pjj+1;
        end

        end
        pii=pii+1;
    end
    %
    for tt=1:size(x,2)
increase_cells(x(1,tt),y(1,tt),zz)=increase_cells(x(1,tt),y(1,tt),zz)+1;
        end
        end
        %waitbar(ii*jj*zz/total_compute_amount);
    end
    end
end
%close(h);
```

File Name: cube_local_maximum.m

Description: Get the local maximum points in 3D

Source Code:

```
function [peaks_sorted]=cube_local_maximum(oima);
%
%get the maximum value in the image
% and the corespond position
%
% the flat area is also took as peaks
%
%
%
%
%
%
% [rs, cs, zs]=size(oima);
% [b, pos1]=max(oima);
% [c, pos2]=max(b);
% pos=[pos1(1, pos2), pos2]

[a, b, c]=size(oima);
threshold=sum(sum(sum(oima)))/(a*b*c);

oima=padarray(oima, [1 1 1], 0, 'both');

peaks=[0 0 0 0];

% [PeakValue  x y z
% PeakValue  x y z
% PeakValue  x y z
% PeakValue  x y z
%           ...]

[rs, cs, zs]=size(oima);

next_border=[];
neighbour=[-1 0 0;...
           -1 1 0;...
           0 1 0;...
           1 1 0;...
           1 0 0;...
           1 -1 0;...
           0 -1 0;...
           -1 -1 0;... % 8
           -1 0 -1;...
           -1 1 -1;...
```

```

    0 1 -1;...
    1 1 -1;...
    1 0 -1;...
    1 -1 -1;...
    0 -1 -1;...
    -1 -1 -1;...
    0 0 -1;... % 9
    -1 0 1;...
    -1 1 1;...
    0 1 1;...
    1 1 1;...
    1 0 1;...
    1 -1 1;...
    0 -1 1;...
    -1 -1 1;...
    0 0 1;... % 9
];

check_point=zeros(26,3);

%h = waitbar(0,'Find maximum Please wait...');

total_compute_amount=(rs-2)*(cs-2)*(zs-2);

for ii=2:(rs-1)
    for jj=2:(cs-1)

        %only for progress ID
        temp=' local max x y';
        disp(temp);
        temp=[ii jj];
        disp(temp);
        for hh=2:(zs-1)

            if oima(ii,jj,hh)>threshold
                point=[ii jj hh];
                check_globe_max=1;
                for zz=1:26
                    check_point(zz,:)=point + neighbour(zz,:);
                    if
(oima(check_point(zz,1),check_point(zz,2),check_point(zz,3))>oima(point(1,1),point(1,2)
),point(1,3)) ...
                        )
                        check_globe_max=0;
                        break;
                    end
                end
            end
        end
    end
end

```

```
        if check_globle_max==1

peaks=vertcat(peaks, [oima(point(1, 1), point(1, 2), point(1, 3)), point(1, 1), point(1, 2), poin
t(1, 3)]);
            end
        end

            %waitbar((ii-1)*(jj-1)*(hh-1)/total_compute_amount);
        end
    end
end

%close(h) ;

%temp=peaks(:, 1);
%[peaks_sorted, index_sorted]=sort(temp);
peaks(1, :)=[];
peaks_sorted=flipud(sortrows(peaks, 1));
peaks_sorted(:, [2 3 4])=peaks_sorted(:, [2 3 4])-1;
```


File Name: get_circle_x_y_coordinate.m

Description: Get the coordinates of the points which belong to a circle (centreX, centreY, r)

Source Code:

```
function [x,y] = get_circle_x_y_coordinate(centreX,centreY,r)

% r=23.3;
% centreX=50;
% centreY=60;

x=(0:r)+centreX;
y=(r^2-(x-centreX).^2).^0.5+centreY;

y1=(0:r)+centreY;
x1=(r^2-(y1-centreY).^2).^0.5+centreX;

% figure(1);plot(x,y);
% figure(2);plot(x1,y1);

t=size(x1,2);

for ii=1:size(x,2)
    jj=1;
    while jj<=t
        if abs(x(1,ii)-x1(jj))<0.00001
            x1(:,jj)=[];
            y1(:,jj)=[];
            t=t-1;break;
        end
        jj=jj+1;
    end
end

x=horzcat(x,x1);
y=horzcat(y,y1);

x_4=fliplr(x);
y_4=fliplr(2*centreY-y);
%
x_4a=x_4(1,2:size(x_4,2));
y_4a=y_4(1,2:size(y_4,2));

x_3=(2*centreX-x);
y_3=fliplr(y_4);
```

```
%  
x_3a=x_3(1,2:size(x_3,2));  
y_3a=y_3(1,2:size(y_3,2));  
  
x_2=fliplr(x_3);  
y_2=fliplr(y);  
  
x_2a=x_2(1,2:size(x_2,2)-1);  
y_2a=y_2(1,2:size(y_2,2)-1);  
  
% x_2=centreX-(x-centreX);  
% x=horzcat(x,x_2);  
% y=horzcat(y,(y));  
  
x=horzcat(x,x_4a);  
y=horzcat(y,y_4a);  
x=horzcat(x,x_3a);  
y=horzcat(y,y_3a);  
x=horzcat(x,x_2a);  
y=horzcat(y,y_2a);  
  
% figure(1);  
% plot(x,y);
```

File Name: command_set.m

Description: Used to validate Hough transform.

Source Code:

```

oima=zeros(50,50);
[x,y] = get_circle_x_y_coordinate(20,20,12);
for i=1:size(x,2)
oima(uint16(x(1,i)),uint16(y(1,i)))=1;
end
% [x,y] = get_circle_x_y_coordinate(25,25,22);
% for i=1:size(x,2)
% oima(uint16(x(1,i)),uint16(y(1,i)))=1;
% end
% [x,y] = get_circle_x_y_coordinate(20,20,18);
% for i=1:size(x,2)
% oima(uint16(x(1,i)),uint16(y(1,i)))=1;
% end
imshow(oima);
BW= oima;
radius=[12];
radius=[10:2:16];
cell_1=circular_hough_transform(BW, radius );
[z]=cube_local_maximum(cell_1);
circle_number=size(z,1);
if circle_number >40
    circle_number=40
end
z1=z(1:circle_number,:);
z1(:,[ 2 3 ])=z1(:,[ 2 3 ])-max(radius);
for i=1:size(z1,1)
    z1(i,4)=radius(1,z1(i,4));
end
z1
msg='finish';
disp(msg);
hold on;

[x,y] = get_circle_x_y_coordinate(z1(1,2),z(1,3),z1(1,4));
plot(x,y,'color','red');

```

Appendix E: Hardware Specification

Specification of JS-5/10 Robots:

(From http://www.kawasakirobotics.com/products/Robots/Retired/Retired_JSeries.asp)



Axes: 6 standard (7 optional)

Robot Arm Type: Articulated




Applications: Material handling, palletizing, machine loading, packaging, dispensing, assembly, water jet cutting.

Basic Description: The Kawasaki Js5/10 Series robots are intended for small duty payload, very high speed, and high accuracy applications. The Js5/10 robot utilizes a sealed construction in which it can work under the most difficult and undesirable conditions. All wiring and piping for motors, encoders, and tooling is built into the arm and hidden inside the robot structure to eliminate application interference.

Model	Axes	Payload	Horizontal Reach	Vertical Reach	Repeatability
Js-5	6 (7)*	5 (11)	1020 (40)	1280 (50)	±0.05 (±.002)
Js-10	6 (7)*	10 (22)	1475 (58)	1775 (70)	±0.1 (±.004)
* With traversing unit.		kg (lbs)	mm (in)	mm (in)	mm (in)

Specification of SFT 4001H-ISA Ultrasonic card:

(From http://www.sofratest.com/site/Sales/Catalog/SFT_4001H-ISA/sft_4001h-isa.html)

	
<h2>SFT 4001H - ISA Bus</h2>	
<p>Ultrasonic programmable PC Board Dual Channel with A/D Converter</p>	
* 2 pulsers and receivers on one board	
* 2 simultaneous channels 8 bits A/D converters	
* Suitable for applications and development in industrial and medical domains.	
* Comes with software for immediate settings.	
* Microsoft C libraries and Windows available	
* Ideal for OEM applications, multichannel operation	
	
Technical Specifications	Downloading
<p><i>If you have technical questions, need price information's , want to place an order or to know the contact with the Distributor in your country, please contact SOFRATEST head office directly.</i></p>	

Appendix F: Source Code CD