

2005

Sonar sensor interpretation for ectogeneous robots

Wen Gao

College of William and Mary - Virginia Institute of Marine Science

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Acoustics, Dynamics, and Controls Commons](#), and the [Artificial Intelligence and Robotics Commons](#)

Recommended Citation

Gao, Wen, "Sonar sensor interpretation for ectogeneous robots" (2005). *Dissertations, Theses, and Masters Projects*. Paper 1539616656.

<https://dx.doi.org/doi:10.25773/v5-dpv5-4238>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

SONAR SENSOR INTERPRETATION
FOR ECTOGENEOUS ROBOTS

A Dissertation

Presented to

The Faculty of the Department of Applied Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

By

Wen Gao


2005

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of

The requirements for the degree of

Doctor of Philosophy



Wen Gao

Approved by the Committee, April 2005



Mark K. Hinders, Chair



Gregory D. Smith



Zia-ur Rahman



Weizhen Mao
Computer Science

To Professor Mark Hinders

To Yongming Cui

To Mom and Dad, Ying Sun and Zhong Gao

Table of Contents

	Page
Acknowledgements	viii
List of Tables	ix
List of Figures	x
Abstract	xix
Chapter 1. Introduction	1
1.1 Ecotobot	1
1.2 Housework Robots	2
1.3 DARPA Grand Challenge	5
1.4 Overview of the Biosonar Research	6
<i>1.4.1 How Bats and Dolphins Detect Objects</i>	6
<i>1.4.2 Ultrasonic Systems That Mimic Bats</i>	9
<i>1.4.3 Robotic Ultrasound Systems</i>	11
<i>1.4.4 More Sophisticated Detection and Map Building Techniques</i>	12
<i>1.4.5 Image Fusing Technologies</i>	14
1.5 Summary	15
Chapter 2. Four Generations of Robotic Systems	16
2.1 First Generation of Robotic Scanning System	16
<i>2.1.1 Apparatus</i>	16
<i>2.1.2 Sample Results from the First Generation — B-scan Images</i>	21
<i>2.1.3 Limitations of the first generation apparatus</i>	21

	Page
2.2 Second Generation of Robotic Scanning System	24
2.2.1 <i>Apparatus</i>	24
2.2.2 <i>Improvement of the Transducers to Narrow the Sound Beam</i>	29
2.2.3 <i>Sample Results from the Second Generation</i>	51
2.3 Third Generation of Robotic Scanning System	56
2.3.1 <i>Apparatus</i>	56
2.3.2 <i>Sample Results from the Third Generation — B-scan and 3-D Images for Outdoor Objects</i>	59
2.4 Fourth Generation of Robotic Scanning System	62
Chapter 3. Sonar A-scan Classification Methods for Indoor Objects	64
3.1 A-scan Analysis of Different Materials	64
3.2 Wavelet Analysis for A-scans from Different Objects	69
3.3 Template Matching Analysis	76
3.3.1 <i>Development of Template Matching</i>	76
3.3.2 <i>Template Matching Analysis — Initial Study</i>	81
3.3.3 <i>Template Matching Analysis — Automatic Feature Extraction and classification</i>	86
3.3.4 <i>Template Matching Analysis — Statistical Method</i>	90
3.4 Summary	94
Chapter 4. Analysis of Outside Cylindrical Objects	95
4.1 Trees vs. Poles	95
4.2 Overview of Pattern Recognition	98

	Page
4.3 Methodology	103
4.3.1 <i>Physics</i>	103
4.3.2 <i>Signal Pre-processing</i>	105
4.3.3 <i>Analysis and Results</i>	107
4.4 Conclusion	119
Chapter 5. Analysis of Extended Objects Outside	120
5.1 Object Scanning and Data Analyzing	120
5.1.1 <i>Object Scanning</i>	120
5.1.2 <i>Data Analyzing</i>	123
5.2 Deformable Template Matching	135
5.3 Feature Extraction	142
5.3.1 <i>First Step of Feature Extraction — Deviation vs. N (Template) Plots</i>	142
5.3.2 <i>Second Step of Feature Extraction — Best Fit Template and its Deviation</i>	147
5.4 Objects Categorization	151
5.5 Conclusion	159
Chapter 6. Analysis of Outside Rectangular and Cylindrical Objects	160
6.1 Object Scanning	160
6.2 Data Analysis and Feature Extraction	163
6.3 Further Feature Extraction and Object Classification	176
6.4 Conclusion	179
Chapter 7. Conclusions and Future Work	180

	Page
Appendix A. Air-coupled Ultrasound	187
Appendix B. Matlab Code for Template Matching Analysis	198
Appendix C. Matlab Code for Analysis of Outside Cylindrical Objects	227
Appendix D. Matlab Code for Analysis of Extended Objects Outside	246
Appendix E. Matlab Code for Analysis of Outside Rectangular and Cylindrical Objects	266
Bibliography	275
Vita	285

Acknowledgements

First, I would like to thank Professor Mark Hinders for giving me this golden opportunity to work in his lab here in Williamsburg. It is the most fruitful years I have ever had. My knowledge increased dramatically. From academic aspects to the English language, from the American Culture to the way I see the world, he opened a new world for me, not to mention all the guidance from big pictures to the small notations. Without his efforts and patience, this work would not have been possible.

Second, I would like to thank Professor Gregory Smith, Professor Zia-ur Rahman and Professor Weizhen Mao for reviewing my work and being committee members. I would also like to thank Jonathan Stevens for constructing much of the instrumentation used in this work, including building the robots named William and Mary. I would also like to thank Kevin Rudd for correcting English, helping with building the robots and collecting the data.

Lastly, I would like to thank my family for their love, support and encouragement. Without them, this could not have been done.

List of Tables

Table	Page
3.1 Results from the computer for single echo analysis	89
3.2 Results from the computer for multiple echoes analysis	93
4.1 Surface characteristics + diameter of the two kinds of cylindrical objects— trees and poles	108
5.1 Categorization results of the 15 data sets of the unknown objects	158
6.1 Features for the square trashcan	173
6.2 Features for the round trashcan	173
6.3 Features for the flat brick wall	174
6.4 Features for the convex brick wall	174
6.5 Features for the square lamppost	175
6.6 Features for the round lamppost	175

List of Figures

Figure	Page
1.1 (a) iMow mowing the grass. (from www.amazon.com/exec/obidos/tg/detail/B000066RWA (b) Roomba cleaning the floor (from www.kickassgear.com/Reviews/RumbaRoomba.htm)	4
2.1 Apparatus of the first generation of robotic scanning system	17
2.2 Sub-systems of the first generation of robotic scanning system	17
2.3 LabView interface of the first generation of robotic scanning system	20
2.4 (a) Two bottles scanned by Airmar AT-120 transducer. (b) B-Scan images when they are 4.5 inches apart. (c) B-Scan images when they are 7 inches apart. (d) B-Scan images when they are 10 inches apart	22
2.5 (a) Two bottles scanned by the Airmar AR50-8 narrow beam transducer. (b) B-Scan images when they are 4.5 inches apart. (c) B-Scan images when they are 7 inches apart. (d) B-Scan images when they are 10 inches apart	23
2.6 Apparatus of the second generation of the robotic scanning system	25
2.7 Sub-systems of the second generation of the robotic scanning system	25
2.8 AR50-8 narrow beam transducer and two stepper motors to control rotations about the horizontal and the vertical respectively	28
2.9 LabView Interface of the total scan system for the second generation of robotic system	28
2.10 The amplitude vs. scan step plot of the two angles on the left and their B-scan image on the right. It is acquired by the flat face transducer. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 5 inches	31
2.11 The amplitude vs. scan step plot of the two angles scanned by the flat face transducer RI. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 7 inches	33
2.12 The amplitude vs. scan step plot (left) and the B-scan image (right) of the two angles scanned by the flat face transducer. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 9 inches	33

Figure	Page
2.13 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 2 feet. The transducer R7 was used	34
2.14 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 2 feet. The transducer is R4	35
2.15 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 2 feet. The transducer is R2	36
2.16 B-Scan image of the two angles when they are 5 inches apart. The distance from the sensor to the angle is 2 feet. The transducer we use is R2	37
2.17 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is RI	38
2.18 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is R7	39
2.19 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is R4	41
2.20 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is R2	42
2.21 B-Scan images of the two angles when they are 5 inches apart from each other. The distance from the sensor to the angles is 3 feet 4 inches. The transducers are (a) RI (b) R7 (c) R4 (d) R2	43
2.22 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is RI	44
2.23 The amplitude vs. scan step when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is R7	45

Figure	Page
2.24 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is R4	46
2.25 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is R2	47
2.26 B-Scan images of the two angles when we use (a) RI (b) R7 (c) R4 (d) R2 as the transducer. The two angles are 5 inches apart. The distance from the sensor to the angles is 4 feet 8 inches	49
2.27 Lateral resolution results of the four transducers	50
2.28 (a) Gate built with cinder blocks (b) C-Scan image of the gate. (c) Improved C-Scan image (d) Binary image of the C-Scan image (e) Binary C-Scan image after erosion (f) Binary C-Scan image after further dilation	52
2.29 (a) Two columns of the cinder blocks (b) C-Scan image of the two columns of cinder blocks (c) Improved C-Scan image (d) Binary image of the C-Scan image (e) Binary C-Scan image after erosion (f) Binary C-Scan image after further dilation	53
2.30 (a) Two-block tower (b) C-Scan image of the tower (c) Improved C-Scan image (d) Binary image of the C-Scan image (e) Binary C-Scan image after erosion (f) Binary C-Scan image after further dilation	54
2.31 (a) "H" shape block (b) C-Scan image of the "H" shape block (c) Binary image of the C-Scan image (d) Binary C-Scan image after erosion (e) Binary C-Scan image after further dilation	55
2.32 (a) Two-block tower (b) 3-D image of the two-block tower	57
2.33 The third generation of robotic scanning system packaged as a sensor head for mobile robot —RWilliam	58
2.34 Diagram of the scanner head. The dotted lines represent the two axes that the camera and ultrasound transducer rotate about	58
2.35 (a) Staircase (b) Ivy	60
2.36 (a) 3-D image of the staircase (b) 3-D image of the ivy	60
2.37 (a) B-scan image of the staircase (b) B-Scan image of the ivy	61

Figure	Page
2.38 RMary on her first run down the hallway	63
3.1 (a) Foam block (b) Cardboard (c) Aluminum plate	65
3.2 (a) Echo from the foam block (b) Echo from the Aluminum plate (c) Echo from the cardboard	65
3.3 (a) Echo from the Aluminum plate (b) Echo from the thin cardboard when the plates are 0.7 meter away from the sensor	66
3.4 (a) Echo from the Aluminum plate (b) Echo from the thin cardboard when the plates are 1.0 meter away from the sensor	66
3.5 (a) Aluminum plate (b) A-scan of the Aluminum plate when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan	71
3.6 (a) Thin cardboard (b) A-scan of the thin cardboard when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan	71
3.7 (a) Foam block (b) A-scan of the foam block when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan	71
3.8 (a) Plant (b) A-scan of the plant when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan	72
3.9 (a) Electrical fan with the power off (b) A-scan of the electrical fan with the power off when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan	72
3.10 (a) Electrical fan with the power on (b) A-scan of the electrical fan with the power off when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan	72
3.11 (a) Chair (b) A-scan of the chair when the incident wave is perpendicular to the front surface of its back and the corresponding CWT of the A-Scan	73
3.12 (a) Chair with a bag on it (b) A-scan of the chair when the incident wave is perpendicular to the front surface of the bag and the corresponding CWT of the A-Scan	73

Figure	Page
3.13 (a) Chair with a plane section on it (b) A-scan of the chair when the incident wave is perpendicular to the front surface of the plane section and the corresponding CWT of the A-Scan	74
3.14 (a) Stool (b) A-scan of the chair when the incident wave is perpendicular to the side surface of its legs and the corresponding CWT of the A-Scan	74
3.15 Comparison of the tested echo from (a) Aluminum plate (b) Cardboard (c) Foam block (d) Plant (e) Brick wall (f) Ivy on Brick Wall (g) Block (h) Chair (i) Chair with Bag (j) Thin Cardboard with the ten reference echoes from these ten different objects in the library	83
3.16 The improved phase plane of the comparison of the tested echo from (a) Aluminum plate (b) Cardboard (c) Foam block (d) Plant (e) Brick wall (f) Ivy on Brick Wall (g) Block (h) Chair (i) Chair with Bag (j) Thin Cardboard with the 10 sample echoes from 10 different objects in the library	87
3.17 The statistical research of the phase plane of the comparison of the tested echo from (a) Aluminum plate (b) Cardboard (c) Foam block (d) Plant (e) Brick wall (f) Ivy on Brick Wall (g) Block (h) Chair (i) Chair with Bag (j) Thin Cardboard with the 10 sample echoes from 10 different objects in the library	91
4.1 Mobile robotic platform with computer-controlled scanner	96
4.2 (a) A lamp post. (b) A tree	97
4.3 A typical A-Scan echo	106
4.4 (a) An echo from block (b) Fast Fourier Transform of the echo (c) Absolute value of the echo without the low frequency filtering (d) Absolute value of the echo with the low frequency filtering	106
4.5 Backscatter plots of pole No.1 when the sensor is (a) 100cm (b) 125cm (c) 150cm (d) 175cm (e) 200cm (f) 225cm (g) 250cm (h) 275cm (i) 300cm from the pole	109
4.6 Backscatter plots of tree No.14 when the sensor is (a) 75cm (b) 100cm (c) 125cm (d) 150cm (e) 175cm (f) 200cm (g) 225cm (h) 250cm (i) 275cm from the tree	111
4.7 Backscatter plots of tree No.18 when the sensor is (a) 100cm (b) 120cm (c) 140cm (d) 150cm (e) 160cm (f) 170cm (g) 180cm (h) 190cm (i) 200cm from the tree	112

Figure	Page
4.8 Backscatter plot of Pole No.1 at a distance of 200cm	114
4.9 Backscatter plot of tree No.14 at a distance of 100cm	114
4.10 Asymmetry-Deviation phase plane of the pole group and two tree groups	117
4.11 Asymmetry-Deviation phase plane of pole No.1 and tree No.14	117
4.12 Asymmetry-Deviation phase plane for tree No.14	118
4.13 Average Asymmetry-Average Squared Euclidean Distance phase plane of trees No.14-20 and poles No.1-3, 9-10	118
5.1 (a) Picket fence scanned from inside (b) Metal fence with brick pillar (c) Flat brick wall (d) Hedge (e) Picket fence scanned from outside (f) Brick wall with thin buttress (g) Brick wall with thick buttress	121
5.2 Five scanning locations for one unit of the fence. The two blocks at the ends are posts for the picket fence	124
5.3 The retro-reflection phenomenon happens at the second scan	124
5.4 Backscatter plots of a unit of the picket fence scanned from inside with the robot facing five different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	125
5.5 Backscatter plots of a unit of the metal fence with brick pillar with the robot facing different locations from left to right at a distance of 100cm.(a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	128
5.6 Backscatter plots of a unit of flat brick wall with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	129
5.7 Backscatter plots of a unit of hedge with the robot facing different locations from left to right at a distance of 100cm.(a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	130
5.8 Backscatter plots of a unit of picket fence scanned from outside with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	131

Figure	Page
5.9 Backscatter plots of a unit of brick wall with thin buttress with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	132
5.10 Backscatter plots of a unit of brick wall with thick buttress with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan	133
5.11 Backscatter plot for a picket fence and all its local and major apices, stars '*' mark all the local apices and squares '□' mark all the major apices	136
5.12 Backscatter plots for a picket fence and its n-peak templates (a) Backscatter plot and its one-peak template (b) Backscatter plot and its two-peak template (c) Backscatter plot and its triple-peak template (d) Backscatter plot and its four-peak template (e) Backscatter plot and its five-peak template (f) Backscatter plot and its six-peak template (g) Backscatter plot and its seven-peak template	139
5.13 Backscatter plots of the second scan of a picket fence and its (a) one-peak template (b) two-peak template	144
5.14 Deviation vs. n (template) plot of the second scan for the picket fence scanned from inside	144
5.15 Deviation vs. n (template) plots (a) picket fence scanned from inside (b) Metal fence with brick pillar (c) flat brick wall (d) hedge (e) picket fence Scanned from outside (f) brick wall with thin buttress (g) brick wall with thick buttress on one side and thick buttress with a scatter on the other side. Circles 'o' represent the first scan. Symbols '×' represent the second scan. Plus '+' represent the third scan. Stars '*' represent the fourth scan. Squares '□' represent the fifth scan	145
5.16 Bar chart of the Euclidean distances of the two feature vectors of an unknown object to the objects we already known. Black bar shows the Euclidean distance of their N vector. Gray bar shows the Euclidean distance of their D vector	154
5.17 Bar charts of the Euclidean distances of the two feature vectors of fifteen unknown objects to the seven objects we already known. Black bar shows the Euclidean distance of their N vector. Gray bar shows the Euclidean distance of their D vector	155

Figure	Page
6.1 (a) Square trashcan (b) Round trashcan (c) Flat brick wall (d) Round brick wall (e) Square lamppost (f) Round lamppost	161
6.2 Amplitude vs. scan angle (degrees) plots of a rectangular trashcan when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the trashcan	165
6.3 Amplitude vs. scan angle (degrees) plots of a cylindrical trashcan when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) six feet (f) eight feet (g) nine feet (h) ten feet away from the trashcan	166
6.4 Amplitude vs. scan angle (degrees) plots of a flat brick wall when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) six feet (f) eight feet (g) nine feet (h) ten feet away from the brick wall	167
6.5 Amplitude vs. scan angle (degrees) plots of a convex brick wall when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the brick wall	168
6.6 Amplitude vs. scan angle (degrees) plots of a square lamppost when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the lamppost	169
6.7 Amplitude vs. scan angle (degrees) plots of a round lamppost when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the lamppost	170
6.8 The amplitude vs. scan angle plot of a rectangular trashcan when the robot is three feet away from the trash can. (a) The amplitude vs. scan angle is divided into three parts — the left region, the central region and the right region. (b) <i>Width_1</i> is the width of the central top while <i>Width_2</i> is the width of the central peak at the amplitude value 500	172
6.9 Slope 1 vs. Slope2 phase plane of square trashcan, round trashcan, flat brick wall, convex brick wall, square lamppost and round lamppost. The blue square represents the square trashcan. The blue circle represents the round trashcan. The red square and circle represent the flat and convex brick wall respectively. The black square and circle represent the square and round lamppost respectively	178
A.1 0: transducer; 1: matching layer; 2: air	188
A.2 An ultrasonic beam with the divergence δ	190

Figure	Page
A.3 A curved transducer with the focusing length f equal to the radius of curvature R	190
A.4 Plastic convex lens bonds to a flat face transducer	192
A.5 Relationship between F/l_0 and F_{ac}/l_0	192
A.6 A plastic Axicon lens attached to a flat face transducer	194
A.7 A conical lens attached to the flat face transducer	194
A.8 A right angle conical and two parabolic reflectors to redirect the sound waves	195
A.9 An air-coupled Fresnel zone-plate	195
A.10 Double conical lens	197
A.11 Electric focusing or steering by an phased array	197

Abstract

We have studied the physics of full-field sensor interaction with materials, structures and components in order to develop the brains for useful robots. Four generations of sonar scanning systems have been developed to automatically interpret surrounding environment. The first two are stationary 3D air-coupled ultrasound scanning systems and the last two are packaged as sensor heads for mobile robots. The data acquisition and motion control subsystems have been improved at each stage.

For the indoor objects, three methodologies have been discussed to tell different objects and materials. Among these, template matching is the most effective one. It is conducted by comparing the tested echo with the reference echoes. Important features are then extracted and drawn in the phase plane. The computer then analyzes them and gives the best choices of the tested echoes automatically.

For cylindrical objects outside, an algorithm has been presented to distinguish trees from smooth circular poles based on analysis of backscattered sonar echoes. The echo data are acquired by a mobile robot which has a 3D air-coupled ultrasound scanning system packaged as the sensor head. Four major steps are conducted. First, a series of scans (≥ 9) for one object are done by the mobile sonar system. Second, a backscatter vs. scan angle plot is constructed by a 5th order polynomial fit. Then, Asymmetry and Deviation features are extracted from the interpolation plots. Finally, further feature extractions are done based on the Asymmetry-Deviation graph. Average Asymmetry-Average Squared Euclidean Distance phase plane is segmented to tell a tree from a pole by the location of the data points for the objects interested. Results located in the small area near the origin represent poles while those located away from the origin represent trees.

For extended objects outside, we successfully distinguished seven objects in the campus by taking a sequence scans along each object, obtaining the corresponding backscatter vs. scan angle plots, forming deformable template matching, extracting interesting feature vectors and then categorizing them in a hyper-plane. Results show that this approach to distinguish different objects is promising.

We have also successfully taught the robot to distinguish three pairs of objects outside. Multiple scans are conducted at different distances. Instead of the previous backscatter, amplitude vs. scan angle plots are obtained for analysis. A two-step feature extraction is conducted based on the amplitude vs. scan angle plots. The final Slope1 vs. Slope2 phase plane not only separates a rectangular trash can from a cylindrical trashcan, but also separates a flat brick wall from a convex brick wall, a square lamppost from a round lamppost.

Chapter 1

Introduction

1.1 Ecotobot

Echolocation is a technique that bats and dolphins use to locate obstacles by generating their own sounds and receiving the echoes produced by the obstacles. An ectogenous robot — *Ecotobot* — is a mobile robot that uses echolocation to identify objects, and therefore can operate independently in unstructured environments without a need for a remote control.

We have studied the physics of full-field sensor interaction with materials, structures and components in order to develop the brains for useful robots. Four generations of sonar scanning systems have been developed to automatically interpret the surrounding environment. The first two are stationary 3D air-coupled ultrasound scanning systems and the last two are packaged as sensor heads in mobile robots. The data acquisition and motion control subsystems have been improved at each stage.

1.2 Housework Robots

There are currently a few consumer-market robots for sale nationwide. The *iMow* is a \$500 robotic lawn mower as shown in Figure 1.1(a), made by Friendly Robotics of Israel. Before using *iMow*, it's necessary to run a perimeter wire around the lawn area and connect it to a battery-powered signal generator. After driving *iMow* by remote control

into the lawn area, one merely presses start and *iMow* proceeds autonomously. It first does the perimeter cut by following the wire, and then executes a pair of orthogonal back-and-forth patterns which covers most of the lawn. It has pneumatic bumpers to deal with trees and other obstacles, and goes slow enough to be safe. It has no way to know where it has cut the grass and a human has to judge when is enough and then do the remnants via the attached remote control. Overall it performs rather poorly both in speed and endurance, covering about 5,000 sq.ft. on a single battery which takes 24 hours to recharge. Not surprisingly, the company has claimed bankruptcy.

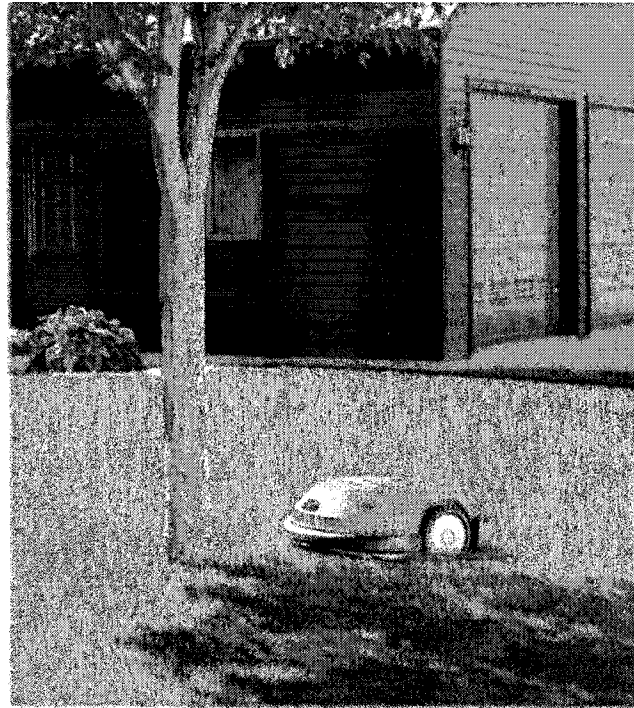
Several retailers such as Target are currently selling a robotic vacuum cleaner, *Roomba* as shown in Figure 1.1(b), made by iRobot of Massachusetts. *Roomba* starts in the center of the room, executing an outward spiral pattern until it encounters a wall or furniture. It has left and right front quadrant bumpers to tell when it hits an object. It also has quite sophisticated algorithms to deal with obstructions and complicated irregular room topologies. It has a downward-looking IR sensor to keep it from tumbling down stairs, as well as a separate IR-projector module that can be used to block off doorways and confine *Roomba* to a particular area. Although primarily a sweeper rather than a vacuum cleaner, it does a good job on an uncluttered 10'x12' office floor.

Now, *Roomba Discovery* is leading the way into the next generation of *Roombas*. With more advancements and improvements, *Roomba Discovery* features dirt detect, a self-charging Home Base(TM) and an Advanced Power System that make this one of the finest robotic floor vacuums on the planet. However, *Roomba Discovery SE* (special

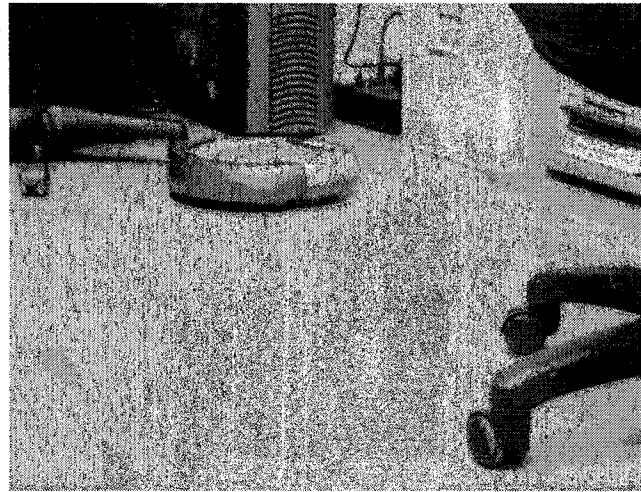
edition) is the top of the line Roomba with some very down to earth features. Like Roomba Discovery, the SE features Dirt Detect, the self-charging Home Base(TM), an extra-large debris bin, the Roomba Remote, three cleaning modes plus a Charging Wall Mount for convenient storage. And, of course, the SE comes with the Advanced Power System that gives all Roombas the energy to get the job done.

The short battery life and slow recharging time are Roombas' biggest limitations. However, the recharging time has been reduced to three hours recently. *Roomba* does do an admirable job of getting itself unstuck when trapped under a chair or wedged under a piece of furniture. On October 25th, 2004, iRobot announced its millionth *Roomba* sold. Although it's a simple matter to press start and let it go, *Roomba* is too loud to have running while carrying on a phone conversation or watching television. In addition, *Roomba* will not work on thick carpet. One advantage over traditional vacuums is that it easily cleans things that otherwise never get swept or vacuumed.

The main deficiency of *Roomba* and *iMow*, aside from their short battery life, is that they have no situational awareness; they are effectively blind so they have to bump into something to tell that it's there. Figure 1.1 shows *iMow* mowing the grass and *Roomba* cleaning the floor. A major goal of this research is to give robots the ability to detect their surrounding environment.



(a)



(b)

Figure 1.1 (a) iMow mowing the grass. (from www.amazon.com/exec/obidos/tg/detail/B000066RWA)
(b) Roomba cleaning the floor (from www.kickassgear.com/Reviews/RumbaRoomba.htm).

1.3 DARPA Grand Challenge

On March 13, 2004, The Defense Advanced Research Projects Agency (DARPA) raised a unique land race in Barstow, CA – Grand Challenge, a contest run by vying to be the first autonomous vehicle to reach Las Vegas and win \$1 million for its builders. The purpose was to boost the development of autonomous vehicles that could find uses in the American military. Among the original 86 applicants, 15 teams passed the previous qualifying testing and attend the race.

Unfortunately, none of the 15 driverless robots made more than about 5 percent of the way through the course. Within 4.5 hours, all vehicles had been withdrawn or were disabled and were being returned to the starting line.

Among the fifteen teams, five did not make it beyond at the beginning. The other ten actually had make their unmanned vehicle run for a while, and then all their vehicles ended up hitting something unexpected. For example, Sandstorm, which made the longest run — 7.4 miles, finally got caught on a berm. SciAutonics, which made the second longest run — 6.7 miles, went into an embankment and became stuck.

The sensor package was the weakest link for these unmanned vehicles. None of the vehicles did a good job of noticing the obstacles before they ran into them. However, the ability of noticing and recognizing obstacles is a pre-requisite for unmanned vehicles to avoid collision. That is also what we try to accomplish in the dissertation.

1.4 Overview of Biosonar Research

1.4.1 How Bats and Dolphins Detect Objects

As early as 1773, it was observed that bats could fly freely in a dark room and that hearing was an important component of bats' orientation and obstacle avoidance capabilities [1]. By 1912, it was suggested that bats use sounds inaudible to humans to detect objects [2], but it wasn't until 1938 that Griffin proved that bats use ultrasound to detect objects [3]. Griffin and his colleagues did a wide variety of experiments over several decades to understand the orientation ability of many different kinds of bats. They thoroughly investigated various properties of bats' ultrasonic pulses, such as duration, frequency, intensity and so on. Overcoming formidable experimental difficulties, the frequency modulation characteristics of bats' pulses were found. Their spectrograms were presented for the first time, showing that during each tone-burst pulse the frequency drops an octave in pitch. When bats are approaching their prey, the duration of their chirp becomes shorter and hence the slope of the spectrogram becomes larger.

Echolocation is a technique bats use to generate their own sounds and locate obstacles by means of their own ears. Griffin also investigated echolocation by blind people and did the definitive experiments to prove the ability of echolocation. As a result, many artificial devices for human echolocation have been built during the last several decades such as KSONAR made by Bay Advanced Technologies Ltd in New Zealand and Ultracane made by Sound Foresight Ltd in UK. They are both sonic navigation devices for the blind. There are also some simpler devices such as the metal heel plate worn by some blind men in order to make their footsteps generate clicking sounds. Like bats,

dolphins also have the ability to echolocate [1]. Unlike bats, which send out chirps, dolphins often operate in a pulse mode, sending out a click and then receiving the target echo before sending out another click.

Many other scientists have also done extensive research on the echolocation by bats. James A. Simmons [4]-[6] classified bats according to the types of ultrasound they send out. Bats that emit constant-frequency (*CF*) components exploit echo Doppler shifts for measuring the velocity of targets. These bats preferentially attack targets that return echoes containing rapid modulations of frequency and amplitude originating from an insect's beating wings. The auditory systems of '*CF* specialists' are adapted for processing the narrow band of frequencies contained in Doppler-shifted and rapidly modulated *CF* echoes. The other category of bats emits frequency-modulated (*FM*) echolocation signals. They will attack flying insects as well as a variety of other airborne objects that they encounter. However, with practice, these bats reject inedible objects while accepting edible objects most of the time, indicating a high degree of discrimination using sonar.

Simmons and his co-workers compared echo data from a disk, a mealworm and different sizes of spheres. For example, one important difference between mealworms and spheres is that spheres reflect the same echo from all directions while mealworms reflect different echoes from different directions. *FFT* and cross-correlation for echoes of each type of target suggest that it would be misleading to say that the bat perceives the

shape of targets on terms of echo spectra. Rather, it's more likely that bats perceive the shape along the spatial dimension of range with the help of spectral cues.

During detection, the bat moves its head from left to right to scan the targets. From one trial to the next, the bat occupies a slightly different position to observe the targets. An ideal sonar receiver is one that replaces the echo with its corresponding cross-correlation function and then estimates the target's location in range from the cross-correlation function's location in time. The envelope of the cross-correlation predicts the accuracy of the entire discrimination. The time-difference between echoes arriving at the two ears provides an important clue for the bat to determine the horizontal direction of targets.

Broadband signals carry more information with which to construct an acoustic image of a target. The increasing bandwidth of the signals provides greater accuracy on perceiving target features and locating the targets' position. Long *CF* or narrow-band signals contribute little to a high-resolution acoustic image. However, their long duration enables precise specification of their frequency, providing good resolution of target velocities from echo Doppler shifts. The high duty cycle inherent in the use of long-*CF* transmissions provides an opportunity to monitor fluttering motion of targets from small amplitude or frequency modulations in echoes.

Rudolf Kober and his co-workers [7] have found that the bats classify insects by their different wing beat rates. Sonagrams and oscillograms are recorded and compared. Both

FFT and *AR*-model (autoregressive spectral analysis) are used for analysis. The frequency resolution of the *AR*-model is better than that of *FFT* because it is not coupled to the length of the time segment.

1.4.2 Ultrasonic Systems That Mimic Bats

In the robotics community Roman Kuc and his students [8]-[16] have developed a series of active wide-beam sonar systems that mimic the sensor configuration of echolocating bats, and can distinguish planes, corners and edges necessary to navigate indoors. Kuc's other system —a sonar-driven robot, *ROBAT*, can track an object moving in three dimensions using qualitative interpretation of the sonar signals. *ROBAT* is equipped with a sonar system consisting of five identical transducers configured in the form of a cross. The center transducer emits an acoustic pulse and then pairs of receivers that flank the transmitter horizontally and vertically receive the echoes. Kuc has also modeled the bat's echolocation of prey and investigated the basic elements in sonar using first-order models. Two transmitters and two receivers are necessary and sufficient to distinguish planes, corners and edges. He modeled an acoustic emission and calculated the echoes. His results matched well to experimental data. Kuc has also developed an adaptive sonar system that is mounted on a robot arm to detect and identify objects using echolocation. The sonar system consists of a center transmitter flanked by two receive transducers. The transmitter points at the object to maximize the incident acoustic intensity and the receivers rotate to maximize the echo amplitude and bandwidth. The system can recognize a collection of ball bearings, machine washers and rubber *O*-rings of different sizes. A database is first established in the learning phase by having the

system detect and scan each object in their collection. The recognition task is accomplished by comparing observed data with the entries in the database using a least square criterion. By averaging one hundred waveforms, objects differing by less than one millimeter can be reliably differentiated.

L.G. Ullate, M.T.Sanchez and E. Villanueva [17] have modeled a B-mode ultrasonic imaging system using three transducers mounted in a static triangular configuration. The system determines the position of single or multiple reflecting points placed in the inner part of a triangle. Signals received by the three transducers are displayed as well as the impulse response caused by three reflectors. Obtaining the impulse response by deconvolution from the echoes significantly increases the system resolution, which can be applied to accurately determine the object position.

Ken Sasaki and Masaharu Takano [18] classified objects' surfaces by calculating the acoustic transfer function between an echo from a reference plane and that from an object being tested. A single condenser type transducer with a *26mm* diameter and *80kHz* resonant frequency was used as both transmitter and receiver. A linear frequency modulated *FM* signal from *120kHz* down to *30kHz* was used as a transmission signal. Echoes from objects both in time-domain and frequency-domain were analyzed. Transfer functions of a time delay, a step and an edge were presented both in time and frequency domain in order to classify different objects' surface.

J.R Llata, E.G.Sarabia and J.P.Oria [19] have also built a three-dimensional artificial vision system for robotic applications using an ultrasonic sensor array, which is made up of one emitter and eight receivers around it.

1.4.3 Robotic Ultrasound Systems

Much interesting work has also been done by Billur Barshan and her colleagues [20-39]. Barshan [20] described a novel, flexible and three-dimensional multi-sensor sonar system to localize the center of a generalized spherical target and estimate its radius. Point, line, and planar targets were included as limiting cases which are important for the characterization of a mobile robot's environment. She also presented a novel method for surface profile extraction based on morphological processing of multiple range sensor data [21]. Ayruolu and Barshan addressed the use of evidential reasoning and majority voting in multi-sensor decision making for target differentiation using sonar sensors [22]. They also compared the performances of different methods for the classification and localization of commonly encountered features in indoor environments with sonar [23].

Lindsay Kleeman and his colleague have also done a large amount of work in sonar research[40-52]. Heale and Kleeman [40] described a new sonar system that can perform target localization in two dimensions and classify structures into planes, concave corners and convex edges. Kleeman [41] presented a new sonar target tracking system that is capable of accurately tracking targets at measurements exceeding 10Hz. Two sonar trackers, each consisting of a transmitter and two receivers, are independently controlled to track sonar targets from bearing and range measurements. Chong and Kleeman [42]

described a mobile robot equipped with a sonar sensor array in a guided-feature based map-building task in an indoor environment. The landmarks common to indoor environments are planes, corners and edges, and these are located and classified with the sonar sensor array. In a recent paper [43], Kleeman described two advanced sonar systems that are used to simultaneously localize and map an indoor environment using a mobile robot. This paper presents the approach and results from on-the-fly map building using a Kalman filter and a new odometry error model that incorporates variations in effective wheel separation and angle measurements.

1.4.4 More Sophisticated Detection and Map Building Techniques

Other research groups are also developing algorithms for use of sensor data. Leonard and Durrant-Whyte [53] have provided a detailed sonar sensor model and an algorithm for model-based localization. They also presented a unified approach to navigation, in which the multiple requirements of localization, obstacle avoidance and map building can be simultaneously addressed in a common multi-target-tracking framework. Dior and his co-workers [54] have trained a neural network to recognize two-dimensional shapes independent of orientation, based on echoes of ultrasonic pulses similar to those used by an echolocating bat. Three types of input representations were used: time domain (waveform and cross-correlation) frequency domain (power spectrum) and time-frequency (spectrogram). The network using the spectrogram representation recognized 90% of the echoes from many orientations. It is well suited to applications such as robotics and autonomous vehicles. Krishna and Kalra [55] have presented an algorithm that detects, tracks and avoids multiple moving objects during real time navigation of a

mobile robot. The main contribution of their algorithm is its ability to classify a given sequence of range data to classify a static or a dynamic object based on the two classification strategies provided.

McKerrow and his colleagues have also done some interesting biosonar research. McKerrow and Harper [56] described a plant-recognition system using Continuous Transmission Frequency Modulated (CTFM) ultrasonic sensors. They have also modeled plant echoes from a CTFM ultrasonic sensor using an acoustic density profile model [57]. Then, they identified a set of features that represent plant geometric characteristics and used these to perform an inverse transform from echo features to plant geometry. This plant-recognition technique was then applied to mobile robot navigation [58]. Ratner and McKerrow [59] reported on research into navigation along concrete paths through a park using the edge of the path as a continuous landmark. The robot measures its location to the path edge with a CTFM ultrasonic sensor and steers to follow a trajectory relative to the edge. When it loses the edge, it commences a recovery strategy.

Crowley [60] described a system for autonomous navigation by an intelligent mobile robot in a known domain. It is based on maintaining a description of the external environment of the robot using a focused rotating ultrasonic ranging device. The system is designed to provide autonomous navigation by an intelligent mobile robot.

Carelli and Freire [61] proposed a mobile robot control law for corridor navigation and wall following based on sonar and odometric sensor information. A state variable estimation structure is proposed that fuses the sonar and odometric information.

1.4.5 Image Fusing Technologies

Several research groups are attempting to fuse ultrasound with other imaging sensors. Catherine Wykes and her colleagues [62] have built a prototype integrated ultrasonic/vision sensor that uses an off-the-shelf *CCD* camera and an ultrasonic phased array sensor. This system enables the camera to be calibrated using data from the ultrasonic sensor. It overcomes limitations of the individual sensors and provides *3D* information about the targets. The ultrasonic system is made up of a combined single transmitter with a four-element receiving array. The system uses a capacitive transducer, which has a resonant frequency of *100kHz* with a *3dB* bandwidth of *30kHz*. The transmitter is driven with five square pulses at the resonant frequency and produces a burst that consists of five cycles at *100kHz* with a well-defined peak in its envelope. A.M. Flynn and his colleagues [63] have combined sonar and infrared sensors together for mobile robot navigation. The sonar range finder measures the distance to an object, but has poor angular resolution due to its wide beam width. In contrast, the infrared sensor has good angular resolution in detecting the absence or presence of an object but can not measure distance accurately. This combination system can perceive its environment with better accuracy than either sensor used alone.

1.5 Summary

In this chapter, from the Ectobot to the housework robots and then the unmanned vehicles in the DARPA Grand Challenge, the goal is to develop sensor packages for

mobile robots so that they can automatically interpret the environment. We want to build algorithms to teach the robots to recognize objects automatically so that they can have the ability to deal with real world instructions such as “Go down to the trash can and throw away the trash.”

Of all the previous biosonar research, most are limited in the indoor objects of simple shapes. For those who did do the research for outside objects, both the types of objects and the classification algorithms are quite limited. In this dissertation, we will do an extensive research on a variety of outside objects and build algorithms to teach the robots to recognize objects based on the physics of sonar backscatter from the objects.

Chapter 2

Four Generations of Robotic Systems

Four generations of robotic scanning system have been built in our lab. The first two are stationary 3D air-coupled ultrasound scanning systems and the last two are packaged as sensor heads for mobile robots — rWilliam and rMary. A series of non-contact air-coupled transducers made by Airmar, Inc. are used in the systems.

2.1 First Generation of Robotic Scanning System

2.1.1 Apparatus

The first generation of robotic scanning system consists of three sub-systems —data acquisition, motion control and image analysis. Figure 2.1 displays the apparatus and Figure 2.2 describes its corresponding sub-systems.

In the data acquisition system, the Airmar pulser-receiver is connected to a transducer, a power supply and a LeCroy 9310M oscilloscope. Signals are detected by the transducer and sent back to the pulser-receiver to be amplified. Then the signal is digitalized by the LeCroy 9310M digital oscilloscope and acquired by the GPIB data acquisition card on a PC.



Figure 2.1 Apparatus of the first generation of robotic scanning system.

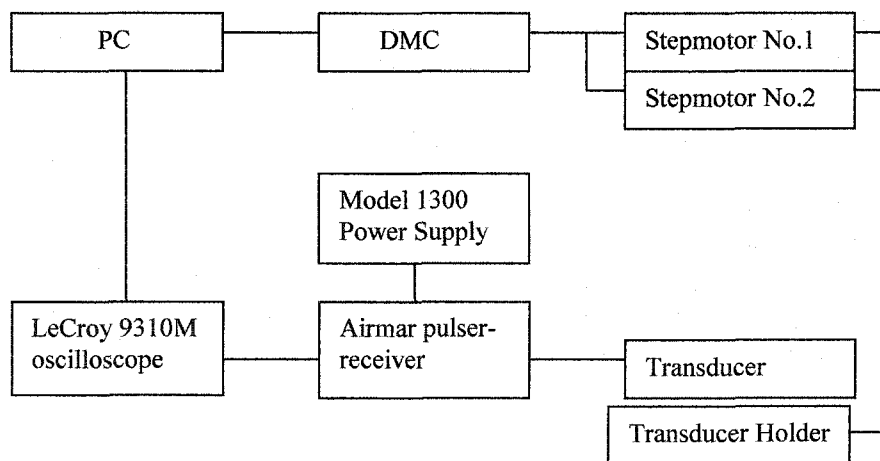


Figure 2.2 Sub-systems of the first generation of robotic scanning system.

Both the air transducer and the development kit are made by AIRMAR Technology Corporation. There are several transducers for use in the first generation such as Airmar AT120 transducer (operation frequency: 120kHz +/-4%, sensor range: 15cm ~ 7m, and the beam width: 12°+/-2°), AT200 transducer (operation frequency : 200kHz +/-4%, sensor range: 10cm ~ 3m, and the beam width: 14°+/-2°) and AR50-8 narrow beam transducer (operation frequency: 50kHz +/-4%, sensor range: 30cm ~ 15m and the beam width: 8°+/-1°). Jumpers can be set to allow different operation frequencies: 25kHz, 40kHz, 50kHz, 120kHz, 200kHz and 225kHz. Both the power and frequency can be slightly adjusted. The echo signals can be amplified by a factor of 1, 10, 100 or 1000.

The motion control system consists of two motors that are connected to a DMC motion controller. One motor controls movement in θ direction and the other controls movement in Z direction.

The major part of the motion control system is the DMC-1000 Series state-of-the-art motion controller plugs into the PC Bus. Operation of the DMC-1000 requires a PC with the appropriate DMC-1000 and LabView drivers. The motion control system is controlled by a double –loop LabView program. The inside loop controls steps in θ direction while the outside loop controls steps in Z direction.

Scan1.vi is a LabView program that combines the data acquisition and motion control. It moves the transducer, acquires the waveform and records the data automatically. Figure 2.3 shows the *LabVIEW* interface of *scan1.vi*. 'Trig Level',

'Volts/div', 'Time/div' are settings to control the signal waveform. *'Xbegin', 'Xend', 'Xsteps' and 'Ybegin', 'Yend', 'Ysteps'* are settings to control the motor. *'A Scan Length', 'X Step pos' and 'Y Step pos'* are indicators monitoring the current position and waveform length during scanning. These settings are saved in data file as well. A Gateway Pentium Two is used as the control PC.

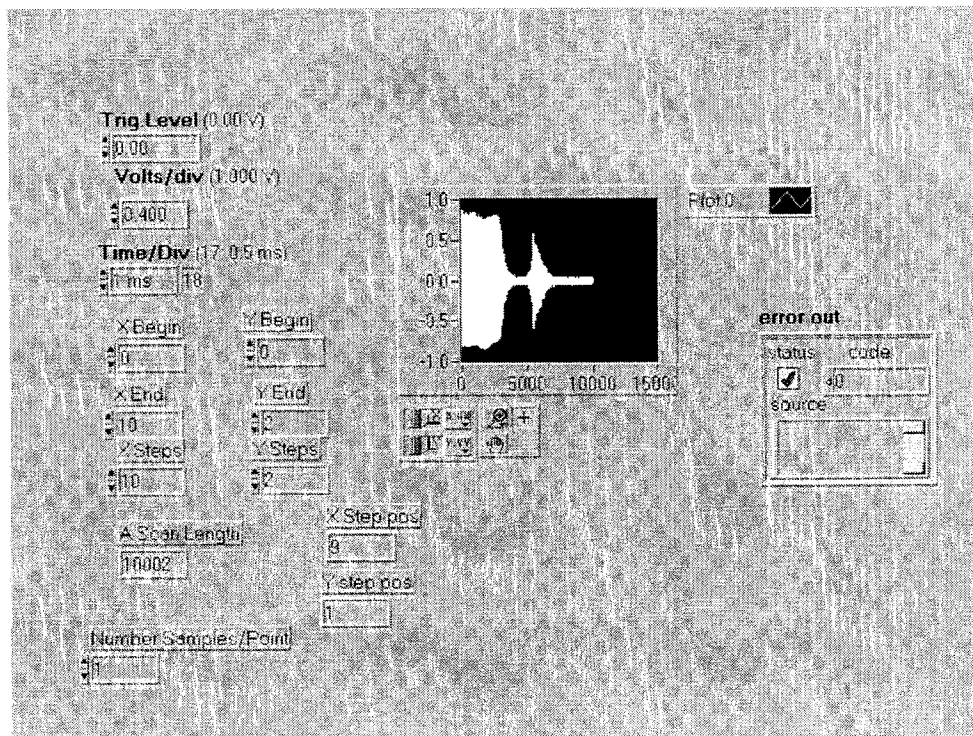


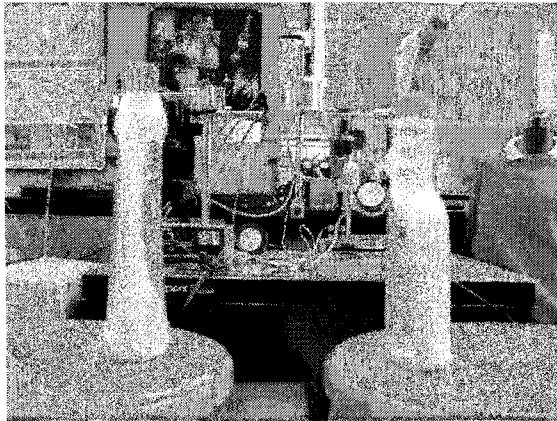
Figure 2.3 LabView interface of the first generation of robotic scanning system.

2.1.2 Sample Results from the first generation —B-Scan Images

A B-Scan image is the 2D image of the plane parallel to the wave propagation. It is built by first stacking many A-scans together and then converting the amplitudes of echoes to grayscale pixels to form an image. Figures 2.4-2.5 show the picture of two bottles and their corresponding B-Scan images when they are 4.5 inches, 7 inches and 10 inches apart. B-scan images in Figure 2.4 are acquired with an Airmar AT120 transducer. We can tell the two bottles apart when they are 10 inches away from each other in the B-Scan images. B-scan images in Figure 2.5 are acquired with an Airmar AR50-8 narrow beam transducer. We can tell the two bottles apart even when they are 4.5” away from each other. The AR50-8 narrow beam transducer gives better spatial resolution than the AT120 transducer.

2.1.3 Limitations of the first generation apparatus

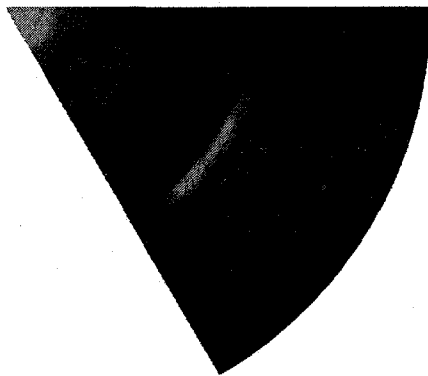
The work we did on the first generation of the robotic system allows it to move the transducer, scan the object and acquire the data automatically for the first time. However, there are several limitations of this apparatus. In the data-acquisition system, the data transfer rates run only a few hundred KB/s for GPIB, which indicates a low transfer rates. In the motion control system, the devices are big although the scan volume is quite limited within a maximum 30cm range in the vertical direction. Further work can't be efficiently done due to these limitations. It is necessary to upgrade the apparatus to the next level.



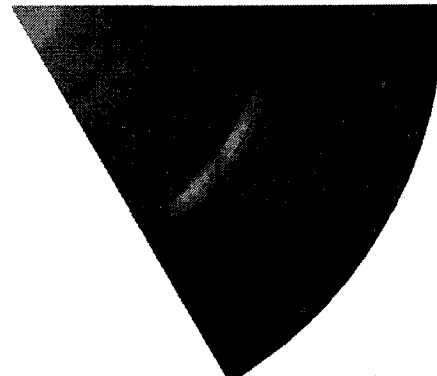
(a)



(b)



(c)



(d)

Figure 2.4 (a) Two bottles scanned by Airmar AT-120 transducer. (b) B-Scan images when they are 4.5 inches apart. (c) B-Scan images when they are 7 inches apart. (d) B-Scan images when they are 10 inches apart.

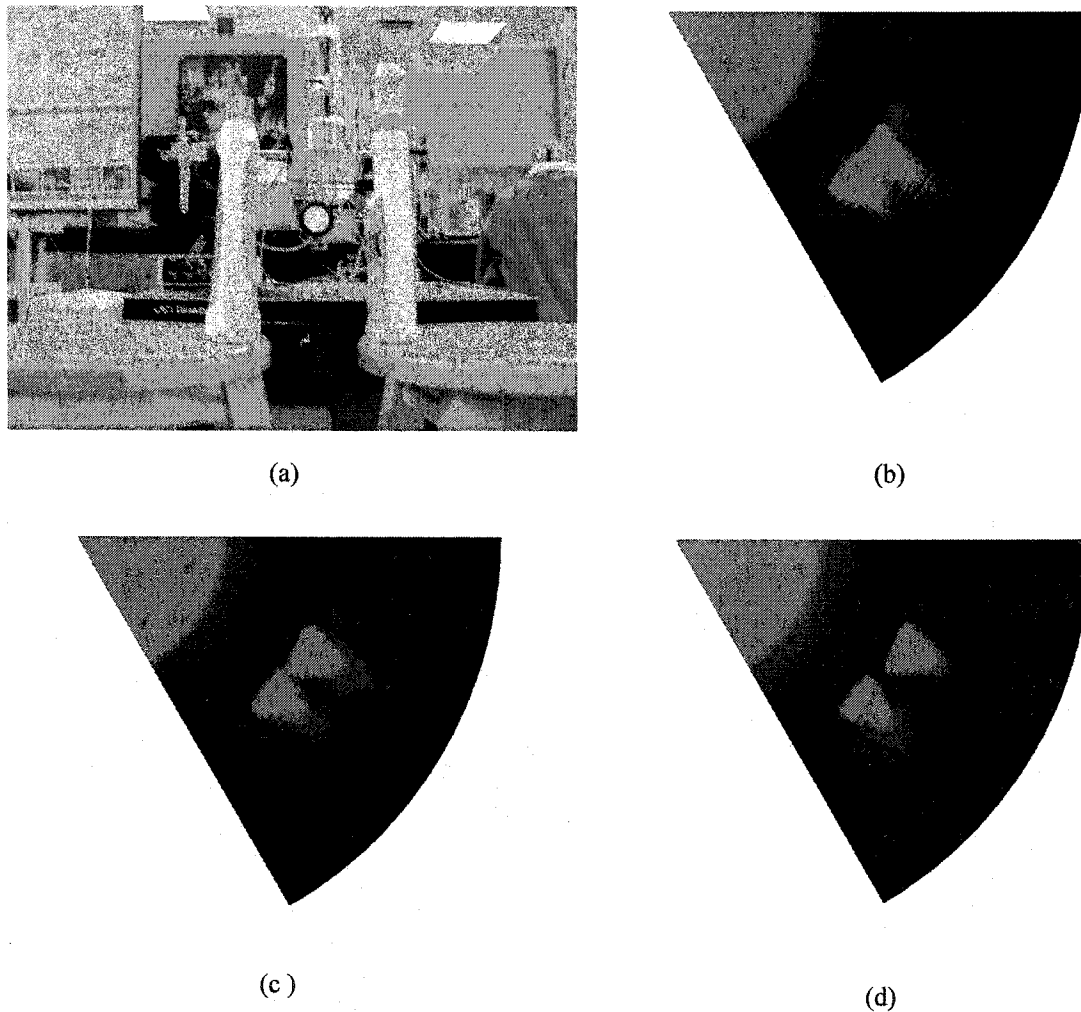


Figure 2.5 (a) Two bottles scanned by the Airmar AR50-8 narrow beam transducer. (b) B-Scan images when they are 4.5 inches apart. (c) B-Scan images when they are 7 inches apart. (d) B-Scan images when they are 10 inches apart.

2.2 Second Generation of Robotic Scanning System

2.2.1 Apparatus

Based on the first generation, a second generation of robotic scanning system has been developed. Both data acquisition and motion control sub-systems have been improved. Figure 2.6 displays the apparatus and Figure 2.7 describes the corresponding sub-systems.

In the data-acquisition system, the GPIB data acquisition card and the oscilloscope have been replaced by a high quality data acquisition board — Gage CompuScope 12100 data acquisition card. The Airmar pulser-receiver is connected to a MODEL SF-9582 AC/DC Power Supply, an AR 50KHz narrow beam transducer and a Gage CS12100 data acquisition card. Signals are detected by the transducer and sent back to the pulser-receiver to be amplified. They are then digitalized by Gage CS12100 data acquisition card and displayed on the computer.

The sensor is now an Airmar AR50-8 narrow beam transducer with an operation frequency of 50 kHz +/-4%. The sensor range is 30cm ~ 15m and the beam width is 8° +/- 1° . The pulser-receiver is still the 3"×4" air transducer development kit except that now it is set to operate at 50 kHz.



Figure 2.6 Apparatus of the second generation of the robotic scanning system.

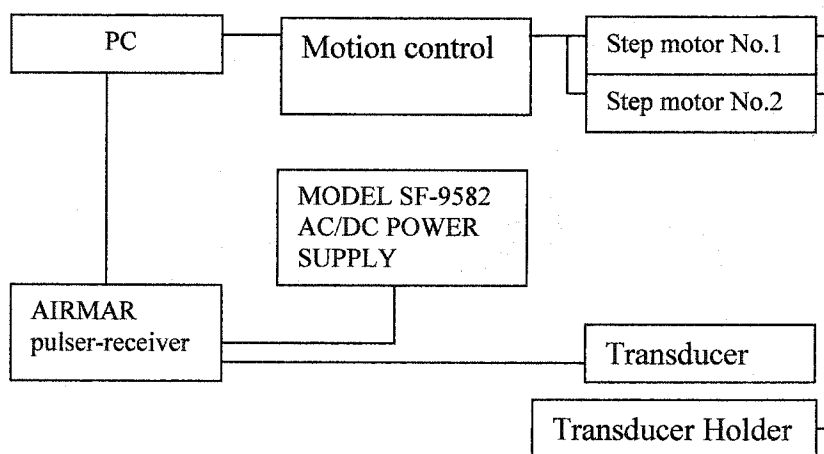


Figure 2.7 Sub-systems of the second generation of the robotic scanning system.

CompuScope cards are much better than other DSO (Digital Storage Oscilloscope) under GPIB control. The CS12100 features 12 bit vertical resolution as opposed to the 8 bit resolution offered by typical DSOs. Data transfer rates from CompuScope memory to PC memory run as high as 100MB/s as compared to a few hundred KB/s for GPIB. Also, CompuScope cards are easier to program because Software Development Kits are available for DOS, Windows 95/98, Windows NT, LabVIEW and MATLAB. Another advantage of CompuScope cards is they are installed inside the PC. Therefore, there is no external box such as a DSO. CompuScope cards are also less expensive than a combination of a DSO and GPIB controller, cabling and software. In addition, the CS12100 has much larger memory buffers than a DSO/GPIB combination. From the CompuScope Software development Kit for LabVIEW, the *Fast16BitOscope.vi* program is chosen to run the data acquisition system.

The motion control system has been made by our senior lab technician Johnthan Stevens. Its two motors are Eastern Air Devices Inc ZB 17EBK-12 stepper motors. One controls rotation about the horizontal axis while the other controls rotation about the vertical axis. A custom motor controller board is connected to the computer via serial interface. The size of the whole control system is 8"×8"×2", which is reduced dramatically from the first generation system. Also the scan volume is greatly increased because the sensor rotates in both horizontal and vertical directions. The apparatus is shown in Figure 2.8.

A LabVIEW driver was created to control the motion system via the PC serial port. It includes an initialization of the serial interface and a double loop of motion control. Similar to the first generation, the inside loop controls rotation in the horizontal axis while the outside loop controls rotation in vertical axis. Delays are added between each step to prevent malfunctions. The amount of delay time is critical to the performance of the system. When the step size increases, the delay should be increased accordingly.

The total system is run by the LabView program *FixGWNewBScan.vi*. Its interface is shown in Figure 2.9. It moves the transducer, acquires the waveform and records the data automatically. For the data-acquisition settings, 'single channel' is chosen for the Capture mode. The sample rate is 5MHz. The trigger source is 'Channel A'. The trigger slope is 'positive'. 'Xstepsize', 'Xsteps', 'Ystepsize' and 'Ysteps' are settings to control the motors. 'Xcount' and 'Ycount' are indicators monitoring current position in real time. The three windows show the waveforms of signals in real time. A Dell RAID computer is used as the control PC. While scanning, the camera atop captures the images of objects in real time.

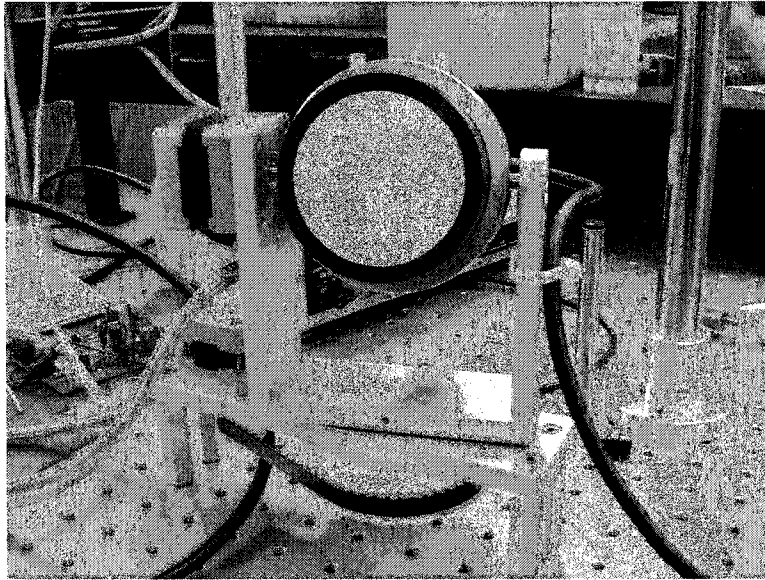


Figure 2.8 AR50-8 narrow beam transducer and two stepper motors to control rotations about the horizontal and the vertical respectively.

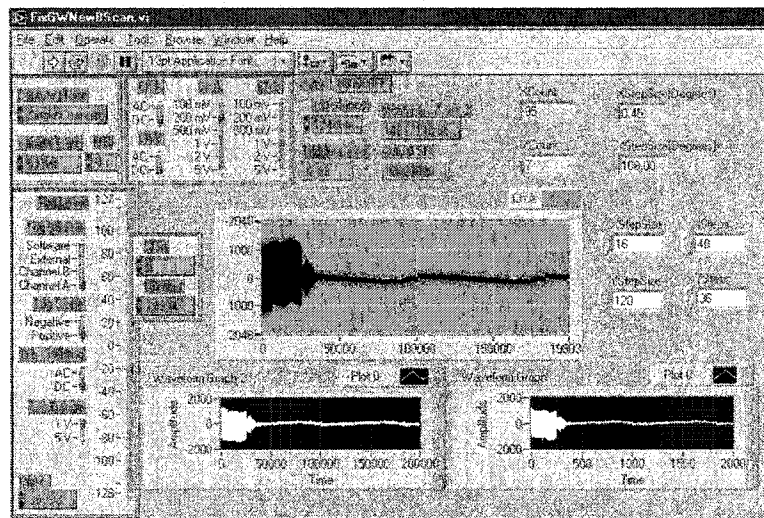


Figure 2.9 LabView Interface of the total scan system for the second generation of robotic system.

2.2.2 Improvement of the Transducers to Narrow the Sound Beam

In Figure 2.4, bottles shown in their B-scan images demonstrate the poor spatial resolution, which is due to the large beam size of the Airmar AT120 transducer. When the Airmar AT 120 transducer is replaced by the Airmar AR50-8 narrow beam transducer, the spatial resolution improves (see Figure 2.5).

However, we still need higher resolution than the Airmar AR50-8 narrow beam transducer provides. In order to narrow the sound beam to achieve higher resolution, we modified the Airmar AR50-8 transducer based on the lens research in Appendix A. In that appendix, it is shown that either the curved concave front face of a transducer or the concave lens bonded to a flat face transducer can focus the sound beam (see Figure A.7-8). Therefore, curving the front face of a transducer might also narrow the sound beam and improve the resolution. However, since the front face matching layer is no longer $\lambda/4$ after the curving, it will cause some mismatching and lead to weaker echoes.

In the second generation system, in order to increase the lateral resolution, we modified Airmar AR50-8 50kHz narrow beam transducers. A concavity was machined into the matching layer to achieve beam half-widths of less than 8° . Therefore, the matching layer also served as a focusing lens. Four AR50-8 transducers with different matching layer curvatures were created and tested. For convenience, we call them R1, R2, R4, and R7. R1 is the original transducer without any change, i.e. a radius of curvature of infinity. R2 has a matching layer with a radius of curvature 2.5 inches. R4 has a matching layer with a radius of curvature 4 inches. R7 has a matching layer with a radius of curvature 7.3 inches. Two parallel steel angles are used for the testing as shown in Figure

2.6. A total of 36 B-scan images of the two angles have been obtained by changing their distance to the sensor (2 feet, 3 feet 4 inches, 4 feet 8 inches) and the distance between them (5 inches, 7 inches, 9 inches). RI, R2, R4 and R7 are tested, respectively, in order to find which transducer gives the best resolution and at the same time preserves the signal energy of echoes.

A 96 scan step test is conducted while the sound beam sweeps across the two angles. The reason we use the two angles to test the transducers is to avoid the effect of different angles of incidence. The two side surfaces make a right angle for each angle. This makes the angle to be a retro-reflector. When the sound wave interacts with a retro-reflector, their original direction is changed by 180 degrees and they return back to the transducer. Therefore, the transducer can receive strong echoes from a retro-reflector regardless of the angle of incidence. Then, the amplitude of the echo at a certain distance in the original A-scan waveform is extracted at each step and shown in an *amplitude vs. scan step* plot, which shows how strong the back scatter echo is at each step.

The left-hand side of Figure 2.10 shows the *amplitude vs. scan step* plot of the two angles [114] scanned by the flat face transducer RI. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 5 inches. Although not very clear, two wide peaks representing echoes from the two angles still show up in the amplitude vs. scan step plot. However, the B-scan image of the two angles on the right hand side doesn't tell them apart very well.

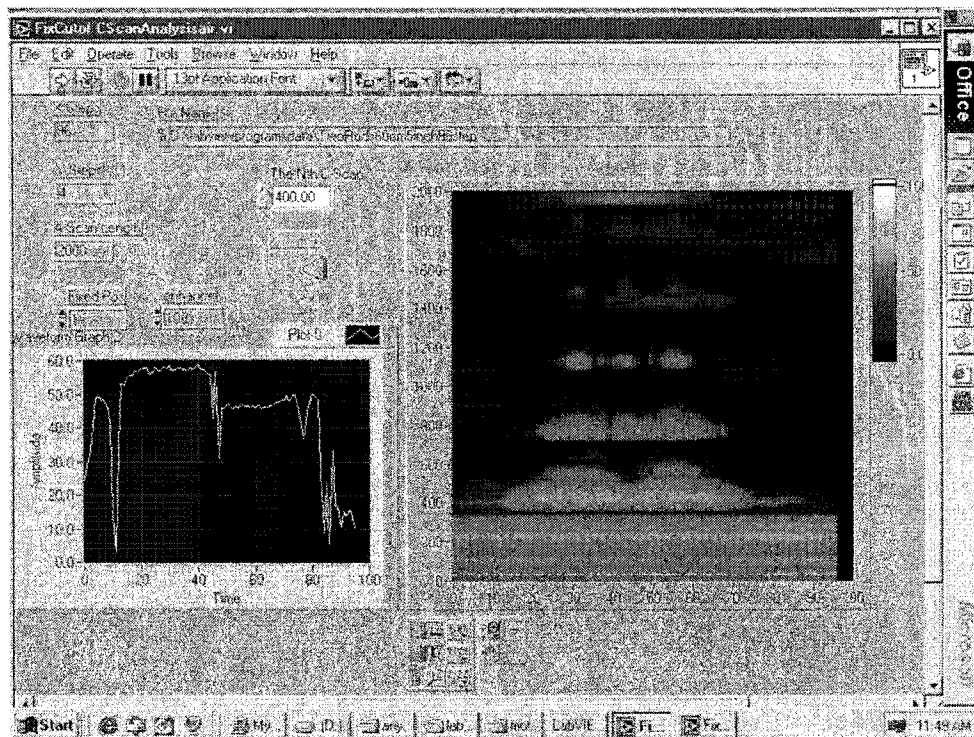


Figure 2.10 The amplitude vs. scan step plot of the two angles on the left and their B-scan image on the right. It is acquired by the flat face transducer. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 5 inches.

We then increased the distance of the two angles to 7 inches and obtained the corresponding *amplitude vs. scan step plot* in Figure 2.11. The two peaks representing the two angles show up clearly. Figure 2.12 displays the results when the two angles are 9 inches away from each other. Both the *amplitude vs. scan angle plot* and the B-Scan image show the two angles clearly.

Figure 2.13 shows the *amplitude vs. scan step plot* for transducer R7 for each angle configuration. The two peaks representing the two angles still show up, although there are some artifacts. Next, transducer R4 was used for testing. Figure 2.14 displays the *amplitude vs. scan step plot* for each angle configuration. They still indicate the two angles while the artifacts increase. Figure 2.15 displays *amplitude vs. scan step plots* when the transducer R2 is used for each angle configuration. Figure 2.16 shows the corresponding B-scan image when the two angles are 5 inches apart. Two major peaks show up clearly in each case. Therefore, when the two angles are 2 feet away from the sensor, R2 gives the best resolution.

The distance from the sensor to the angles was changed to be 3 feet 4 inches. The flat face transducer RI is tested first. The distances between the two angles are 5 inches, 7 inches and 9 inches respectively. Figure 2.17 shows the *amplitude vs. scan step plot* for

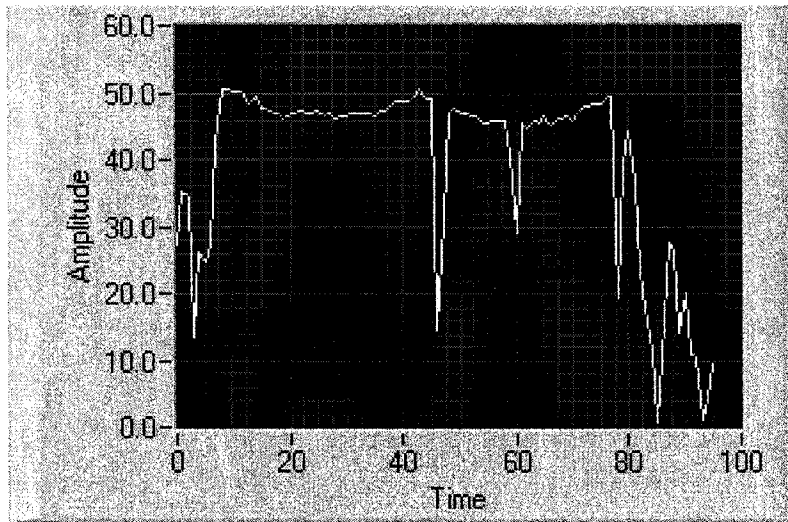


Figure 2.11 The amplitude vs. scan step plot of the two angles scanned with the flat face transducer RI. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 7 inches.

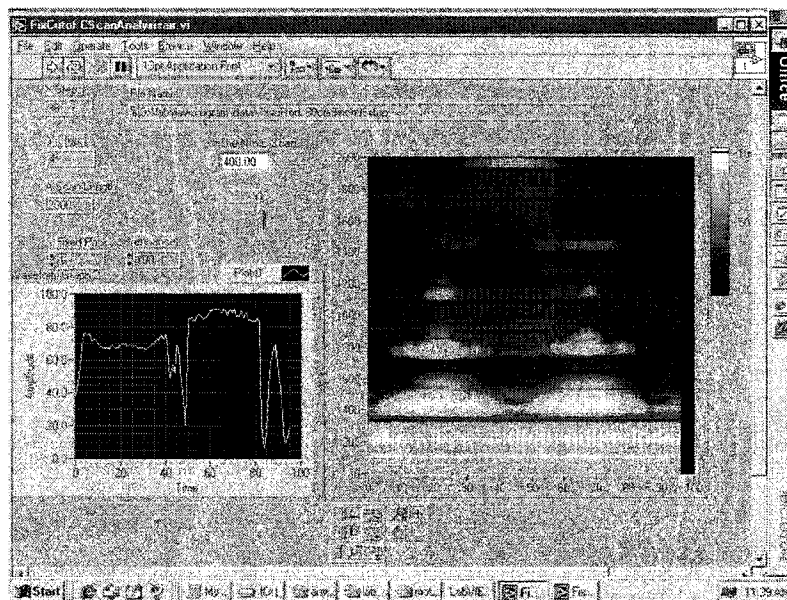
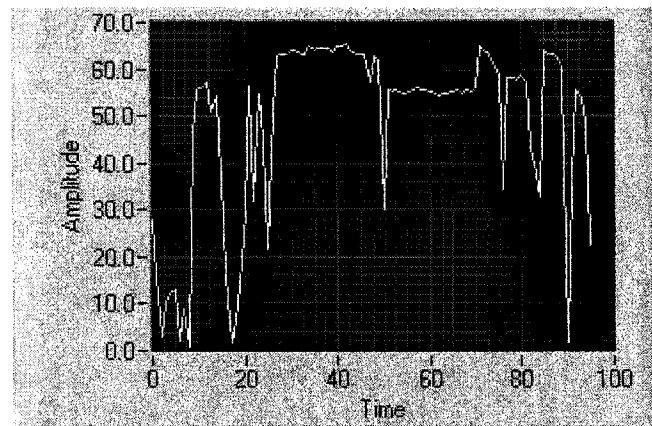
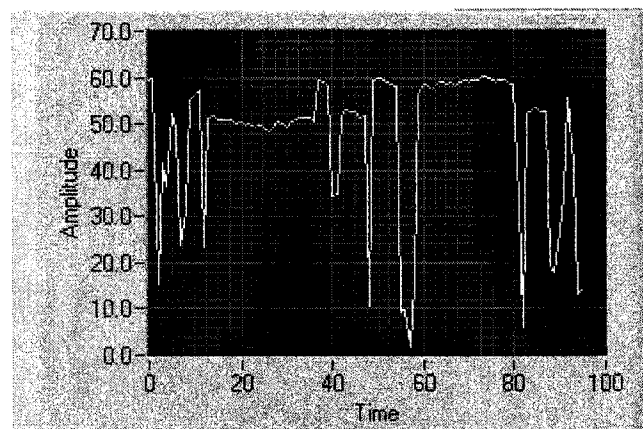


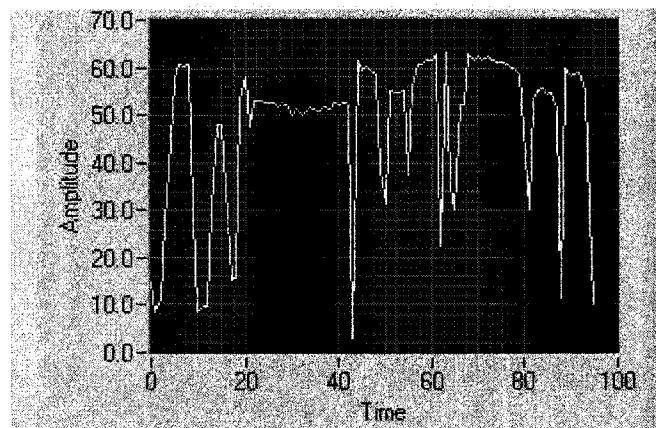
Figure 2.12 The amplitude vs. scan step plot (left) and the B-scan image (right) of the two angles scanned with the flat face transducer. The distance from the angles to the sensors is 2 feet. The distance between the two angles is 9 inches.



(a)

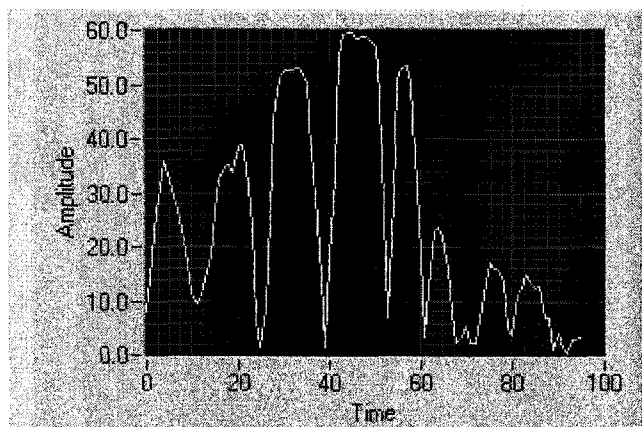


(b)

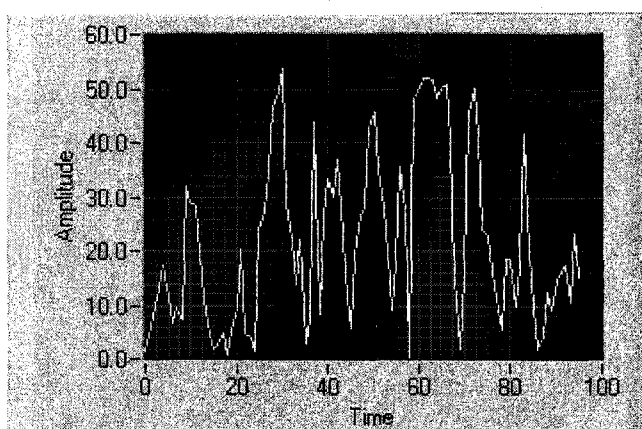


(c)

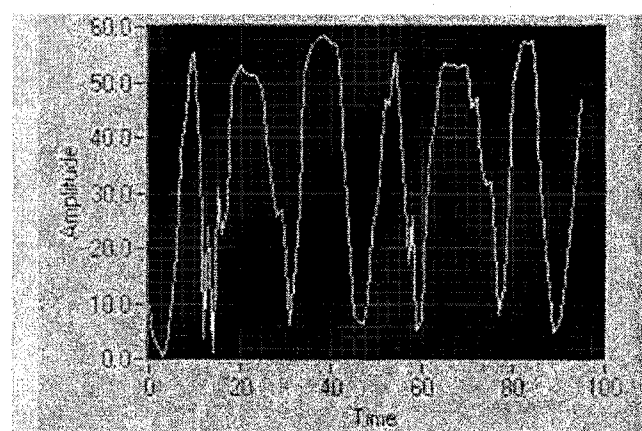
Figure 2.13 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 2 feet. The transducer R7 was used.



(a)

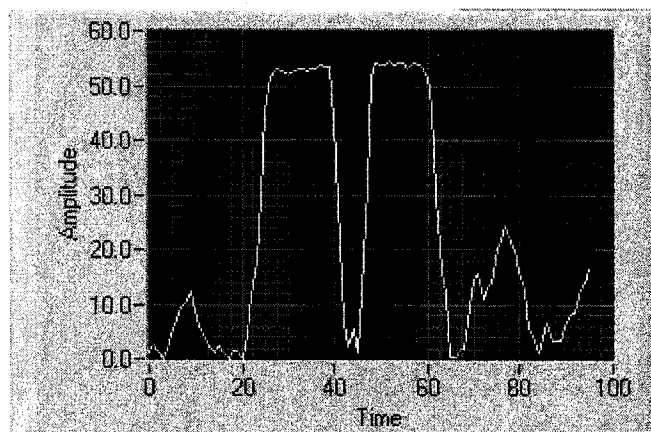


(b)

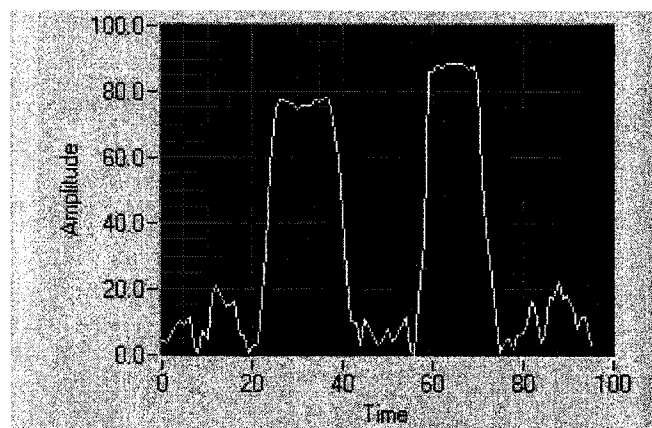


(c)

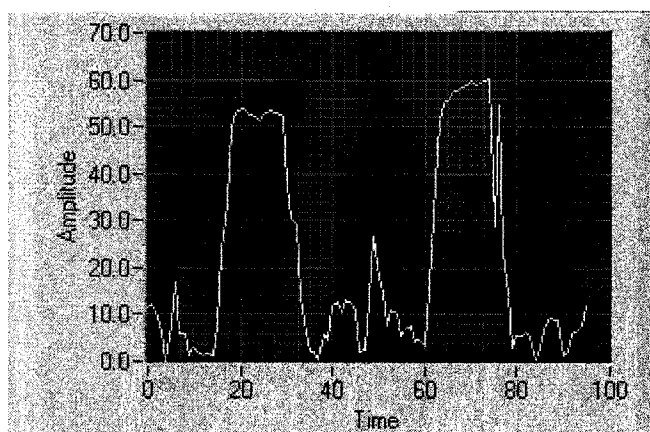
Figure 2.14 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 2 feet. The transducer is R4.



(a)



(b)



(c)

Figure 2.15 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 2 feet. The transducer is R2.

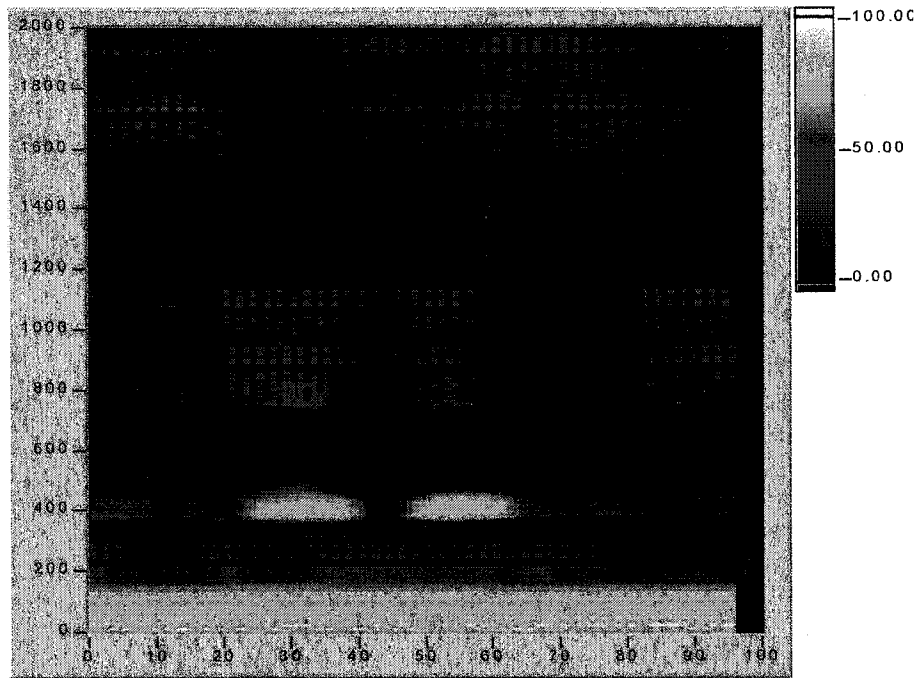
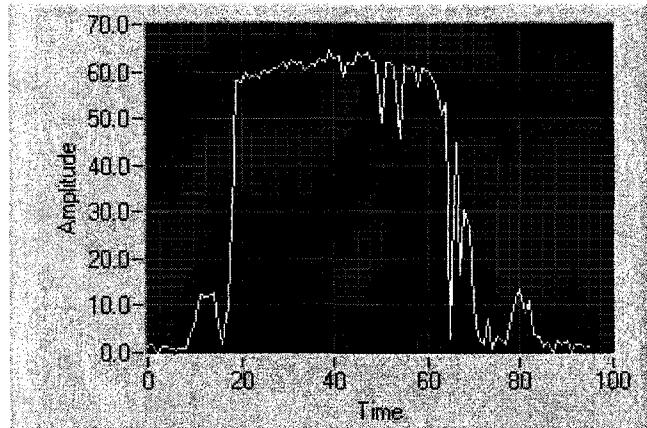
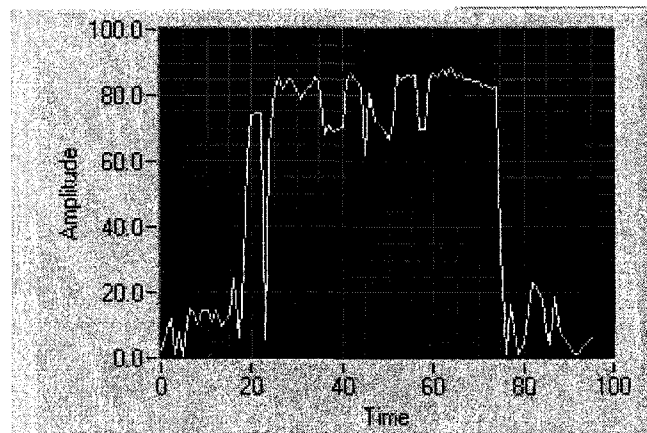


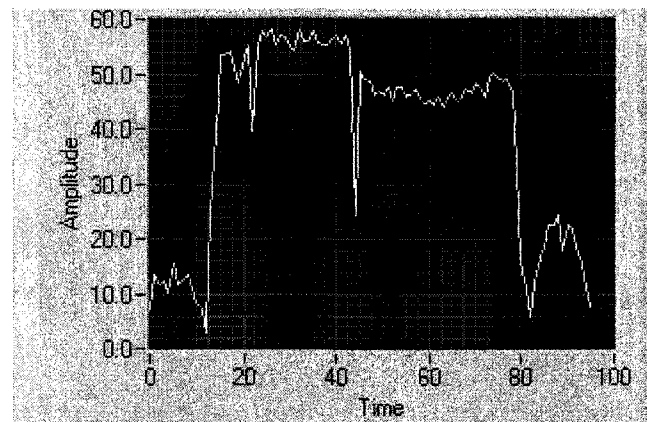
Figure 2.16 B-Scan image of the two angles when they are 5 inches apart. The distance from the sensor to the angle is 2 feet. The transducer we use is R2.



(a)

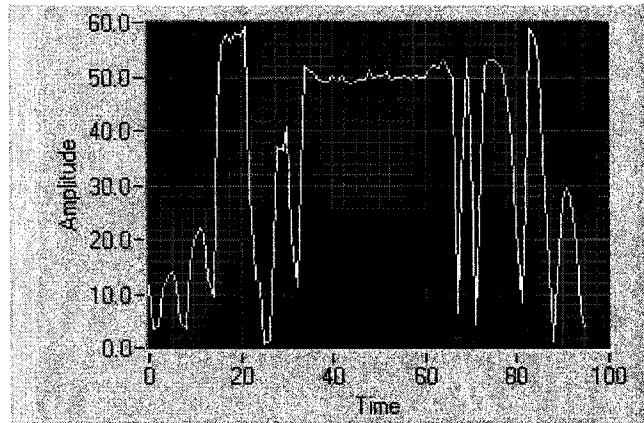


(b)

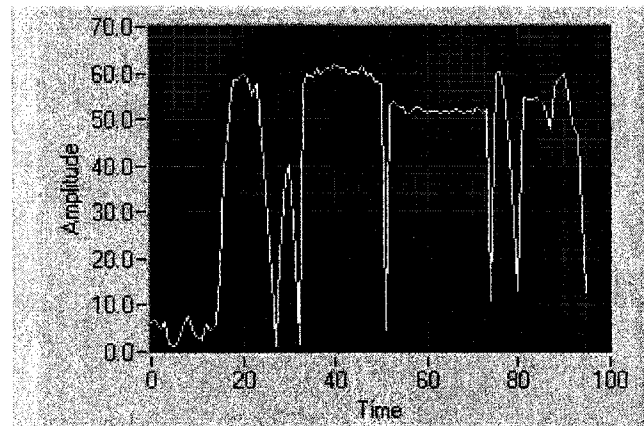


(c)

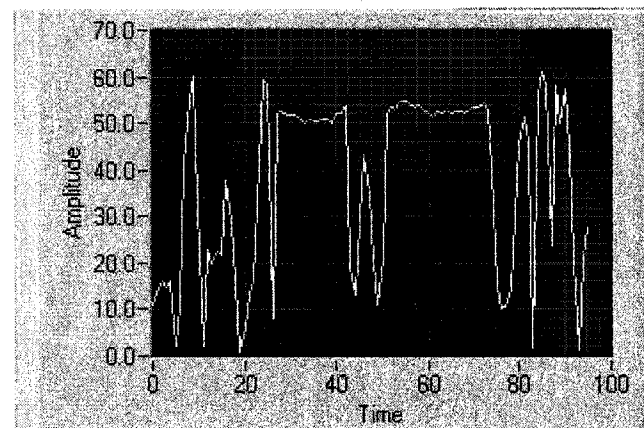
Figure 2.17 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is RI.



(a)



(b)



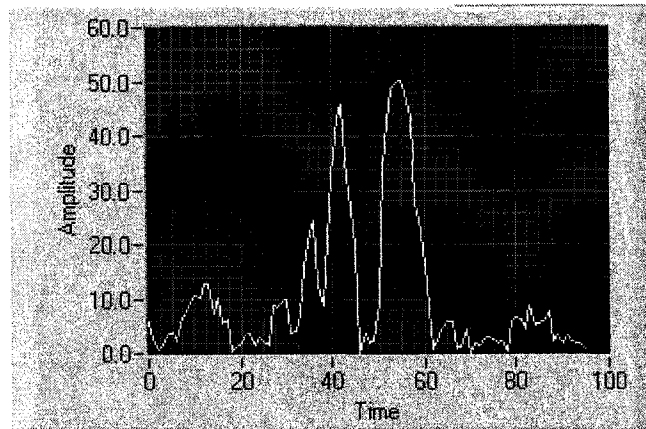
(c)

Figure 2.18 The amplitude vs. scan step plots when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is R7.

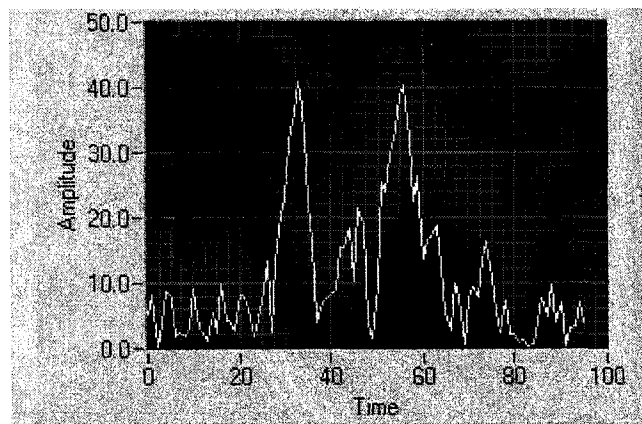
each case. The two peaks representing the angles overlap with each other in the first two cases. However, the peaks are distinct in the images when the angles are 9 inches apart. Figure 2.18 shows the *amplitude vs. scan step* plots when transducer R7 for each case.

The two angles are not resolvable when they are 5 inches apart. However, at the distances of 7 inches and 9 inches apart, two major peaks show up despite some artifacts. Figure 2.19 shows the *amplitude vs. scan step* plots for R4. The widths of the main peaks are narrower than those acquired from RI and R7. It indicates good resolution. Figure 2.20 shows the *amplitude vs. scan step* plots with R2 for each case. The resolution is as good as R4. Figure 2.21 displays the B-scan images when the two angles are 3 feet 4 inches away from the sensor and 5 inches apart from each other. R2, R4 give better resolution than RI, R7. Among these transducers, R2 still gives the best resolution at 3 feet 4 inches.

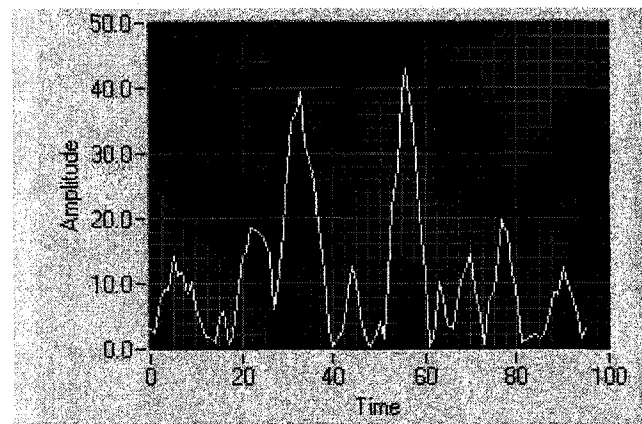
The distance from the sensor to the angles is now increased to 4 feet 8 inches. The distances between the two angles are still 5 inches, 7 inches and 9 inches. Figure 2.22 shows the *amplitude vs. scan step* plots for RI. In none of the plots can we tell the two angles apart. Figure 2.23 displays the *amplitude vs. scan step* plots for R7 in each case. The peaks representing the two angles merge together in the first two cases. However, the angles are resolvable when they are 9 inches apart. Figure 2.24 presents the *amplitude vs. scan step* for R4. Two major peaks show up very well. Figure 2.25 is the *amplitude vs. scan step* plots for R2. The peaks representing the two angles overlap each other when the two angles are 5 inches apart. Nevertheless, they are resolvable when they are 7 inches and 9 inches apart.



(a)

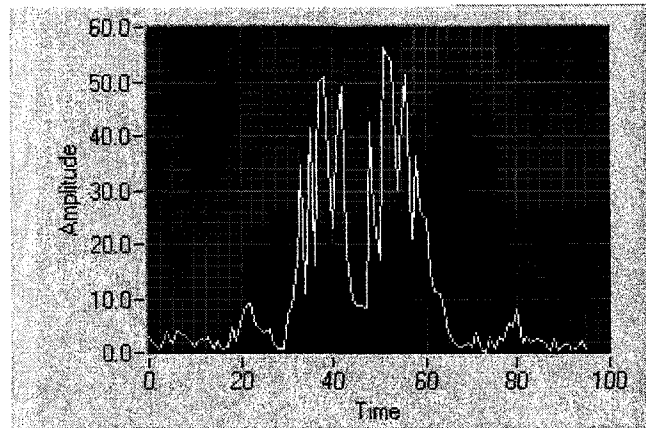


(b)

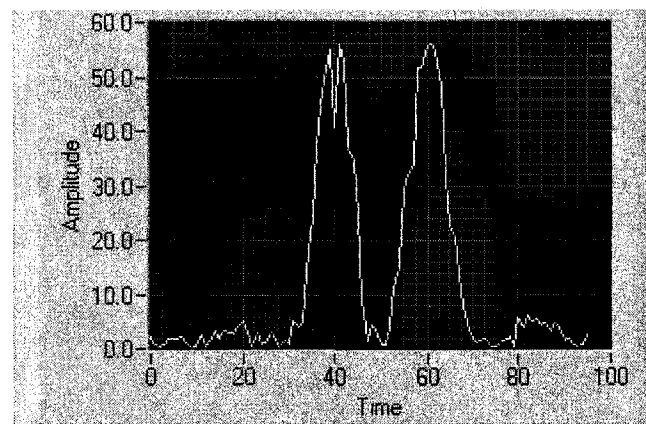


(c)

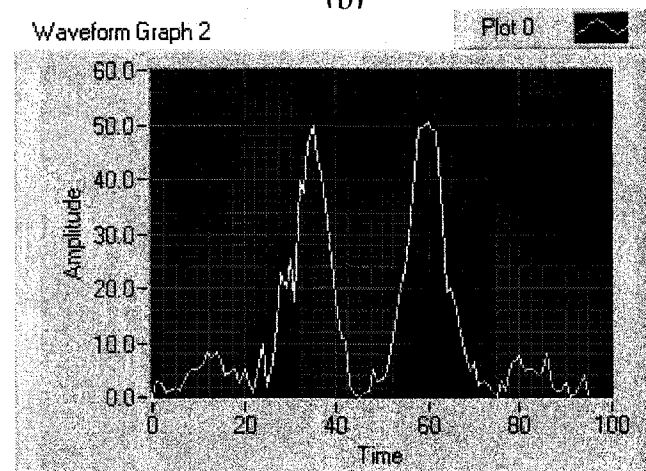
Figure 2.19 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is R4.



(a)

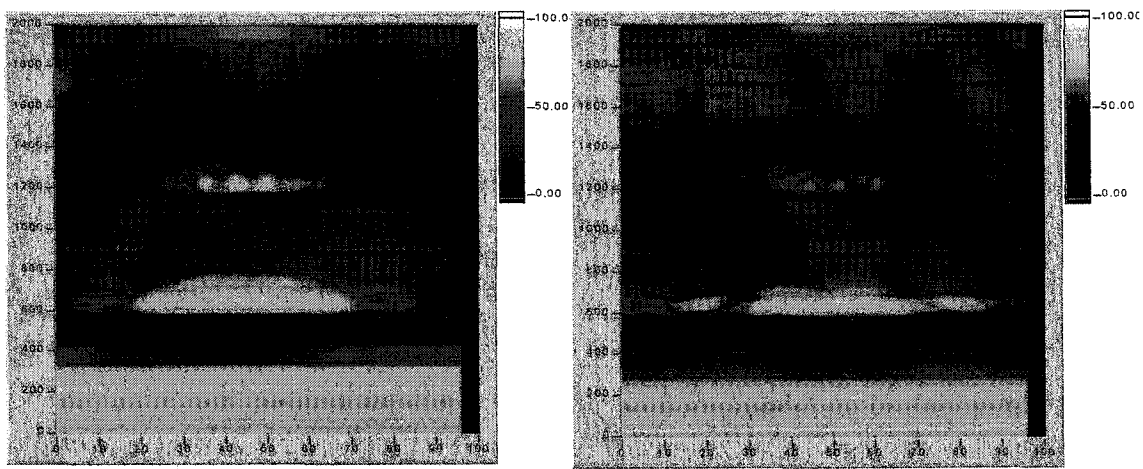


(b)



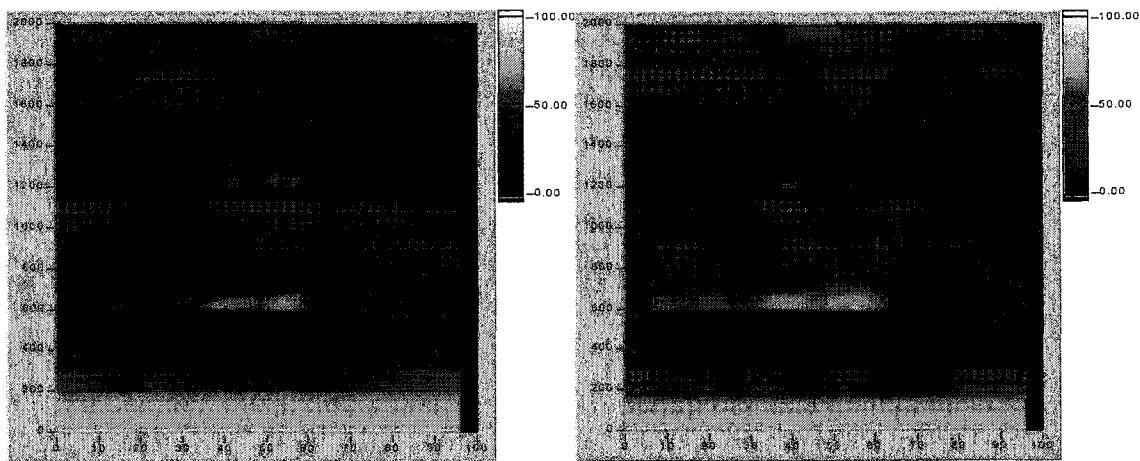
(c)

Figure 2.20 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 3 feet 4 inches. The transducer is R2.



(a)

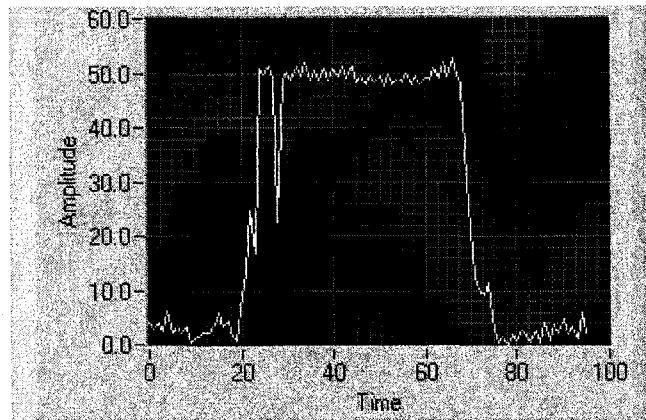
(b)



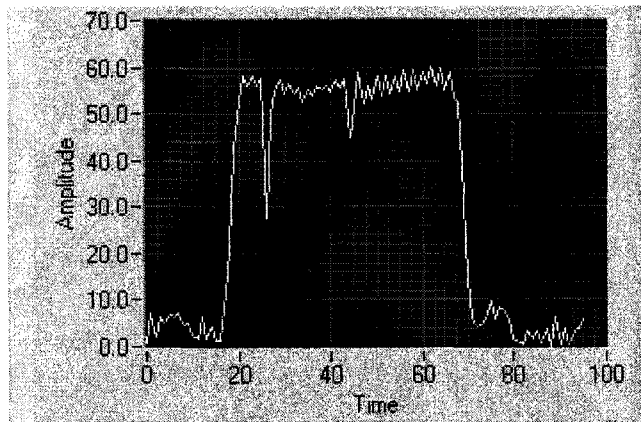
(c)

(d)

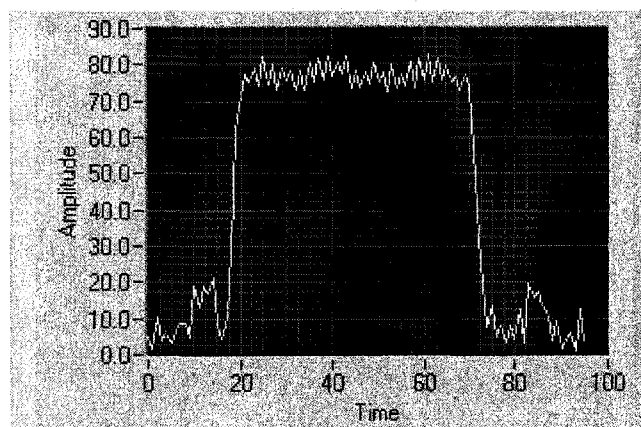
Figure 2.21 B-Scan images of the two angles when they are 5 inches apart from each other. The distance from the sensor to the angles is 3 feet 4 inches. The transducers are (a) RI (b) R7 (c) R4 (d) R2.



(a)

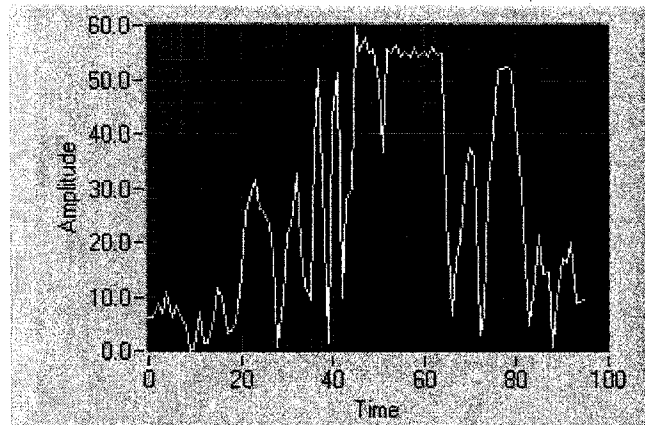


(b)

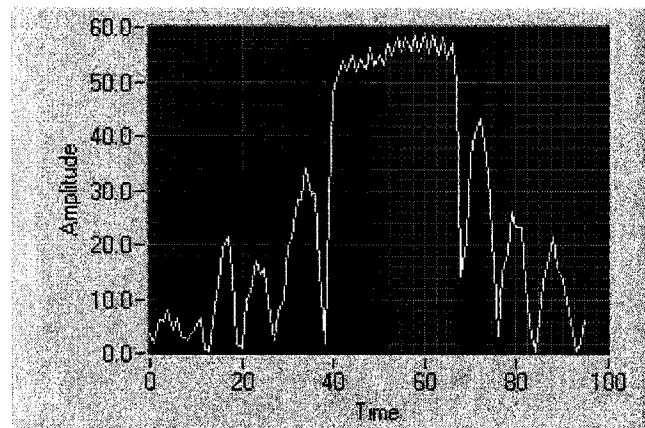


(c)

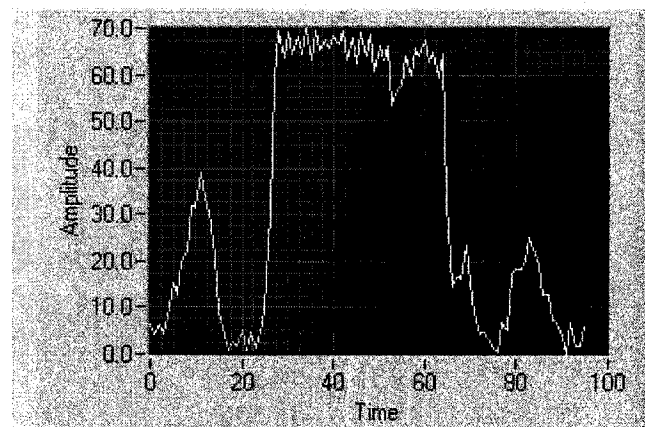
Figure 2.22 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is RI.



(a)

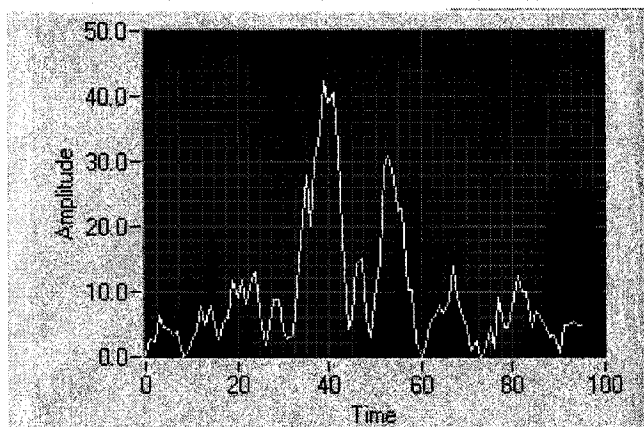


(b)

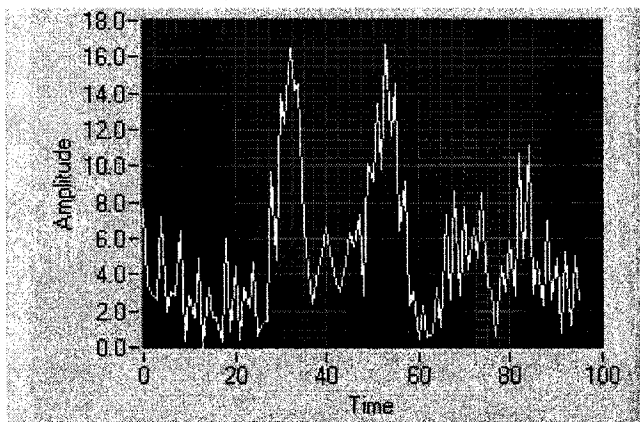


(c)

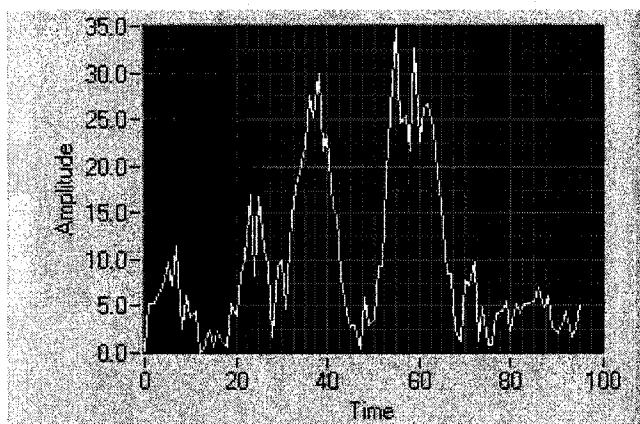
Figure 2.23 The amplitude vs. scan step when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is R7.



(a)

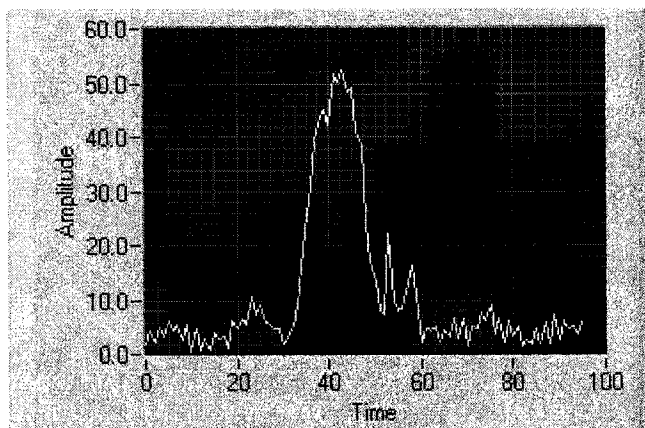


(b)

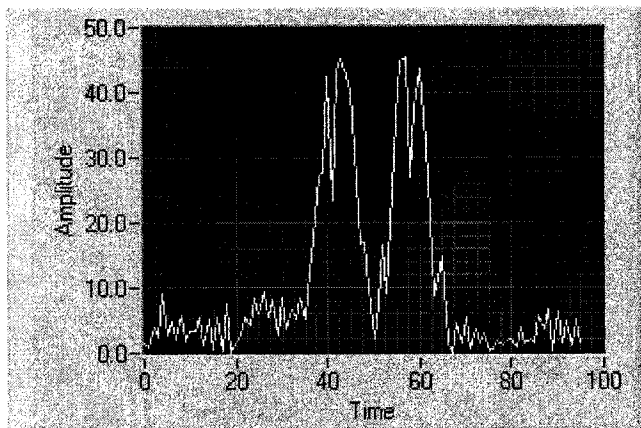


(c)

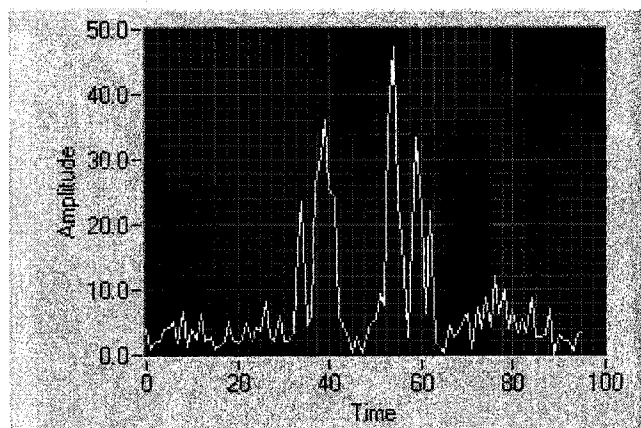
Figure 2.24 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is R4.



(a)



(b)



(c)

Figure 2.25 The amplitude vs. scan step plot when the distance between the two angles are (a) 5 inches (b) 7 inches (c) 9 inches. The distance from the two angles to the sensor is 4 feet 8 inches. The transducer is R2.

Figure 2.26 shows the B-Scan images of the two angles when the distance from the sensor to the angles is 4 feet 8 inches and the two angles are 5 inches apart. It demonstrates that R2 and R4 give better resolution than R1 and R7. Among these four transducers, R4 gives the best resolution. R2 has a shorter focus length than R4, which makes the sound beam of R2 diverge at longer distances.

Based on the experiments above, the lateral resolution of the four different transducers is described in Figure 2.27. Although R2 and R4 have better lateral resolution than R7, their high impedance mismatch causes high reflectance, which presents energy from going through the air. Therefore, according to its high resolution and low reflectance, R7 is chosen as the sensor head of the robotic system.

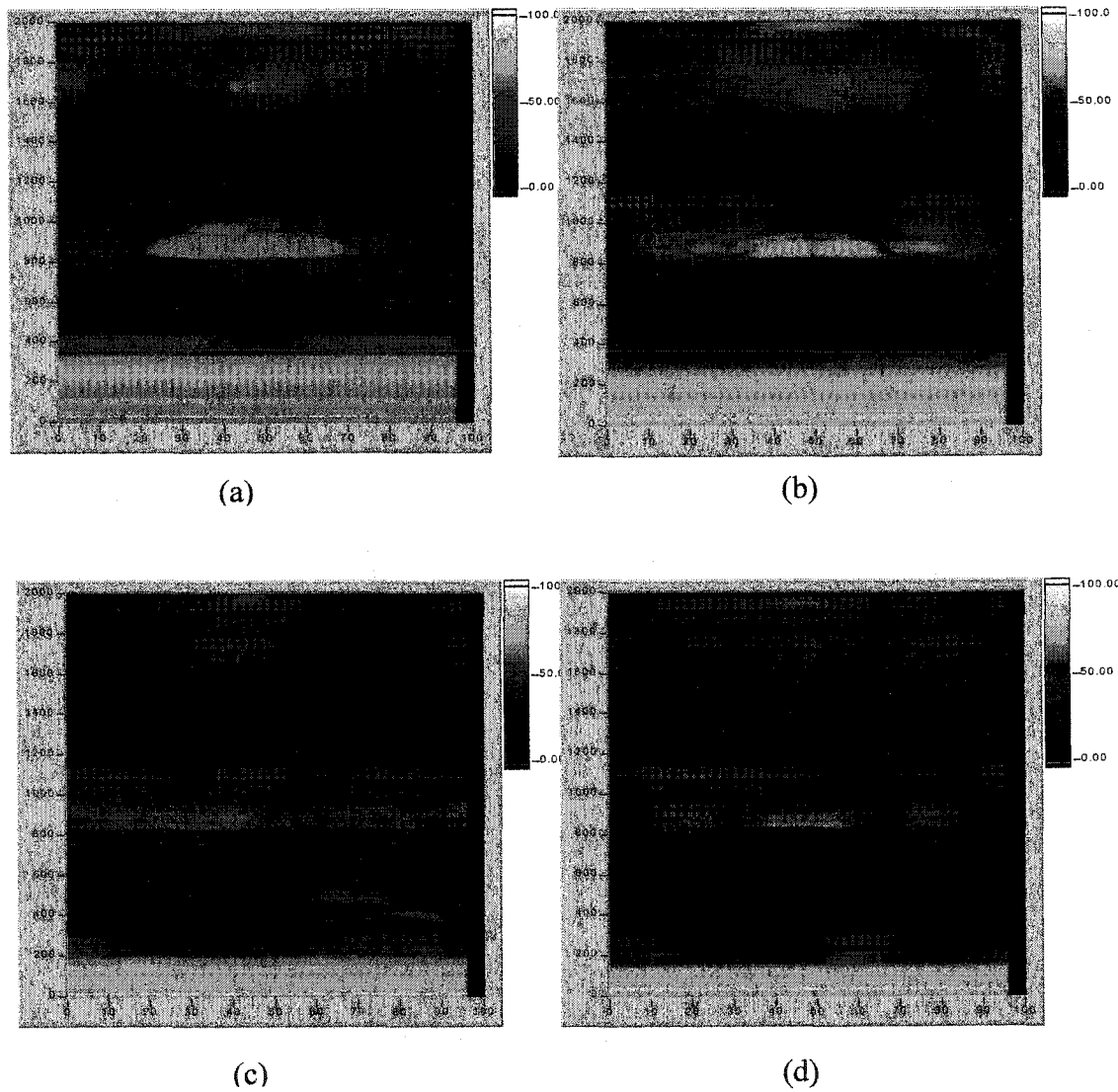


Figure 2.26 B-Scan images of the two angles when we use (a) RI (b) R7 (c) R4 (d)R2 as the transducer. The two angles are 5 inches apart. The distance from the sensor to the angles is 4 feet 8 inches.

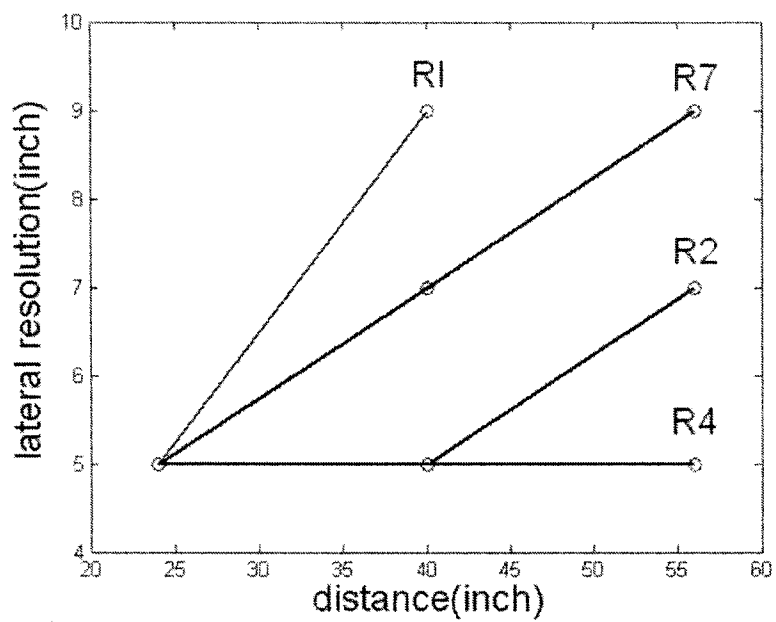


Figure 2.27 Lateral resolution results of the four transducers.

2.2.3 Sample Results from the Second Generation

Figure 2.28(a) shows a gate that was built with cinder blocks. Figure 2.28 (b) displays the original C-Scan image. Its top and bottom are enhanced in Figure 2.28 (c) because of their oblique incidence angles. It's then converted into a binary image in Figure 2.28 (d). Then, the MATLAB operation — “erosion” is conducted in Figure 2.28 (e). It removes everything but large continuous patches of foreground pixels. Finally, the MATLAB operation — “dilation” is performed in Figure 2.28 (f), restoring the rectangles to their original sizes. The gate shows up very well in the image, except that the image of the top block does not show up in the correct position.

Several other simple shapes such as the gate without the top (see Figure 2.29) and two towers (see Figure 2.30) were scanned. They all show up clearly after this series of image processing techniques are applied to the C-scan images. We also scanned some complicated cinder block shapes such as “H” shape tower (see Figure 2.31). The final C-scan image distorts because of the large beam. The degree of distortion depends upon how complicated the shape is. Usually, the more complicated the shape is, the more distorted is its final C-scan image. However, despite all those distortions, the final C-scan images have been improved compared to their original C-scan images.

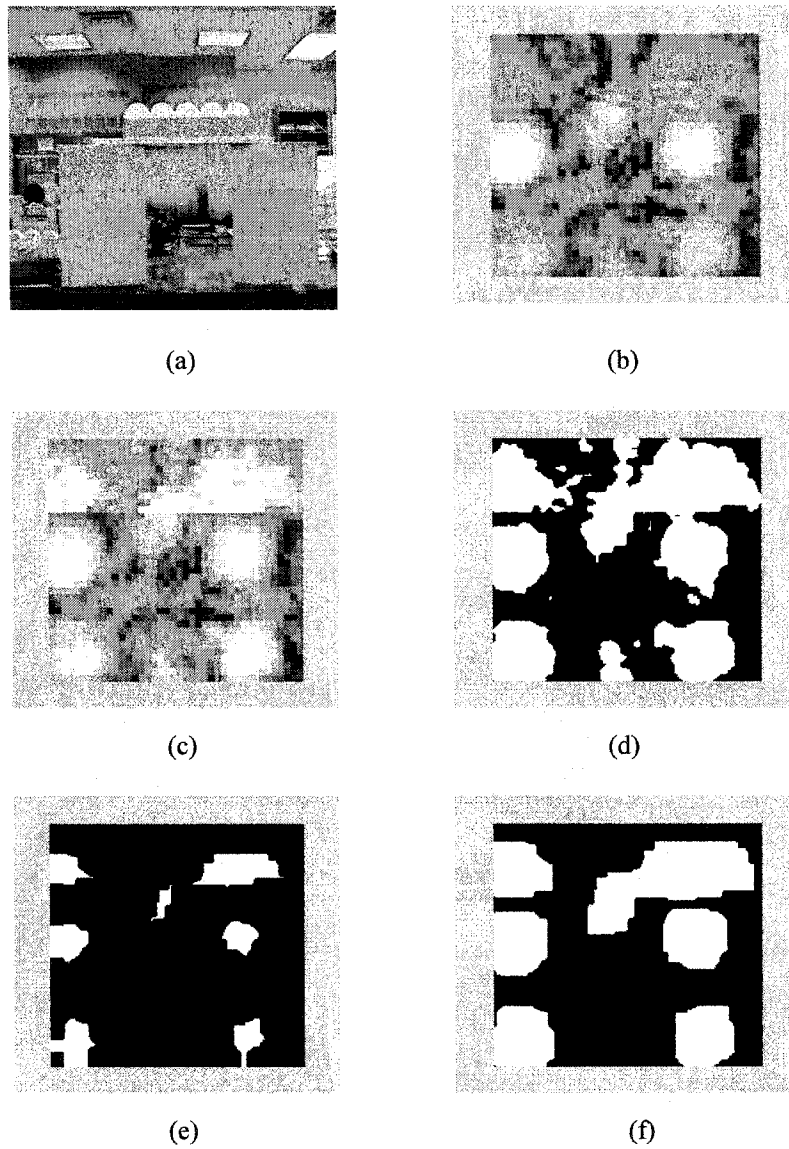


Figure 2.28 (a) Gate built with cinder blocks (b) C-Scan image of the gate. (c) Improved C-Scan image (d) Binary image of the C-Scan image (e) Binary C-Scan image after erosion (f) Binary C-Scan image after further dilation.

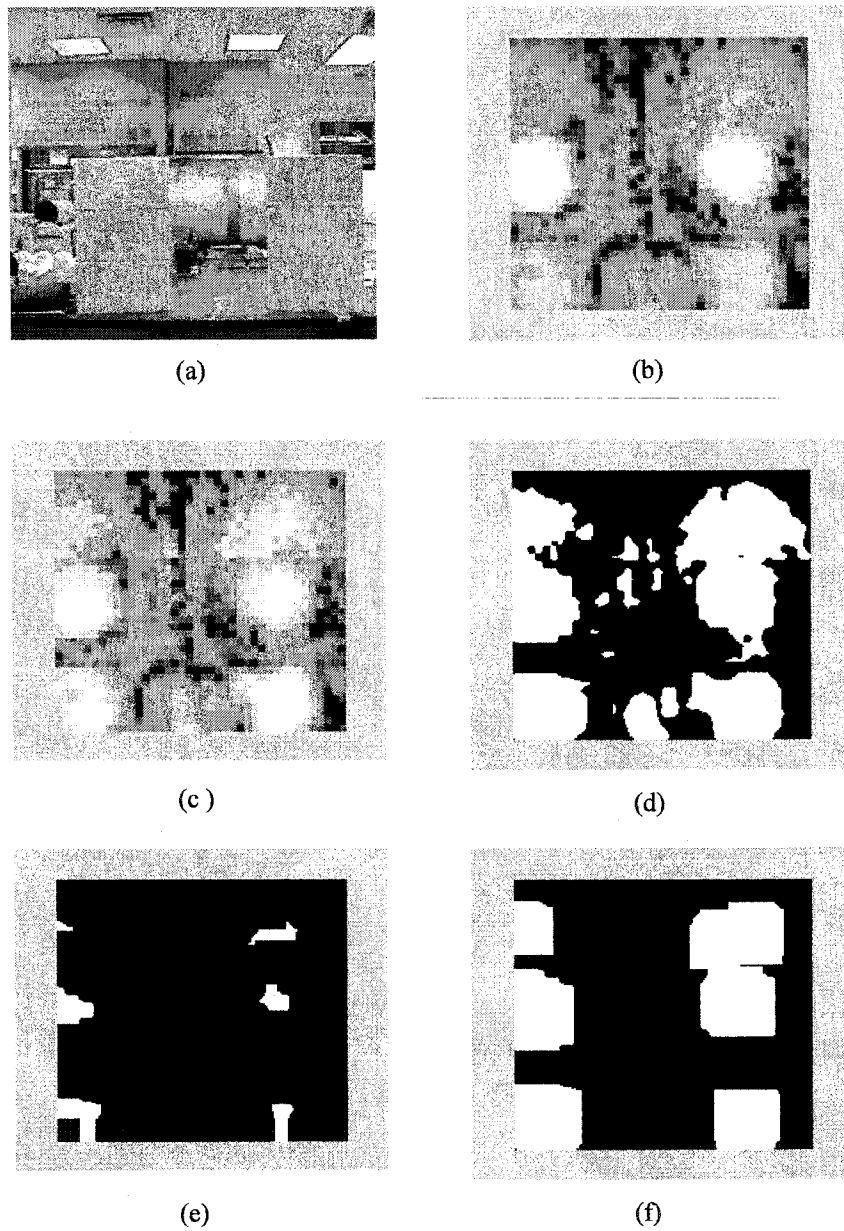


Figure 2.29 (a) Two columns of the cinder blocks (b) C-Scan image of the two columns of cinder blocks (c) Improved C-Scan image (d) Binary image of the C-Scan image (e) Binary C-Scan image after erosion (f) Binary C-Scan image after further dilation.

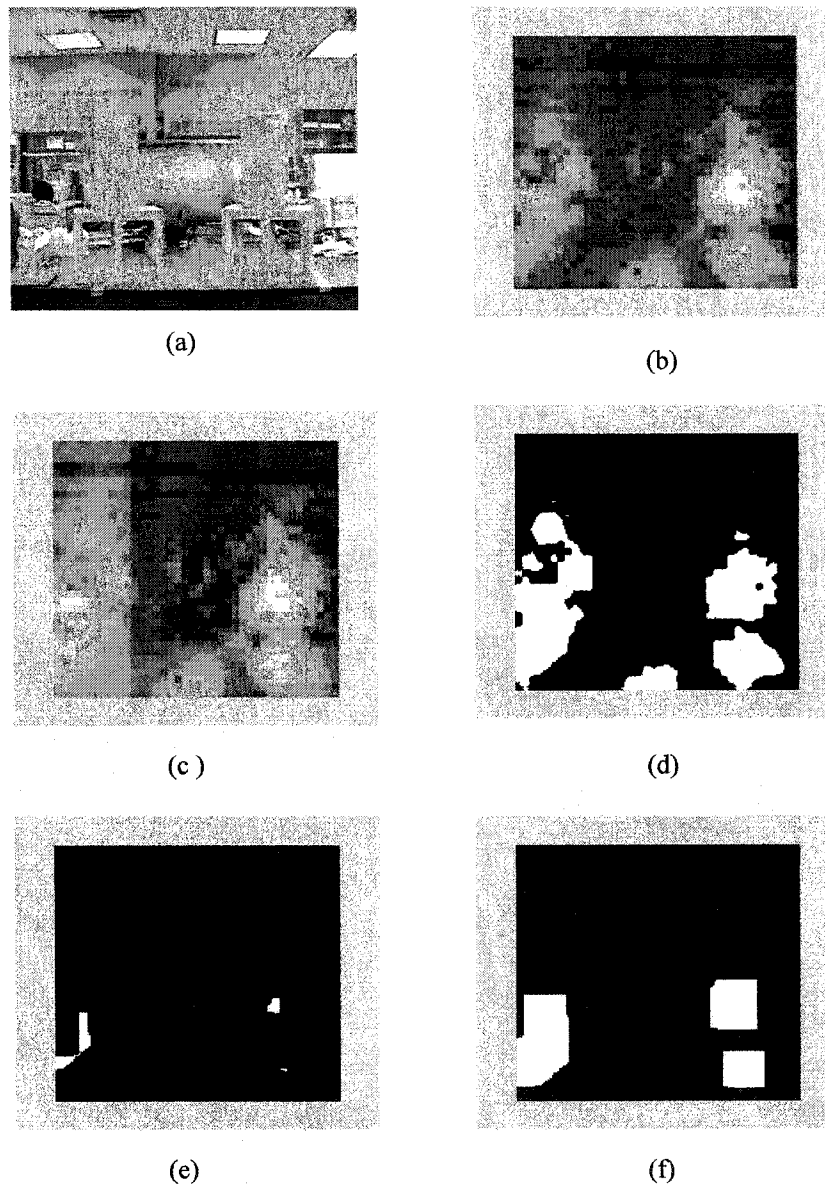


Figure 2.30 (a) Two-block tower (b) C-Scan image of the tower (c) Improved C-Scan image (d) Binary image of the C-Scan image (e) Binary C-Scan image after erosion (f) Binary C-Scan image after further dilation.

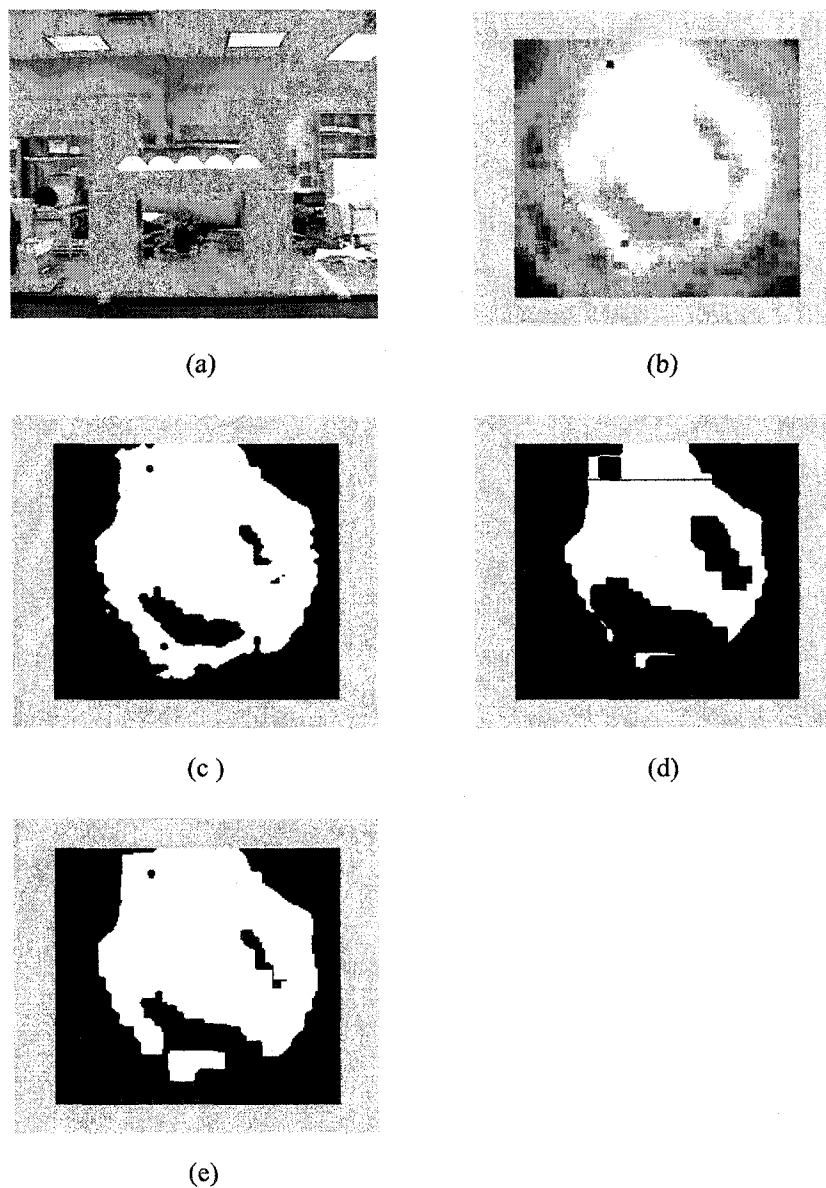


Figure 2.31 (a) “H” shape block (b) C-Scan image of the “H” shape block (c) Binary image of the C-Scan image (d) Binary C-Scan image after erosion (e) Binary C-Scan image after further dilation.

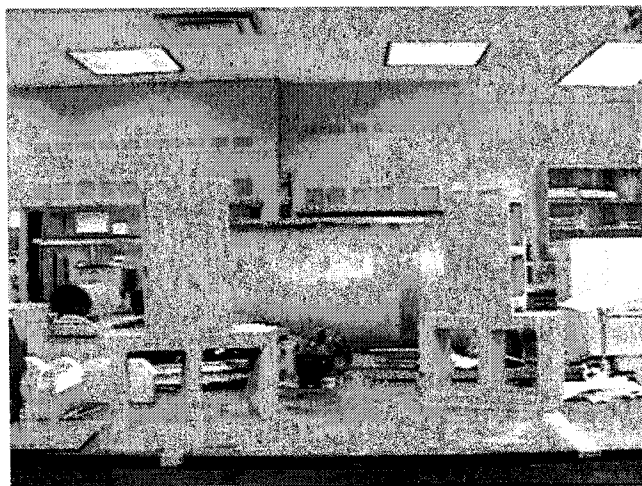
Mayavi is a software package that displays volume data in 3D. Program *ConvertDataToVTK.vi* was built to convert the original waveform data file into Visual Toolkit Format before Mayavi is run. Figure 2.32 shows two towers built by cinder blocks and their 3-D image in Mayavi. Mayavi gives a 3D ultrasonic picture of the two towers. Parameters such as Iso-surface value can be changed to improve the final 3D images.

2.3 Third Generation of Robotic Scanning System

2.3.1 Apparatus

Based on the first and second generation, a third generation of robotic scanning system has been developed. It's packaged as sensor head for a mobile robot — rWilliam. This makes possible for research outdoors. Figure 2.33 shows rWilliam scanning a brick wall.

The BitScope data acquisition board replaces the previous Gage CompuScope12100 data acquisition card. It communicates with the computer through a serial interface. After its LabVIEW drivers were improved, it worked as well as the Gage CompuScope 12100 card and is 1/30th the price of the Gage CompuScope. Another attractive feature of BitScope is that it is portable. However, the Gage CompuScope 12100 needs to be plugged into a PCI or ISA slot of a desktop. Therefore, a Dell Latitude laptop replaces the Dell RAID desktop. By doing so, the cost and volume of the total scanning system have been reduced dramatically.



(a)



(b)

Figure 2.32 (a) Two-block tower (b) 3-D image of the two-block tower.

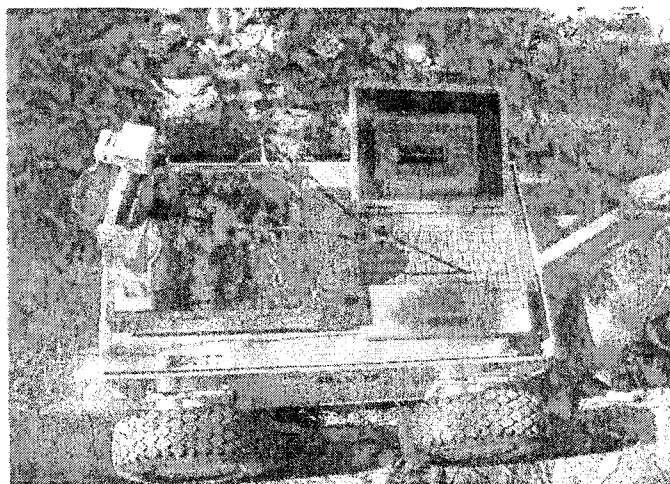


Figure 2.33 The third generation of robotic scanning system packaged as a sensor head for mobile robot —rWilliam.

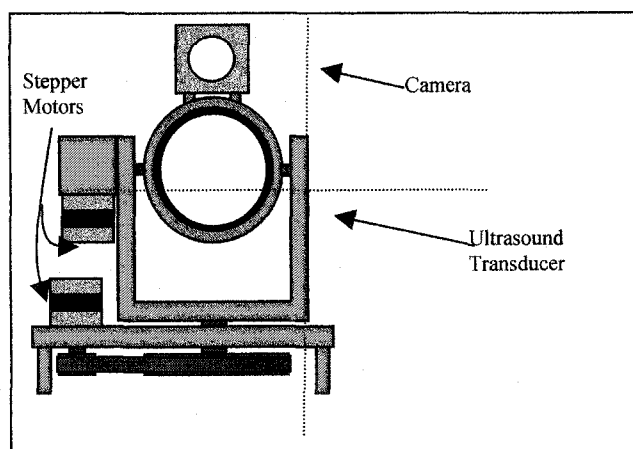
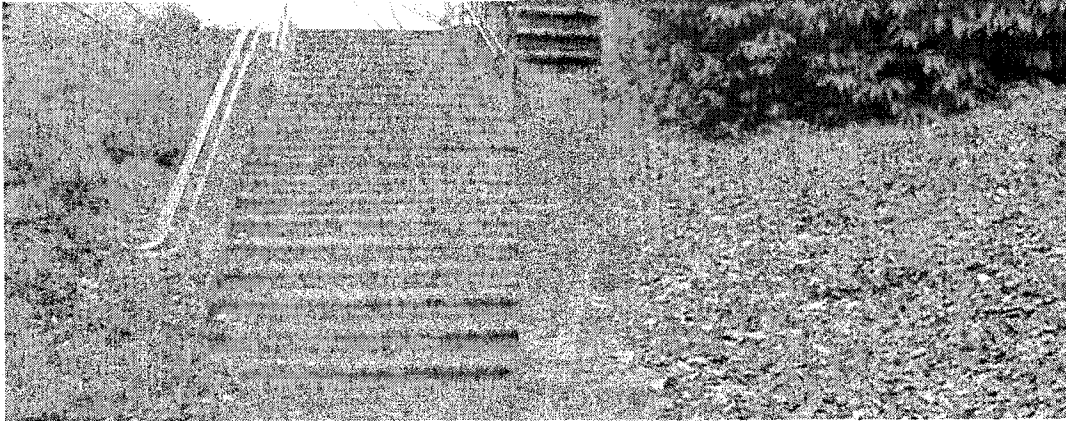


Figure 2.34 Diagram of the scanner head. The dotted lines represent the two axes that the camera and ultrasound transducer rotate about.

Figure 2.34 is a schematic of the biaxial sonar scan head. A narrow-beam Airmar AR50-8 transducer is used to obtain our results. Based on the previous transducer research on narrowing the beam, we modified the transducer by machining a concavity into the matching layer in order to achieve beam half-widths of less than 8° . It's actually R7 with a concave matching layer. It has a radius of curvature 7.3 inches. The transducer scans over perpendicular arcs via stepper motors. One motor controls rotation about the horizontal axis while the other controls rotation about the vertical axis. A custom motor controller board is connected to the computer via a serial interface. The robot's scanning process is paused briefly at each step while a series of sonar tone-bursts are generated and the echoes are recorded, digitized and archived on the computer.

2.3.2 Sample Results from the Third Generation —B-Scan and 3-D Images for Outdoor Objects

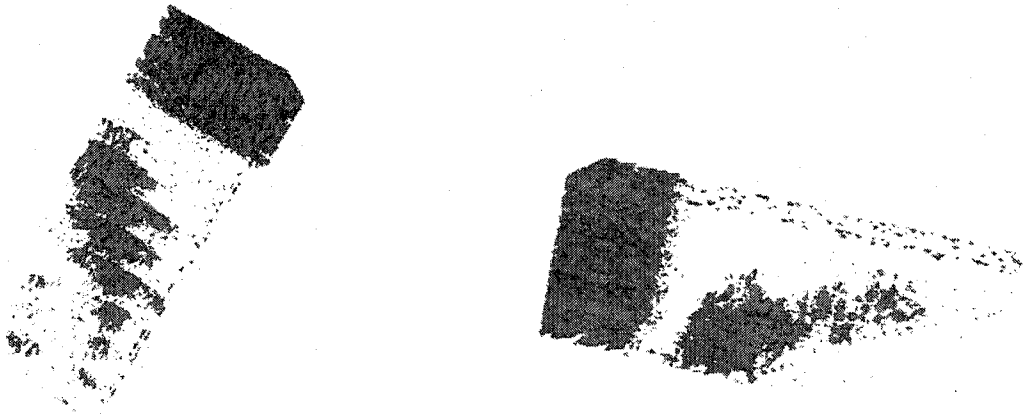
After the third generation of robotic scanning system was built, interesting objects outside can be studied for the first time. Figure 2.35 shows a staircase and some ivy near MS Hall at the College of William and Mary. Both have a slope. Figure 2.36 (a) and (b) show the 3-D image of the staircase and ivy respectively. The steps of the staircase show up very well in both B-scan (see Figure 2.37) and 3D images. The difference between staircase and ivy and their slopes also show up clearly in their B-Scan and 3D images.



(a)

(b)

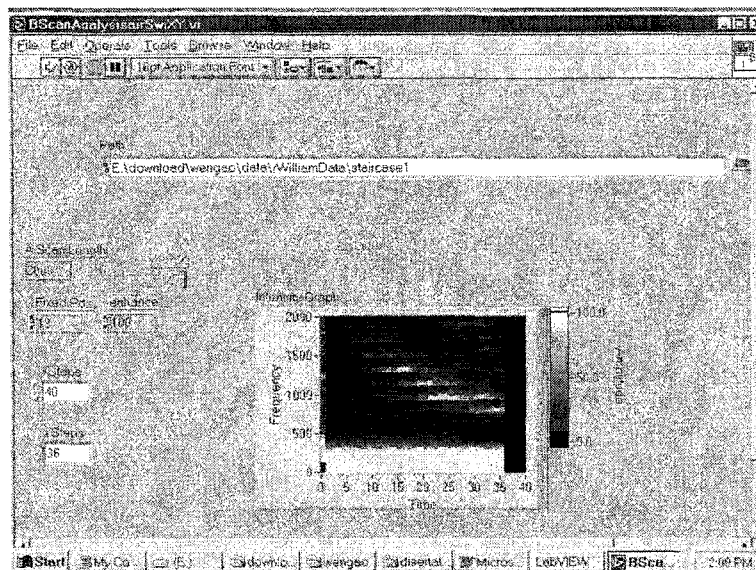
Figure 2.35 (a) Staircase (b) Ivy.



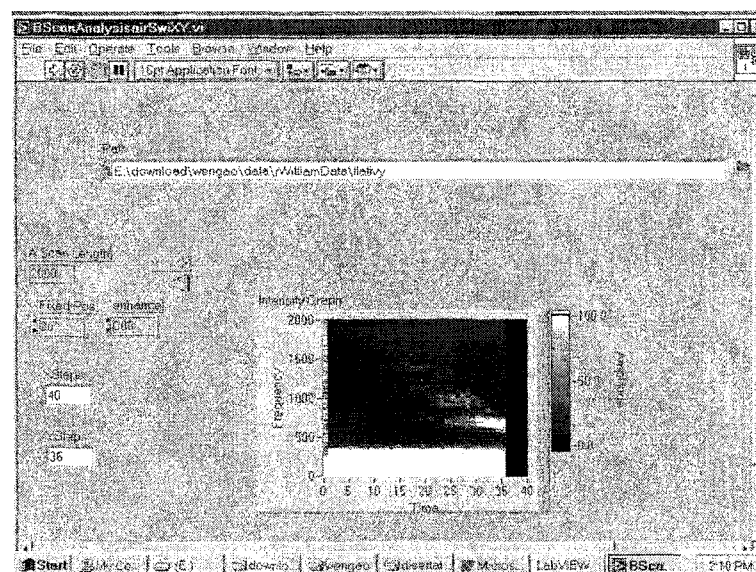
(a)

(b)

Figure 2.36 (a) 3-D image of the staircase (b) 3-D image of the ivy.



(a)



(b)

Figure 2.37 (a) B-scan image of the staircase (b) B-Scan image of the ivy.

2.4 Fourth Generation of Robotic Scanning System

The fourth generation of robotic scanning system is currently under construction. It will be packaged as a sensor head for another mobile robot —rMary. Figure 2.38 shows rMary on her first run down the hallway.

The Airmar AR50-8 narrow beam transducer will be replaced by a phased array system for better performance. A robotic arm will be built to hold the sensor system. This will allow for more freedom to move the sensor head. An embedded computer will replace the laptop to control its movement and acquire data.

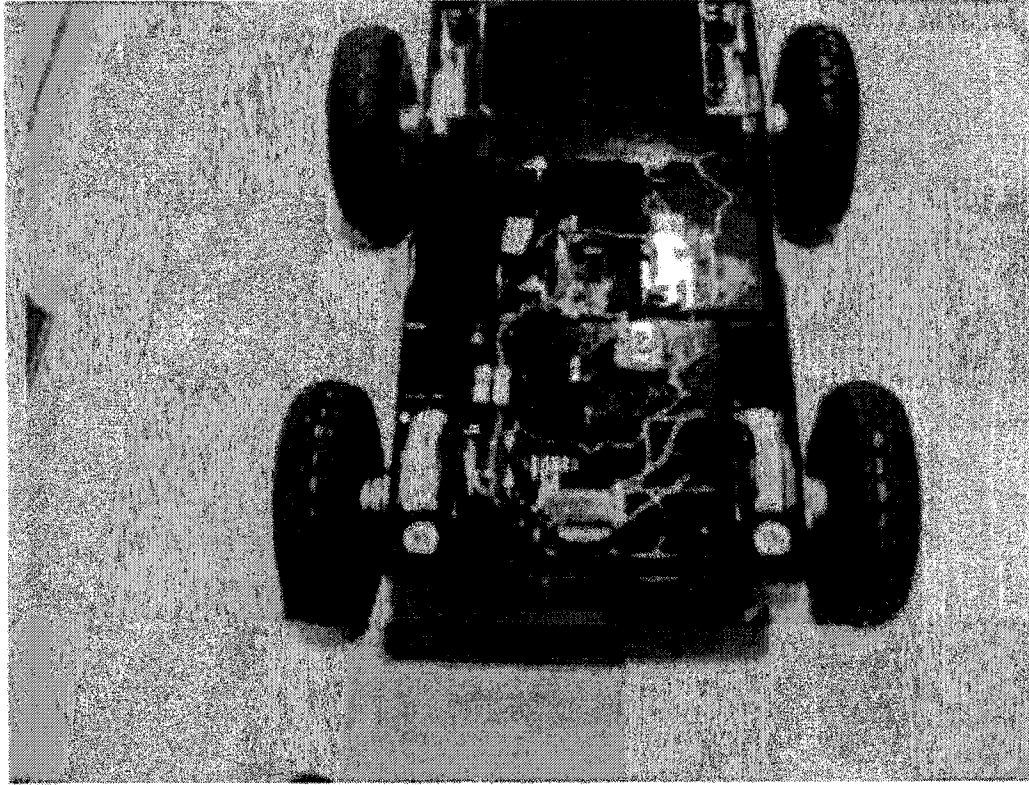


Figure 2.38 rMary on her first run down the hallway.

Chapter 3

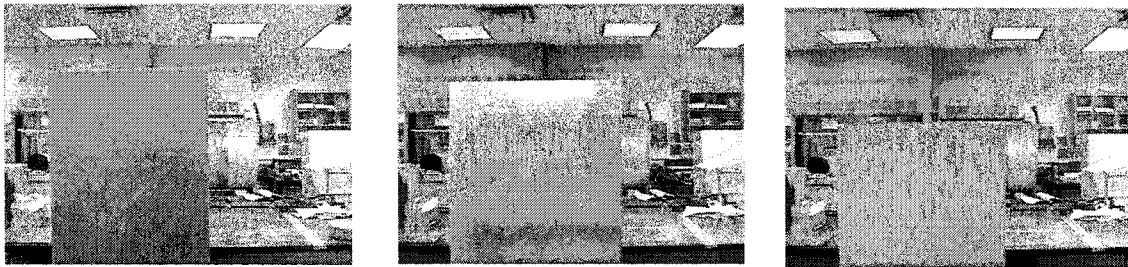
Sonar A-scan Classification Methods for Indoor Objects

Of the sample image results shown in Chapter 2, some give clear images of the object of interest while most still have poor spatial resolution. Since our goal is to teach robots to recognize objects so that they can respond accordingly, it is important that the scanning system has the ability to distinguish different objects. However, imaging comes to a dead end because of the large beam size. We need more sophisticated techniques to identify the material. Three methodologies are discussed in this chapter. Among these, template matching is the most effective one.

3.1 A-Scan Shape Analysis of Different Materials

In this section, we investigated echoes from three different materials — foam block, cardboard and aluminum.

Figure 3.1 shows their pictures when they are 1.5m away from the sensor. Figure 3.2 displays the echoes from the foam block, the aluminum plate and the cardboard respectively. The incident wave is perpendicular to the objects' surfaces. Different materials give different echoes. An echo from the foam is much weaker than those of aluminum and cardboard. Therefore, it's easy to tell the foam from the other two materials. Echoes from aluminum and cardboard are similar except that the tail of the echo from cardboard is bigger than that of the echo from an aluminum plate.

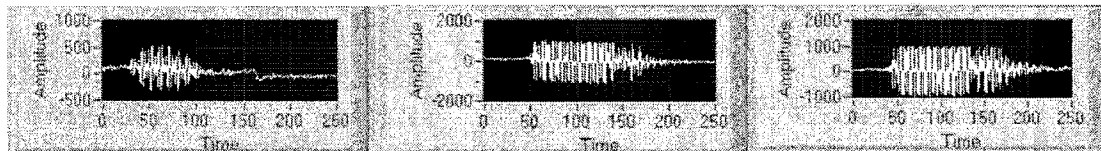


(a)

(b)

(c)

Figure 3.1 (a) Foam block (b) Cardboard (c) Aluminum plate.

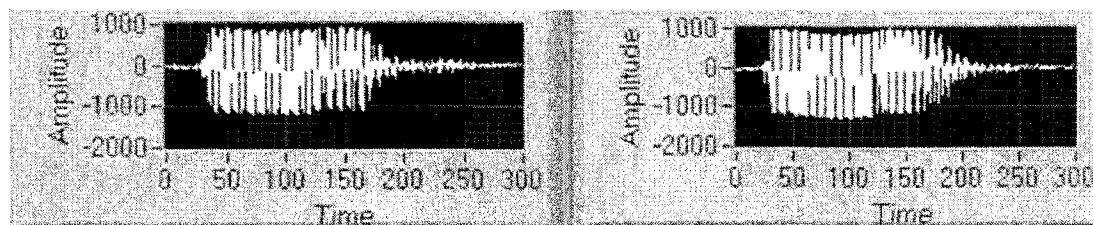


(a)

(b)

(c)

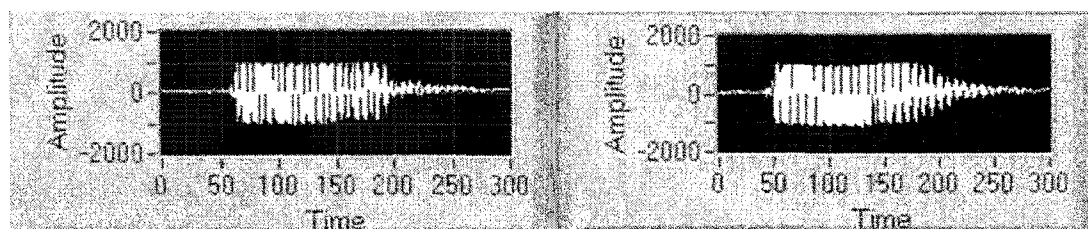
Figure 3.2 (a) Echo from the foam block (b) Echo from the Aluminum plate (c) Echo from the cardboard.



(a)

(b)

Figure 3.3 (a) Echo from the aluminum plate (b) Echo from the thin cardboard when the plates are 0.7 meter away from the sensor.



(a)

(b)

Figure 3.4 (a) Echo from the aluminum plate (b) Echo from the thin cardboard when the plates are 1.0 meter away from the sensor.

The cardboard we used previously was thicker than the aluminum plate. We don't know whether the thickness of the material or the material itself is the cause of the bigger tail. Here we tried another piece of cardboard with the same thickness as the aluminum plate. We will call it thin cardboard. Figure 3.3 displays the echoes from the aluminum plate and the thin cardboard respectively. The incident wave is perpendicular to the plates' surfaces. Both of the plates are 0.7 meter away from the sensor. The echo from the thin cardboard still has a bigger tail than echo from the aluminum plate.

We then changed the distances from the plates to sensor to 1.0 meter. Figure 3.4 (a) and (b) show the echoes respectively. Still, the echo from the thin cardboard has a bigger tail than echo from the aluminum plate. The tail of the echo when the incident wave is perpendicular to the plate surface is a possible way to distinguish aluminum from cardboard. The echo from cardboard has bigger tail which decreases slowly. The echo from aluminum plate has shorter tail which decreases rapidly.

Therefore, from the echoes obtained when the incident wave is perpendicular to the plates' surfaces, we can distinguish different materials. Echoes from materials with low impedance such as foam are much weaker than those from materials with high impedance such as aluminum and cardboard. For plates with high impedance, due to the extreme impedance mismatch at the plates' surfaces, almost all of the energy is reflected, which leads to large echoes. For plates with low impedance, there is only a small impedance mismatch at the plate-air interface and most of the energy is absorbed. The remaining energy reflects off the interface and leads to weak echoes.

Materials with high impedance such as cardboard and aluminum are hard to distinguish from each other because both give strong echoes. In this aluminum-cardboard case, we successfully distinguished them by investigating the shapes of their echoes. An echo from cardboard has larger tail which decreases slowly. An echo from an aluminum plate has a shorter tail which decreases rapidly.

3.2 Wavelet Analysis of A-scans from Different Objects

Wavelet and Fourier Transforms are both powerful techniques in signal processing. The Fourier Transform has non-local basis functions. Small changes of the signal in one place can affect a large number of transform coefficients of the basis functions. Compared to the Fourier Transform, wavelets have basis functions that are contained in finite regions. A local change of the signal only affects a limited number of the transform coefficients of the basis functions [64]. Since our A-scan waveforms of echoes are local in time, it's appropriate to use a transform that uses local basis functions. This better preserves the time information of the signals. Therefore, the wavelet transform is chosen to investigate the local difference between different signals in this section.

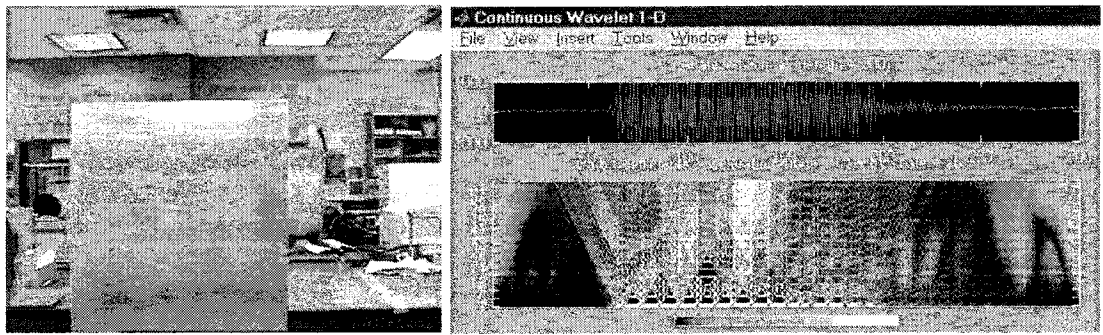
With wavelets, we can perform multi-resolution analysis, literally sorting signal components by their location and resolution scale. Whereas classical Fourier methods sort signals into their spectra, the wavelet transforms sort signal details into a locale-scale collection.

The Continuous Wavelet Transform in the Wavelet Toolbox in MATLAB is used to investigate the A-Scans of different objects. Again, the incident wave is perpendicular to the objects' front surfaces. The objects of interest are an aluminum plate, a piece of thin cardboard, a foam block, a plant, an electric fan with the power off, an electric fan with the power on, a stool, a chair, a chair with a bag on it and a chair with an airplane section on it.

Figures 3.5-3.7 show the aluminum plate, the cardboard and the foam block. It also shows their A-scans when the incident wave is perpendicular to the front surfaces and the corresponding CWT (Continuous Wavelet Transform). Each material has its typical A-scan shape and corresponding CWT patterns. An aluminum plate gives a strong echo with a short tail while cardboard gives a strong echo with a longer tail. This difference shows up on the right hand side of their CWT. The brightness in the CWT of Aluminum plate decreased abruptly due to its short tail while the brightness in the CWT of cardboard decreased gradually due to its longer tail. The foam block gives a weak echo and its CWT shows only a limited region corresponding to the echo is bright.

Figures 3.8-3.10 show the plant, the electrical fan with the power off and on, along with their perpendicular A-scans and the corresponding CWT patterns. The plant gives a long and weak echo. The electric fan with the power off gives a short and weak echo. The same fan with the power on gives a stronger echo. All these different echo shapes show up clearly as different patterns in their CWT 2D patterns.

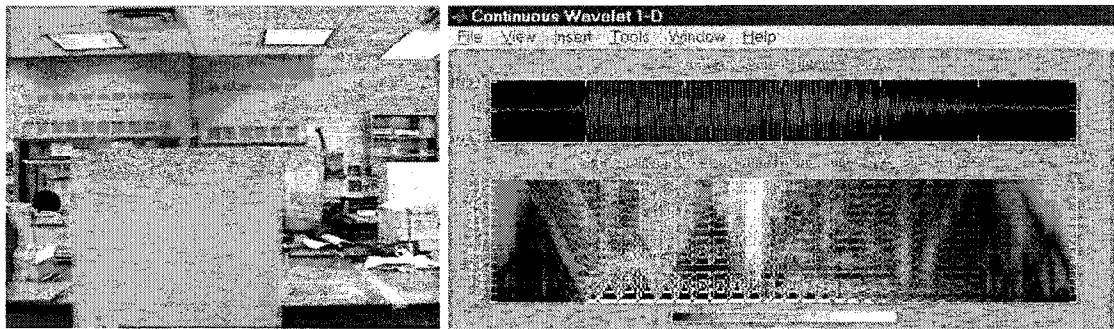
Figures 3.11-3.13 compare the chair, chair with a bag, chair with an airplane section and a stool showing their perpendicular A-scans and the corresponding CWT pattern. The echo from the back of the chair is weak. The echo from the chair with a bag on it is strong with a long tail. The echo from the chair with an air plane section is strong with a very short tail. The echo from the stool has two parts. All these different echo shapes show up clearly as different patterns in their CWT 2D patterns.



(a)

(b)

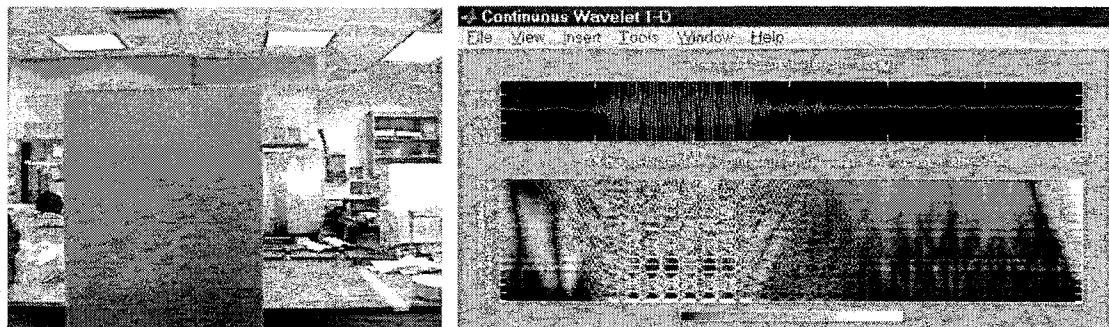
Figure 3.5 (a) Aluminum plate (b) A-scan of the aluminum plate when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan.



(a)

(b)

Figure 3.6 (a) Thin cardboard (b) A-scan of the thin cardboard when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan.



(a)

(b)

Figure 3.7 (a) Foam block (b) A-scan of the foam block when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan.

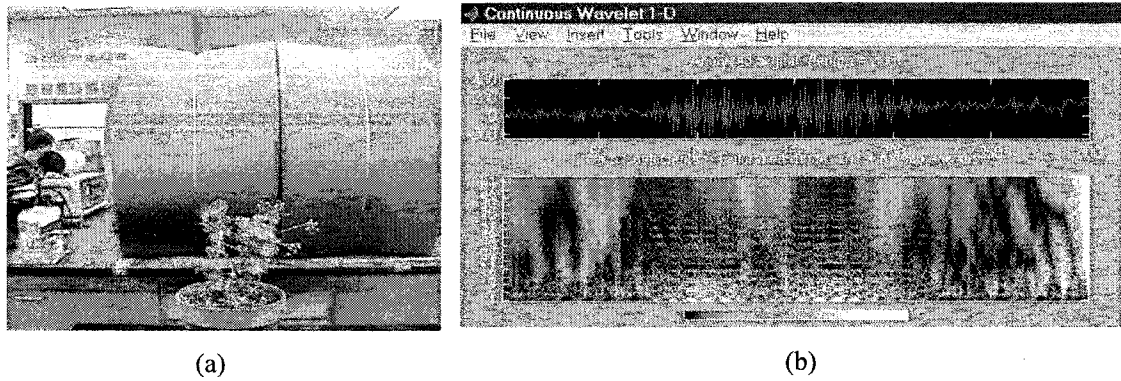


Figure 3.8 (a) Plant (b) A-scan of the plant when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan.

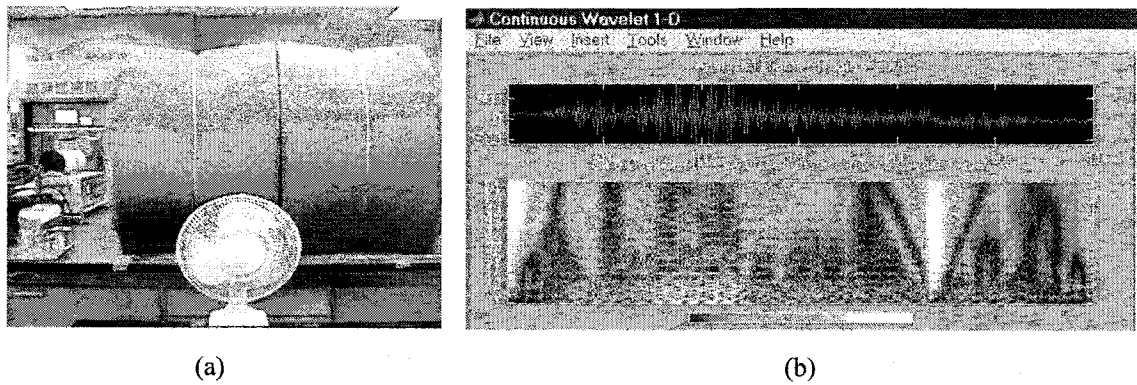


Figure 3.9 (a) Electrical fan with the power off (b) A-scan of the electrical fan with the power off when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan.

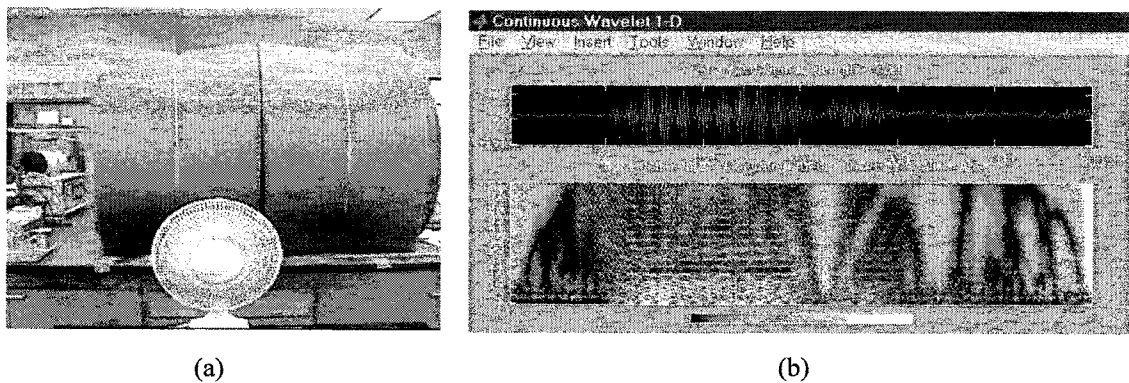


Figure 3.10 (a) Electrical fan with the power on (b) A-scan of the electrical fan with the power off when the incident wave is perpendicular to the front surface and the corresponding CWT of the A-Scan.

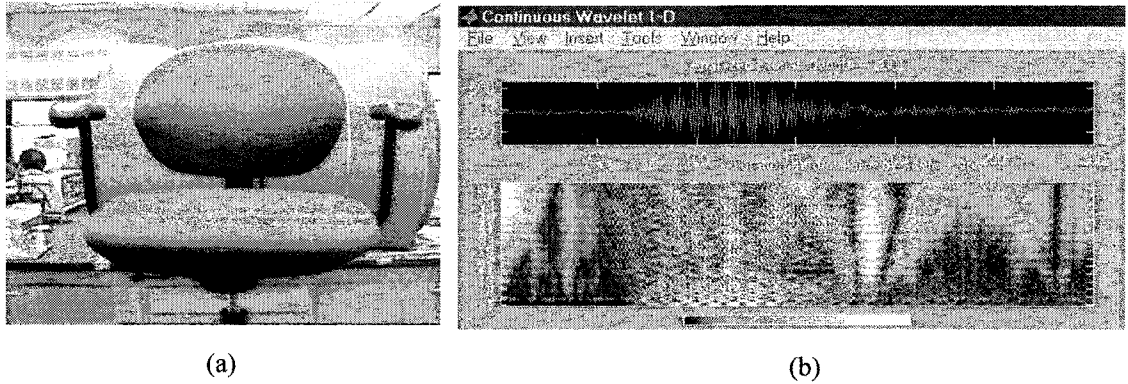


Figure 3.11 (a) Chair (b) A-scan of the chair when the incident wave is perpendicular to the front surface of its back and the corresponding CWT of the A-Scan.

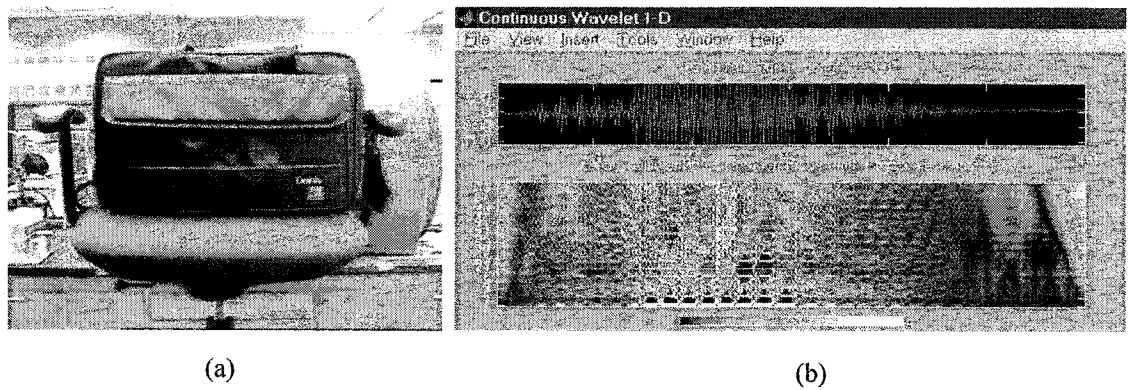


Figure 3.12 (a) Chair with a bag on it (b) A-scan of the chair when the incident wave is perpendicular to the front surface of the bag and the corresponding CWT of the A-Scan.

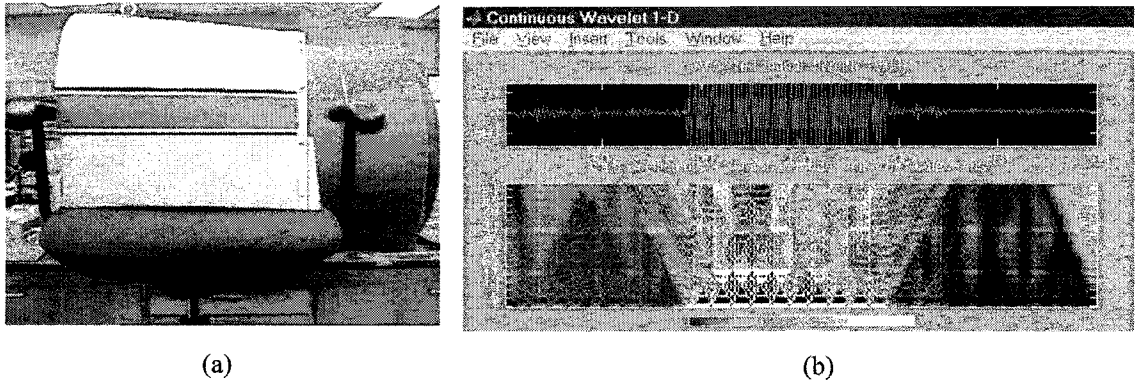


Figure 3.13 (a) Chair with an air plane section on it (b) A-scan of the chair when the incident wave is perpendicular to the front surface of the plane section and the corresponding CWT of the A-Scan.

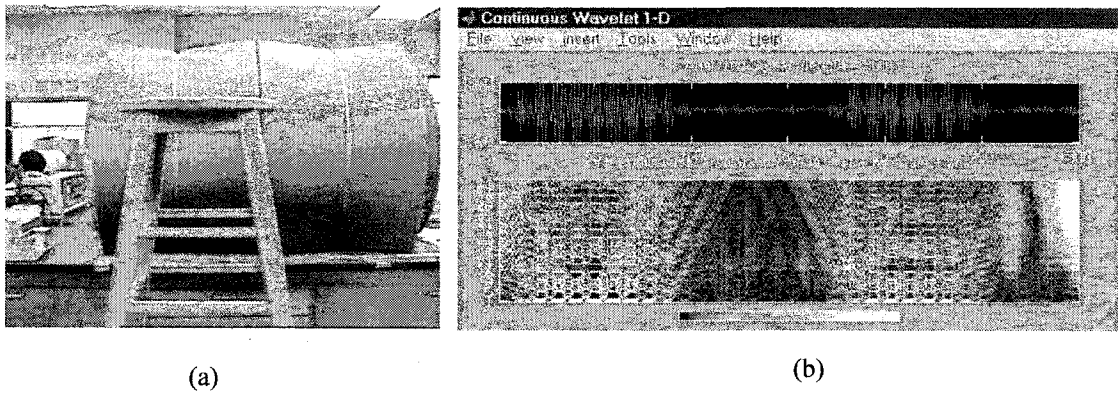


Figure 3.14 (a) Stool (b) A-scan of the chair when the incident wave is perpendicular to the side surface of its legs and the corresponding CWT of the A-Scan.

Different materials have different A-scans and the corresponding different continuous wavelet transform 2D pattern. Using CWT might be a possible way to distinguish between objects.

3.3 Template Matching Analysis

3.3.1 Development of Template Matching

In order to distinguish different objects, template matching is conducted to compare the waveform of the tested echo with the reference echoes stored in the library. According to this methodology, a set of reference patterns (*templates*) from known objects are assumed available. Then an unknown pattern (*test pattern*) is compared to the templates to see which it best matches.

Known as one of the most important techniques in pattern recognition, template matching has been studied for decades. In 1969, Azriel Rosenfeld introduced template matching to recognize objects from images [65]. Two techniques have been mentioned. One method is to compute the average difference between the two picture functions f and g . This can be done by calculating the absolute difference $\iint |f-g|$ or the squared difference $\iint (f-g)^2$ divided by the area over which the integration is performed. The other method is to compute their cross-correlation. The normalized cross correlation of two picture functions f and g can take on the value 1 only for displacements at which g exactly matches f .

In 1972, R Nagel and A. Rosenfeld described that digital template matching can be made more efficient by selecting the order in which template points are compared to the corresponding picture points [66].

Almost at the same time, Daniel I. Barnea and Harvey F. Silverman introduced a class of sequential similarity detection algorithms (SSDAs) to determine similarity between two structured data sets [67]. This method was far more efficient than other methods used that time.

In 1977, Gordon J. Vanderbrug and Azriel Rosenfeld developed a two-stage template matching with sum of absolute differences as the similarity measure [68][69]. They presented a methodology using only a sub-template and applying the rest of the template only when the sub-template's degree of match exceeds a threshold [68]. It reduces the computational cost of template matching dramatically. They also described another method of reducing the computational cost of the template matching [69]. A low-resolution ("coarse") template was applied to the low-resolution picture. A full-resolution ("fine") template was only used when the coarse template's degree of mismatch with the picture is below a given threshold.

Based on this early work in the 1970's, further work was done in 1980's. Rosenfeld and Kak in their book summarized the popular matching techniques at that time [70]. In the same year, Cheng and Huang presented a method of image registration where they used relational structures as representations of images and they registered two images by finding a match between the two relational structures [71]. This method is insensitive to scaling, rotation, gray level modification and small relative motion between two images. Therefore, it is better than traditional cross-correlation like techniques. Tanimoto presented Pyramids (Hierarchical template matching) which allows both a saving on

computation time and a considerable degree of insensitivity to noise [72]. An important feature of template matching using pyramids is the ability to easily enforce certain contextual constraints on the matching locations that would be difficult to enforce and even to express on the traditional nonhierarchical framework. Smith and Jain studied a shape matching method based on concepts tied to integral geometry [73]. The methods consist of computing the distribution of random chords over the figures to be matched and comparing them by means of the Kolmogorov-Smirnov test. Ryall and Sandor presented a technique based on goodness-of-fit statistics for classification of binary images via template matching [74]. The approach is essentially rotation invariant and is independent of the distribution generating the images.

In the 1990's and 2000's, template-matching technique was developed to its mature stage. Yuille and Jain made significant contributions in this area at that time. Yuille, Hallinan and Cohen proposed a method for detecting and describing features of faces using deformable templates, a parameterized template which described the feature of interest, an eye, for example [75]. The template interacts dynamically with the image by altering its parameter values and deforming itself to find the best fit. Further work has been done based on Yuille's work. Brunelli and Poggio developed and implemented two new algorithms for computer recognition of human faces [76]. One is based on the computation of a set of geometrical features. The other is based on almost-gray-level template matching. The results favor the implementation of the template-matching approach. Deng and Lai proposed an improved method for eye-feature extraction, descriptions, and tracking using deformable template [77]. Rather than using original

energy minimization for matching the templates in the traditional way, they proposed a region-based approach for template deformation. Miao et.al. presented a novel faster search scheme of gravity-center template matching. Compared with the traditional search method in an image for human face detection, it significantly saves the time consumed in rough detection of human faces in a mosaic image [78]. Kondo and Yan presented a system for automatic human face detection and recognition under non-uniform illumination [79]. Template-matching (a normalized cross correlation between the test face image and the face template) is used in the fourth step of the total five steps.

Su investigated similar techniques using image subtraction to find flaws in cosmetic products [80] very accurately with the inspection done by machine automatically.

Kawaguchi and Rizon proposed a new algorithm using intensity and edge information to detect the irises of both eyes from a face image [81]. The algorithm first detects the face region in the image and then extracts intensity valleys from the face region. Next, the algorithm extracts iris candidates from the valleys using the feature template and the separability filter. Finally, the algorithm selects a pair of iris candidates corresponding to the irises.

Jain et.al. proposed a general object localization and retrieval scheme based on object shape using deformable templates [82]. A Bayesian scheme, which is based on prior knowledge and the edge information in the input image, is employed to find a match between the deformed template and objects in the image. Computation efficiency is achieved by a coarse-to-fine implementation of the matching algorithm. Jain and Vailaya

described a method for trademark image database retrieval [83]. Their system achieved both the desired efficiency and accuracy using a two-stage hierarchy. In the first stage, simple and easily computable shape features are used to quickly search the database to generate a moderate number of plausible retrievals when a query is presented. In the second stage, the candidates from the first stage are screened using a deformable template matching process to discard spurious matches. Zhong and Jain addressed the problem of localizing objects using color, texture and shape [84]. The proposed algorithm operates in two stages. In the first stage, they use local texture and color features to find a small number of candidate images in the database. In the second stage, they use a deformable template matching method to match the query shape to the image edges at the locations that possess the desired texture and color attributes.

Much work has also been done following the work of Rosenfeld. Chou and Chen described a template matching method, called moment-preserving pattern matching. It reduces the computational complexity of template matching significantly and provides several advantages in hardware implementation [85]. Uenohara and Kanade presented a fast pattern matching algorithm with a large set of templates [86]. Their algorithm is based on typical template matching speeded up by the dual decomposition: the Fourier transform and the Karhunen-Loeve transform. Choi and Kim proposed a two stage algorithm for a rotation invariant template matching method based on a combination of the projection method and Zernike moments [87]. In the first stage, the matching candidates are selected using a computationally low cost feature. In the second stage,

rotation invariant template matching is performed only on the matching candidates using Zernike moments.

Lane and Stoner [88] have reported their work on the automatic interpretation of sector scan sonar imagery. They described a system capable of performing classifications of observed objects based on available feature measurements, such as size, shape, and gray level characteristics. The problem of determining feature measures is addressed by using qualitative measures to describe object features during matching for recognition. Mignotte et.al. [89] presented an original statistical classification method using a deformable template model to separate natural objects from man-made objects in an image provided by high-resolution sonar.

3.3.2 Template Matching Analysis — Initial Study

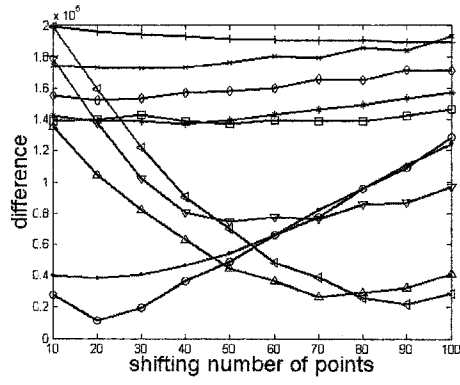
In this section, Template Matching Analysis is conducted to compare the test waveform from the unknown object with the reference echoes (templates) to find the best matches in order to identify the object. There are ten reference echoes (template) in the library, representing echoes from an aluminum plate, a cardboard, a foam block, a plant, a brick wall, brick wall with ivy, a cinder block, a chair, chair with bag and a thin cardboard respectively. All ten tested echoes are also from these ten different objects.

The first step of the analysis is envelope extraction. For a single echo, data points larger than both its left and right immediate neighbors are extracted as the envelope of the echo.

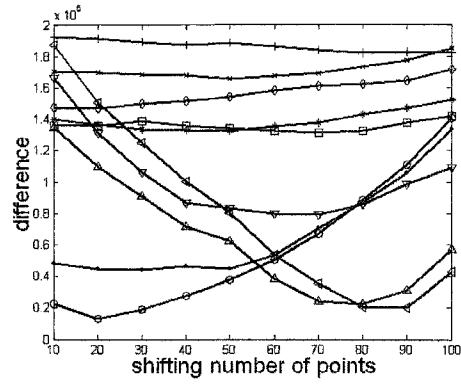
The second step is the comparison. The envelopes of the tested echoes are compared to those of the reference echoes. When two echoes are compared to each other, the difference depends not only on their shapes, but also on their relative position. In order to make a better comparison, we fixed the location of the reference echo, moved the tested echo step by step from left to right, and obtained the difference of the two echoes at each step. Here a ten-step-move with the step size 10 data points is conducted. A difference vs. step curve with the total range of 100 data points is obtained. A curve with a small minimum difference value and a large curvature means a good match. The minimum difference value is chosen to describe the difference of the two echoes.

Figures 3.15 (a)-(i) show the comparisons of the ten reference echoes with the tested echoes from an aluminum plate, a cardboard, a foam block, a plant, a brick wall, brick wall with ivy, a cinder block, a chair, chair with bag and a thin cardboard respectively.

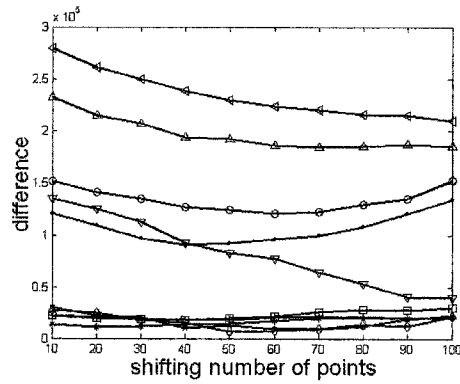
In Figure 3.15 (a), when the tested echo compares with the reference echoes from cardboard, aluminum plate, chair with bag and thin cardboard, the difference curves have small minimum value and large curvatures. Therefore, echoes from cardboard, aluminum plate, chair with bag and thin cardboard are the reasonable choices for the tested echo, which is the echo from the aluminum plate. Figure 3.15 (b) shows the difference curves have small minimum value and large curvatures when the tested echo compares with the reference echoes from cardboard, aluminum plate, chair with bag and thin cardboard. Consequently, echoes from cardboard, aluminum plate, chair with bag and thin cardboard



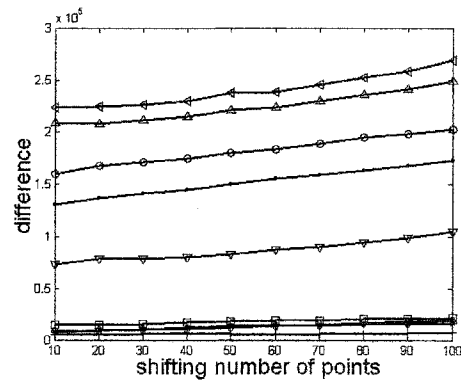
(a)



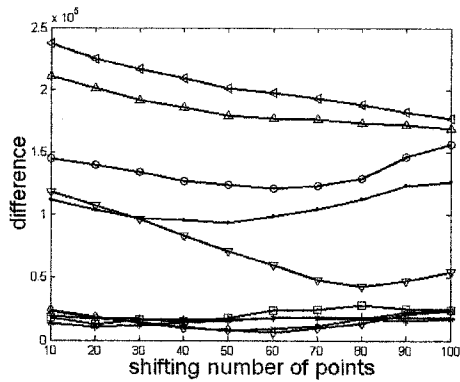
(b)



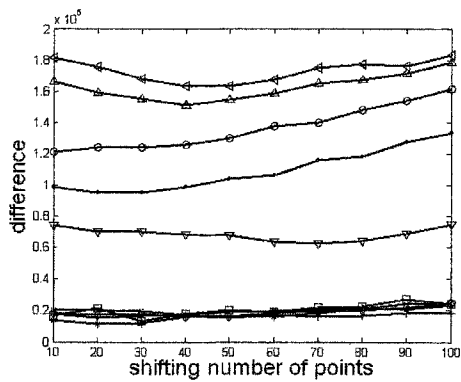
(c)



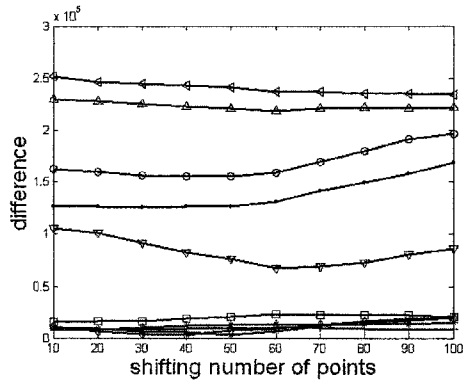
(d)



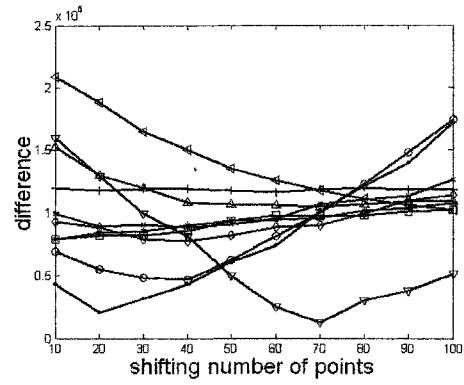
(e)



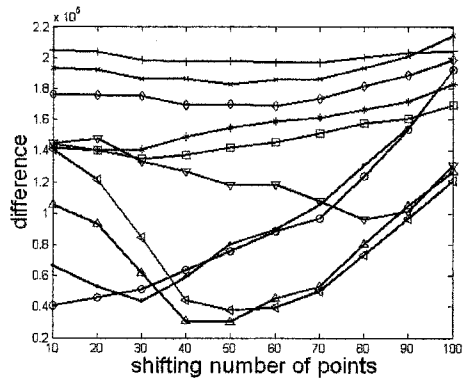
(f)



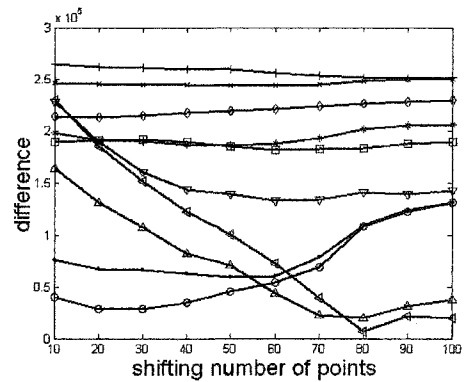
(g)



(h)



(i)



(j)

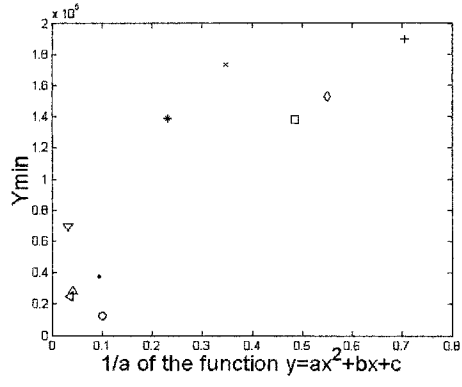
Figure 3.15 The comparison of the tested echo from (a) Aluminum plate (b) Cardboard (c) Foam block (d) Plant (e) Brick wall (f) Ivy on Brick Wall (g) Block (h) Chair (I) Chair with Bag (j) Thin Cardboard with the ten reference echoes from ten different objects in the library. ‘•’ represents aluminum plate. ‘◦’ represents card board. ‘×’ represents foam block. ‘+’ represents plant. ‘*’ represents brick wall. ‘□’ represents ivy on brick wall. ‘◊’ represents block. ‘∇’ represents chair. ‘Δ’ represents chair with bag. ‘◁’ represents thin card board.

are the reasonable choices for the tested echo, which is actually the echo from the cardboard.

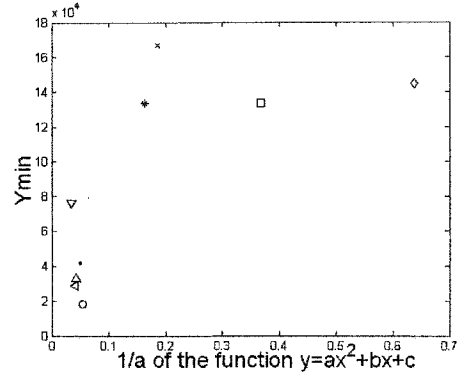
Similarly, from Figure 3.15 (c)-(j), we can also make a reasonable guess about the possible kinds of the unknown object based on the difference curves of the tested echoes with the reference echoes. However, some are different from the others. For example, Figure 3.15(c) shows when the tested echo compares with the reference echoes from foam block, plant, brick wall, ivy on brick wall and block, difference curves have small minimum values. None of the curves have large curvatures. Therefore, we choose those curves with small minimum values. Echoes from foam block, brick wall, ivy on brick wall and block are then the reasonable choices for the tested echo, which is actually the echo from the foam block.

3.3.3 Template Matching Analysis — Automatic Feature Extraction and Classification

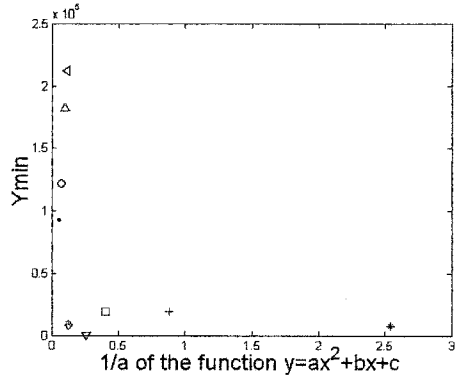
The difference curve with a small minimum value and a large curvature means a good match. However, it's still hard to determine a good match by simply looking at the curves. In this section, we will describe a method where features are extracted automatically so that the computer can make the decision automatically. First, the difference curves are fitted with a parabolic function $y=ax^2+bx+c$, whose curvature depends on a . Its minimum value $(4ac-b^2)/4a$ depends on a , b , and c . Then two features are extracted — the reciprocal of the 2nd order coefficient $1/a$ and the minimum value $(4ac-b^2)/4a$ from the parabolic. Then these values are plotted on a phase plane. The closer $1/a$ and $(4ac-b^2)/4a$ to the origin $(0,0)$ the better the match. Figure 3.16 shows the $1/a$ vs. $(4ac-b^2)/4a$ phase plane for the ten tested echoes respectively.



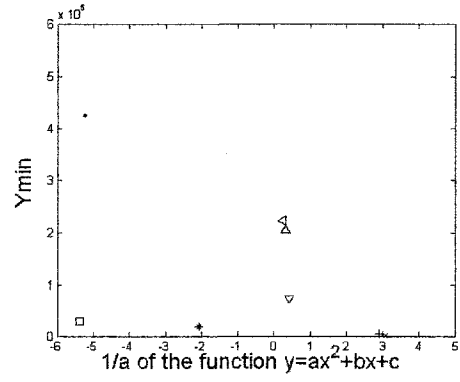
(a)



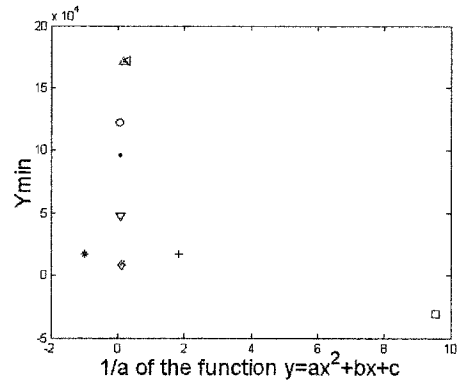
(b)



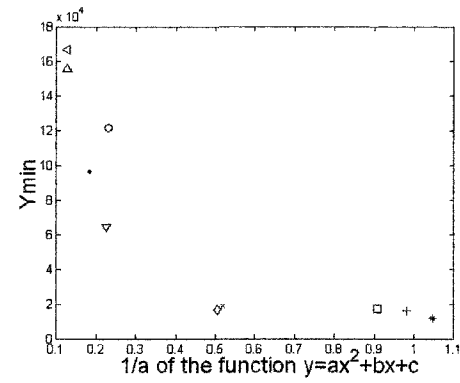
(c)



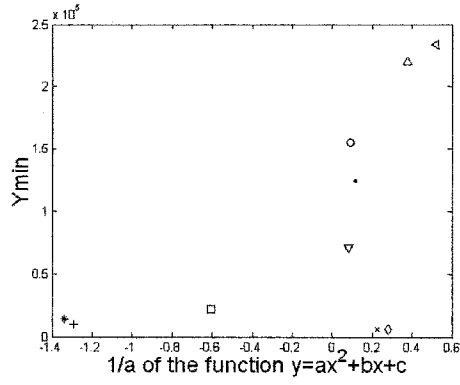
(d)



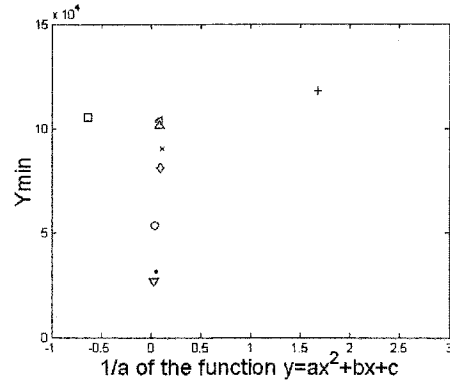
(e)



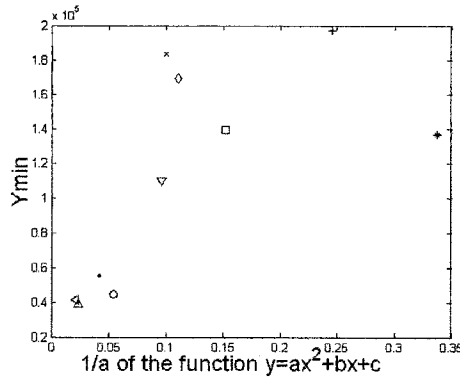
(f)



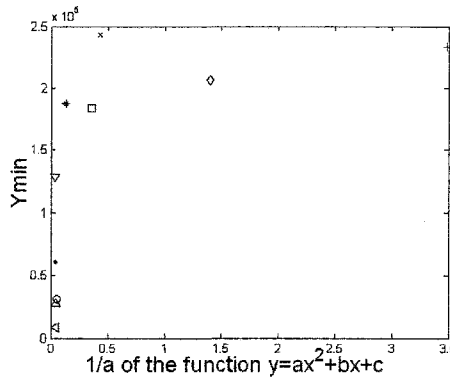
(g)



(h)



(i)



(j)

Figure 3.16 The phase plane of the comparison of the tested echo from (a) Aluminum plate (b) Cardboard (c) Foam block (d) Plant (e) Brick wall (f) Ivy on Brick Wall (g) Block (h) Chair (I) Chair with Bag (j) Thin Cardboard with the ten reference echoes from ten different objects in the library. ‘•’ represents Al plate. ‘◦’ represents card board. ‘×’ represents foam block. ‘+’ represents plant. ‘*’ represents brick wall. ‘◻’ represents ivy on brick wall. ‘◊’ represents block. ‘∇’ represents chair. ‘Δ’ represents chair with bag. ‘◁’ represents thin card board.

Table 3.1 shows the results from the computer.

Table 3. 1 Results from the computer for single echo analysis

<i>Tested echo</i>	<i>Results</i>	<i>1st choice</i>	<i>2nd choice</i>	<i>3rd choice</i>	<i>4th choice</i>	<i>5th choice</i>
Aluminum plate	cardboard	thin cardboard	chair with bag	aluminum plate	chair	
Cardboard	cardboard	thin cardboard	chair with bag	aluminum plate	chair	
Foam block	chair	block	foam block	ivy on brick wall	plant	
Plant	brick wall	plant	foam block	block	ivy on brick wall	
Brick wall	block	foam block	brick wall	plant	chair	
Ivy on brick wall	brick wall	block	plant	ivy on brick wall	foam block	
Block	foam block	block	plant	brick wall	ivy on brick wall	
Chair	chair	aluminum plate	card board	block	foam block	
Chair with bag	chair with bag	thin card board	card board	aluminum plate	chair	
Thin card board	thin card board	chair with bag	card board	aluminum plate	chair	

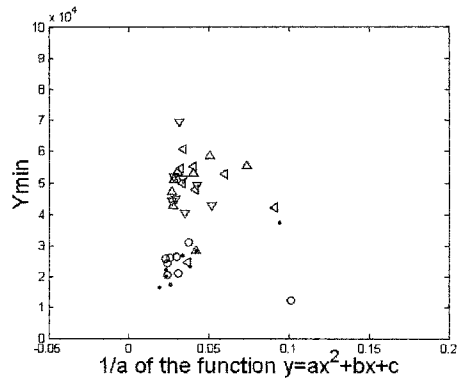
For the tested echo from aluminum plate, aluminum plate is selected as the 4th choice. For the tested echo from cardboard, cardboard is selected as the 1st choice. For the tested echo from foam block, foam block is selected as the 3rd choice, etc. Among the ten tested echoes, four are selected as the 1st choice, two are selected as the 2nd choice, two are selected as the 3rd choice and two are selected as the 4th choice. All of the right choices are within the first five choices. With this methodology, work has been reduced dramatically while promising results have been obtained.

3.3.4 Template Matching Analysis — Statistical Method

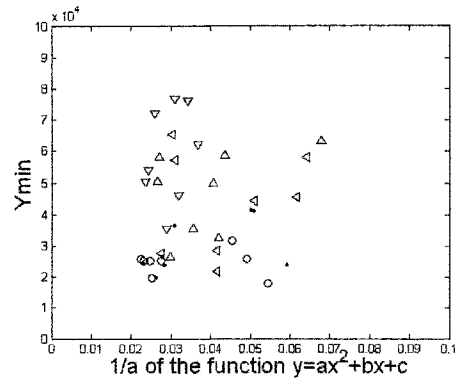
In the previous section, there are ten tested echoes, each from one object. They may or may not represent the characteristic of the echo from the object. To avoid random errors, eight tested echoes are acquired for each object in this section. Figure 3.17 shows the corresponding phase planes.

Figure 3.17 (a) demonstrates that the data points representing echoes from the aluminum plate and cardboard are clustered together near the origin $(0,0)$. They are the most reasonable choices of the eight tested echoes, which are actually the echoes from the aluminum plate. Data points representing echoes from thin cardboard, chair with bag and chair are the second nearest groups from the origin. Consequently, they are the 3rd, 4th and 5th choices. Data points representing echoes from block, plant, brick wall, ivy on brick wall and block are far beyond the range. They are not considered as the possible choices.

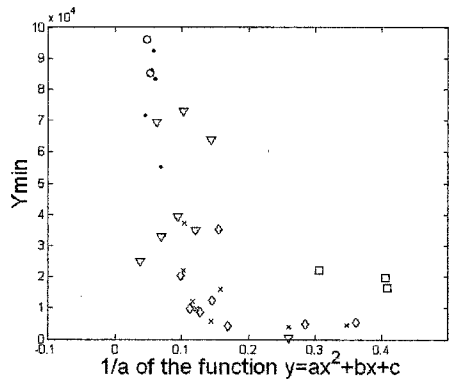
Figure 3.17 (b) indicates data points representing echoes from aluminum plate and cardboard cluster together nearest to the origin. They are the best choices of the eight tested echoes, which are actually the echoes from the cardboard. Data points representing echoes from thin card board, chair with bag and chair are the second nearest groups from the origin. Therefore, they are the 3rd, 4th and 5th choices. Data points representing echoes from other materials are out of range and not considered as matches.



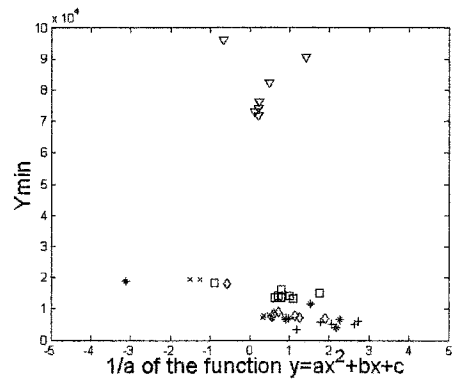
(a)



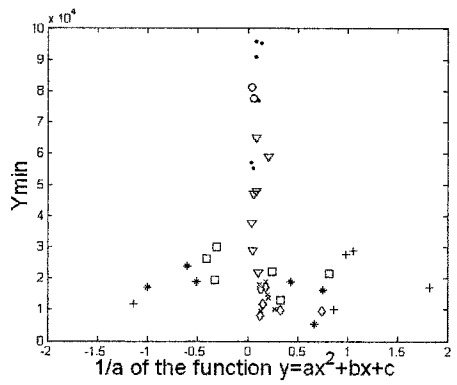
(b)



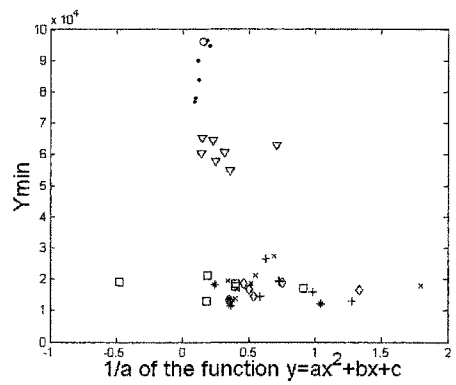
(c)



(d)



(e)



(f)

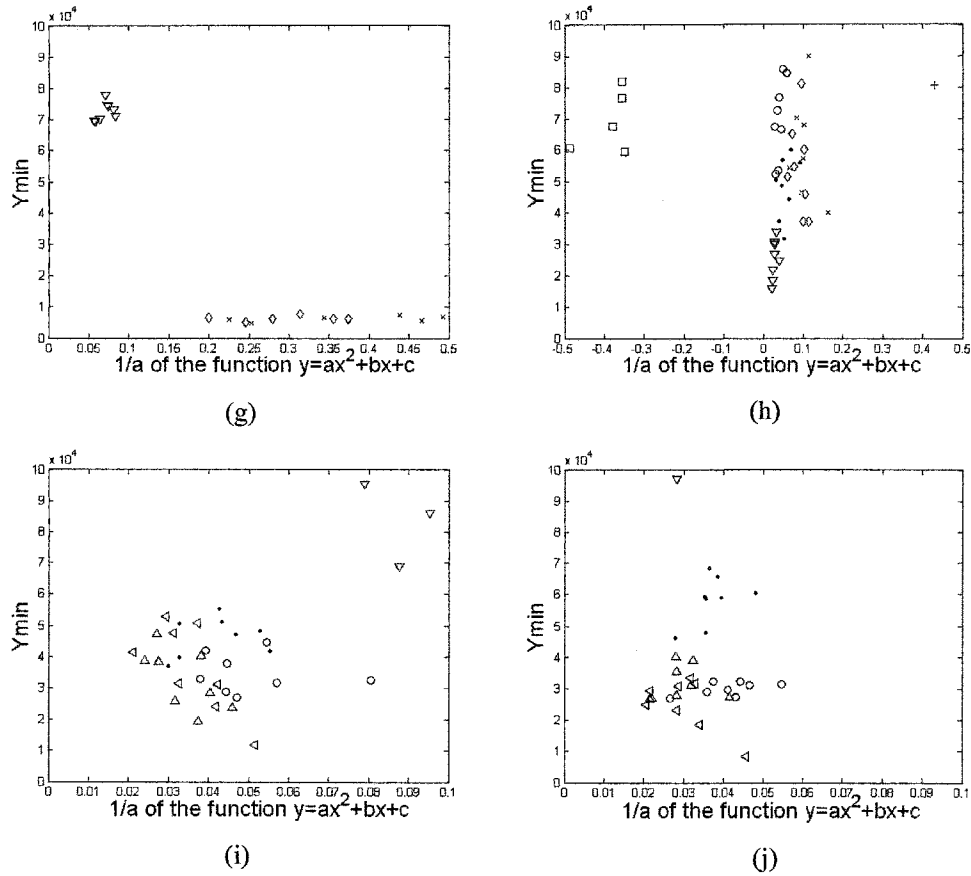


Figure 3.17 The statistical research of the phase plane of the comparison of the tested echo from (a) Aluminum plate (b) Cardboard (c) Foam block (d) Plant (e) Brick wall (f) Ivy on Brick Wall (g) Block (h) Chair (I) Chair with Bag (j) Thin Cardboard with the ten reference echoes from ten different objects in the library. ‘●’ represents Al plate. ‘○’ represents card board. ‘×’ represents foam block. ‘+’ represents plant. ‘*’ represents brick wall. ‘□’ represents ivy on brick wall. ‘◇’ represents block. ‘▽’ represents chair. ‘△’ represents chair with bag. ‘◁’ represents thin card board.

Figure 3.17 (c)-(j) also display the statistical results of the phase planes for the comparison of the tested echoes from foam block, plant, brick wall, ivy on brick wall, block, chair, chair with bag and thin cardboard respectively. The best choices are clustered near the origin.

Table 3.2 shows the results of the computer decision. Distances of the average point of each tested echo group to the origin $(0,0)$ are calculated and compared. Object with the shortest distance is the 1st choice. Object with the second shortest distance is the 2nd choice, etc.

Table 3.2 Results from the computer for multiple echoes analysis

<i>Tested echo</i>	<i>Results</i>	<i>1st choice</i>	<i>2nd choice</i>	<i>3rd choice</i>	<i>4th choice</i>	<i>5th choice</i>
Aluminum plate	cardboard	aluminum plate	thin card board	chair with bag	chair	
Cardboard	cardboard	aluminum plate	thin card board	chair with bag	chair	
Foam block	block	foam block	plant	ivy on brick wall	brick wall	
Plant	foam block	ivy on brick wall	block	plant	brick wall	
Brick wall	block	brick wall	plant	foam block	ivy on brick wall	
Ivy on brick wall	brick wall	ivy on brick wall	plant	foam block	block	
Block	block	foam block	ivy on brick wall	brick wall	plant	
Chair	chair	aluminum plate	block	foam block	card board	
Chair with bag	chair with bag	card board	thin card board	aluminum plate	chair	
Thin card board	thin card board	card board	chair with bag	aluminum plate	chair	

For the tested echoes from aluminum plate, aluminum plate is selected as the 2nd choice. It is better than the previous 4th choice. For the tested echoes from cardboard, cardboard is selected as the 1st choice. It is the same as the previous 1st choice. For the tested echoes from foam block, foam block is selected as the 2nd choice, which is better than the previous 3rd choice. Among the ten groups of tested echoes, five are selected as the 1st choice instead of previous four. Four are selected as the 2nd choice instead of the

previous two. One is selected as the 4th choice. All of the right choices are within the first four choices. 90% are in the first two choices. With this statistical analysis, results have been greatly improved while the random errors have been dramatically reduced.

3.4 Summary

In this chapter, three methodologies have been discussed to distinguish different materials. All are based on the A-scan echoes from the materials. In the A-scan analysis, different shapes of the A-scan waveforms are used to distinguish different materials. In the wavelet analysis, different 2D patterns of the wavelet transforms of A-scan echoes from objects are used to distinguish different materials and objects. In the template matching analysis, template matching is conducted by comparing the tested echo with the reference echoes. Important features are then extracted and projected onto a phase plane. The computer then analyzes them and automatically determines the best choices of the tested echoes. Finally, statistical analysis is applied by adding additional target waveforms to compare against. This addition greatly improves the accuracy of the decision method. Among the ten groups of tested echoes, 50% are on the first choice and 40% are on the second choice.

Chapter 4

Analysis of Outside Cylindrical Objects

The research in the previous chapter was limited primarily to the analysis of indoor objects. Our goal is to give robots the ability to deal with real world instructions such as “Go down to the big tree and turn left.” Figure 4.1 shows the scanner mounted atop a mobile robotic platform, which allows us to scan out-door objects.

4.1 Trees vs. Poles

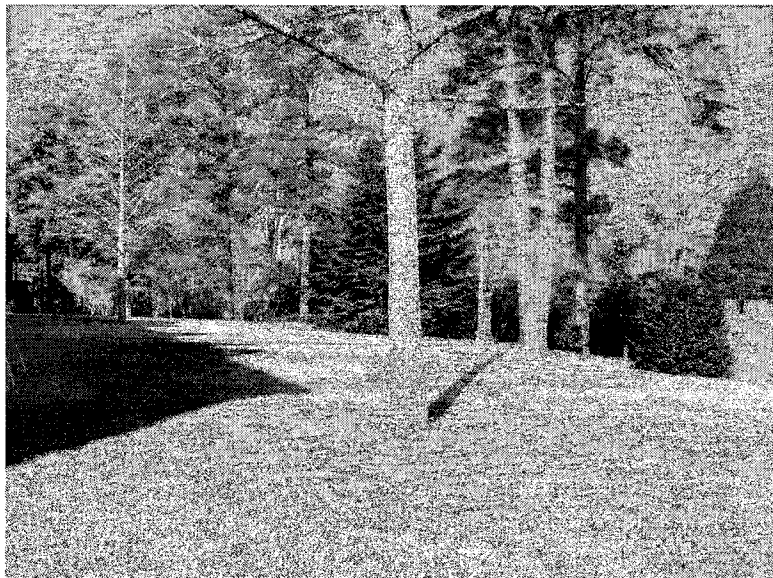
At the frequency range of interest ($\sim 50\text{kHz}$) both the surface features (roughness) and overall shape of objects affect the back-scattered echo. Although the beam-width is too broad to image in the traditional sense, as the beam is swept across a finite object, variations in the beam profile give rise to characteristically different responses as the various scattering centers contribute constructively and destructively. Here we consider two classes of cylindrical objects: trees and smooth circular poles. In this study, we scanned 20 trees and 10 poles, with up to ten different scans of each object recorded for off-line analysis. Figure 4.2 shows an example of a lamppost and a tree scanned during this research.



Figure 4.1 Mobile robotic platform with computer-controlled scanner.



(a)



(b)

Figure 4.2 An example of (a) A lamp post and (b) A tree scanned during this research.

4.2 Overview of Pattern Recognition

From 1960s, Pattern recognition became a rapidly growing research field. Now, it's widely used in all kinds of fields in science and engineering. Not a single theory can describe it because it has different and unique approach for different problems. The most independent theory is classification theory, which classifying patterns based on statistical decision theory and mathematical procedures [90].

Initial pattern recognition research may be traced to the early 1950's when the digital computer became available for information processing. It was used to program computers for automatic decision making, and to develop specialized hardware to read patterns such as printed alphanumeric characters [91].

In the late 1950's, Rosenblatt introduced the Perceptron algorithm which was an early model for information storage and organization in the brain [92]. During this period, major approaches to the pattern recognition problem were based primarily upon statistical decision theory and threshold logic principles.

In the late 1960's and 1970's it was becoming apparent that statistical pattern recognition had its limitations and should not be pushed too hard. Several researchers realized that features were everything. The key task is to find features that not only preserve information but also cluster tightly within a class, while being far apart from other classes. Tou [93] in his paper discussed the selection of mathematical features on

the basis of entropy minimization and introduces the concept of extracting statistical features by the method of kernel approximation.

Structural pattern recognition is a good method to extract features [94]. It relies on segmentation (of images, curves, etc) and the extraction of interesting relationships (possibly syntactic) between the parts found during segmentation. Features are defined either as properties of parts or as properties of relationships between parts. The features found in this way may be used as input to a statistical classifier. However, structural pattern recognition has no theoretical basis. It is based on the common sense observation that problems must be first understood before they can be solved.

In the 1970's, Maroy and Berthod [95] presented a program that understands Natural Languages Commands. They showed that by structural pattern recognition techniques, the program "learns" the vocabulary and the structure of the sentences.

Another type of pattern recognition is syntactic pattern recognition [94]. It actually predates structural pattern recognition. It also describes the input to the pattern recognition system as an aggregate of parts. However, it places major emphasis on the rules of composition. Moayer an Fu [96] presented a syntactic approach to the recognition of fingerprint pattern.

There is a combination of the two methods of pattern recognition. Haralick [97] discussed the general supervised pattern discrimination problem from a structural point of

view. The methodology suggested in his paper provides a structural pattern recognition generalization to phrase-structured syntactic pattern recognition.

Structural pattern recognition has developed progressively during the last two decades. Recently, Ahmed and Ward [98] introduced an expert system for analysis and recognition of general symbols. The system uses the structural pattern recognition technique for modeling symbols by a set of straight lines referred to as segments. Suganthan [99] presented a genetic algorithm based optimization procedure for the solution of structural pattern recognition problem using the attributed relational graph representation and matching technique.

More recently, numerous papers have been published on artificial neural networks. ANNs are widely used in all kinds of pattern recognition problems because of their ease of use and because good ANN learning algorithms already exist. For example, Kulikowski [100] described an extension of the k-NN pattern recognition algorithm on the case of non-metric observation spaces.

Pardo and Sberveglieri [101] in their paper not only reviewed the classical statistical learning scenario but also introduced some modern (ensembles) methods developed inside the machine learning field. Ensembles techniques have drawn considerable attention in recent years. A set of learning machines increased classification accuracy with respect to a single machine.

Among numerous methodologies, template-matching and feature-extraction are the basic techniques in Pattern Recognition arena.

For template-matching, an incoming stimulus is compared to specific templates that have been stored in memory in an attempt to find a good match. When a good match is found, the stimulus is identified based on that match. Template matching has been used extensively in object recognition. McQueen discussed a generalized template procedure for recognizing cells in a microscope slide [102]. G.Ryall and Sandor presented a technique based on goodness-of-fit statistics for classification of binary images via template matching [103]. Bhatnagar et.al. described a template-matching based system for discriminating among a number of ground vehicles using their one-dimensional vertical-profile signatures obtained from airborne sensors [104]. Mostafa et.al. described their target recognition performed by matching the generated templates to a library using a sub-pixel contour matching algorithm[105]. Kermit and Eide reported on a system able to classify different signals containing auditive information based on capture of small signal segments present in specific types of sound [106].

For feature extraction, each type of pattern is stored as a set of features and an associated label. An incoming stimulus is compared to this stored information and is identified according to the best match. Murphy described an approach to the problem of detecting and enhancing linear features in noisy digital images [107]. Diaz etc. presented a new method for corner and circular feature detection in gray-level images, which based on the application of standard statistical techniques to the distribution of gradient

orientations in a circular neighborhood of the prospective feature point [108]. Wang et.al. presented a feature-based face recognition system based on both 3D range data as well as 2D gray-level facial images [109]. Fiala and Basu described an approach to detecting features in panoramic non-SVP (single viewpoint) images using a modified Hough transform [110]. Huang et.al. proposed a classification-based method for locating frontal faces in cluttered images. To improve the detection performance, they extracted gradient direction features from local window images as the input of the underlying two-class classifier [111].

If we consider pattern recognition an engineering problem, it is not surprising that no general methodologies are available. We must understand the “physics” of the problem and select from amongst available tools the ones appropriate for the problem.

4.3 Methodology

4.3.1 Physics

Before further research on identifying outside cylindrical objects outside, the relationship between the back scattered energy and the received signal shown on the oscilloscope $v(t)$ is analyzed.

The acoustic energy is defined as the sum of the kinetic energy and the potential energy.

$$E = E_k + E_p = \varepsilon V = \frac{1}{2} \rho_0 \mu^2 V + \frac{1}{2} \rho_0 \frac{p^2}{\rho_0^2 C_0^2} V = \frac{1}{2} m_0 \mu^2 + \frac{1}{2} \frac{p^2 V^2}{m_0 C_0^2} \quad (4.1)$$

The acoustic density is the acoustic energy per volume.

$$\varepsilon = \frac{E_k + E_p}{V} = \frac{1}{2} \rho_0 \left(\mu^2 + \frac{p^2}{\rho_0^2 C_0^2} \right) \quad (4.2)$$

$$\text{For the plane sin(cos) wave, } \varepsilon = \frac{p^2}{\rho_0 C_0^2}, \quad (4.3)$$

ρ_0 is the density of media, C_0 is the speed of sound in the media, and p is the acoustic pressure. The acoustic power is $\overline{W} = \overline{\varepsilon} C_0 S$, S is the area of the surface that the energy passes through. If it's the area of the surface of the transducer, then this equation describes the acoustic power of the transducer.

The sound pressure p is proportional to the output voltage $v(t)$. Figure 4.3 shows a received total A-Scan echo from an object. It's also the output voltage $v(t)$. The wide

signal on the left-hand side is initial tone burst. The smaller signal on the right is the echo from the object.

Since $\varepsilon = \frac{P^2}{\rho_0 C_0^2} \propto v^2$, it can also be written as $\varepsilon(t) = Av(t)^2$. Integrating on both sides over the interval $[t_1, t_2]$,

$$\int_{t_1}^{t_2} \varepsilon(t) dt = A \int_{t_1}^{t_2} v(t)^2 dt \quad (4.4)$$

$$\text{then, } \int_{t_1}^{t_2} \frac{W(t)}{C_0 S} dt = A \int_{t_1}^{t_2} v(t)^2 dt, \quad (4.5)$$

$$\text{and } \int_{t_1}^{t_2} W(t) dt = C_0 SA \int_{t_1}^{t_2} v(t)^2 dt \quad (4.6)$$

Consequently, in the time period $[t_1, t_2]$, the acoustic energy [115] is

$$\text{Acoustic_Energy} = C_0 SA \int_{t_1}^{t_2} v(t)^2 dt \quad (4.7)$$

In the time period $[t_2, t_3]$, the back scattered acoustic energy is

$$\text{Acoustic_Energy} = C_0 SA \int_{t_2}^{t_3} v(t)^2 dt \quad (4.8)$$

$\int_{t_m}^{t_n} v(t)^2 dt$ is defined as the signal energy in time period $[t_m, t_n]$ of the output voltage

signal. It is proportional to the back-scattered acoustic energy. Here instead of signal energy, the sum of the range line of the echo is calculated for analysis, which we define as backscatter.

4.3.2 Signal Pre-processing

A-scan echo pre-processing is necessary before analyzing the backscatter plots. Pre-processing is done to extract the echo part as shown in Figure 4.4 (a). The second step is to remove the very low frequency component of the extracted A-Scan echo to keep its horizontal central line straight. It increases the signal-to noise ratio. Figure 4.4 (a) shows the echo from the block acquired with the Gage card. Figure 4.4 (b) is the Fast Fourier Transform of the signal. The large low frequency component is due to the floating of the Gage card. Therefore it's important to remove it to make the central line straight and decrease the noise. The absolute value of the echo without the low frequency filtering is displayed in Figure 4.4 (c). It has a lot of noise. The absolute value of the echo with the low frequency filtering is shown in Figure 4.4 (d). It preserves the echo and at the same time reduces the noise. The backscatter is obtained as the sum of the absolute value of the echo within the window after the low frequency filtering.

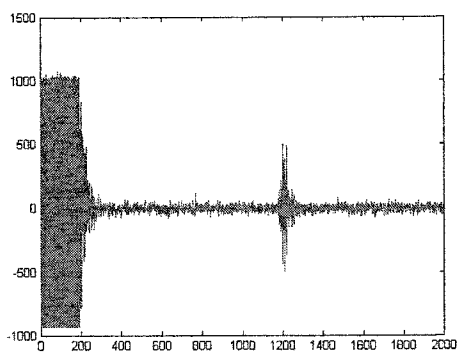


Figure 4.3 A typical A-Scan echo.

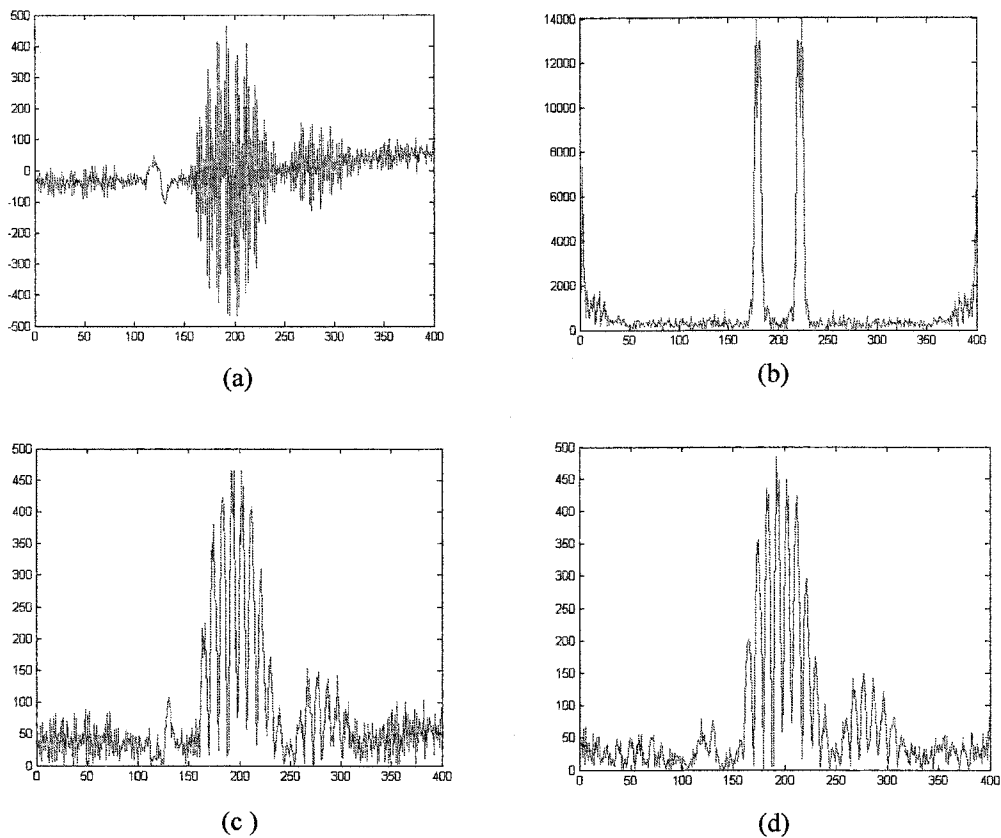


Figure 4.4 (a) An echo from block (b) Fast Fourier Transform of the echo (c) Absolute value of the echo without the low frequency filtering (d) Absolute value of the echo with the low frequency filtering.

4.3.3 Analysis and Results

All the data were acquired at 50kHz via the mobile apparatus. The beam was swept across each object for a range of elevation angles and the RF echoes corresponding to the horizontal fan were digitized and recorded for off-line analysis. For each angle in the horizontal sweep we calculate the backscatter of the echo by low-pass filtering, rectifying, and integrating over the window corresponding to the echo from the object. For the smooth circular metal poles we find, as expected, that the backscatter energy is symmetric about a central maximum where the incident beam axis is normal to the surface. Trees tend to have a more complicated response due to non-circular cross sections and/or surface roughness of the bark. Rough bark can give enhanced backscatter for grazing angles where the smooth poles give very little response. We plot the backscatter vs. scan angle and fit a 5th order polynomial to it. Smooth circular poles always give a symmetric (bell-shaped) central response whereas rough and/or irregular objects often give responses less symmetric about the central peak. In general, one sweep over an object is not enough to tell a tree from a pole. We need a series of scans for each object to be able to robustly classify them. This is equivalent to a robot scanning an object repeatedly as it approaches. Assuming that the robot has already adjusted its path to avoid the obstacles, each subsequent scan gives a somewhat different orientation of the target. Multiple looks at the target increase the robustness of our scheme for distinguishing trees from poles because trees have more variations with changing look angle than do round metal poles. The surface characteristics and diameter measurements of 20 trees and 10 metal poles on campus are given in Table 4.1.

Table 4.1 Surface characteristics + diameter of the two kinds of cylindrical objects —trees and poles.

<i>Tree Group 1</i>	<i>Diameter(cm)</i>	<i>Tree Group 2</i>	<i>Diameter(cm)</i>	<i>Poles</i>	<i>Diameter(cm)</i>
<i>(rough trees)</i>		<i>(smooth trees)</i>			
No.1	35	No.5	22	No.1	14
No.2	32	No.6	22	No.2	14
No.3	50	No.7	15	No.3	29
No.4	10	No.9	15	No.4	11
No.8	15	No.10	15	No.5	10
No.11	25	No.14	19	No.6	11
No.12	24	No.15	10	No.7	11
No.13	12	No.16	32	No.8	12
No.18	30	No.17	23	No.9	14
No.19	30			No.10	24
No.20	60				

Figure 4.5 shows the backscatter plots of pole No.1 (a 14 cm diameter circular metal lamppost) from different distances and their 5th order polynomial interpolations. Each data point was obtained by low-pass filtering, rectifying, and integrating over the window corresponding to the echo from the object to calculate the backscatter. For each object, 16 data points were calculated as the beam was swept across it. The polynomial fits to these data points are shown by the solid curve for 9 different scans. All of the fits in Figure 4.5 are symmetric (bell-shaped) near the central scan angle, which is characteristic of a smooth circular pole.

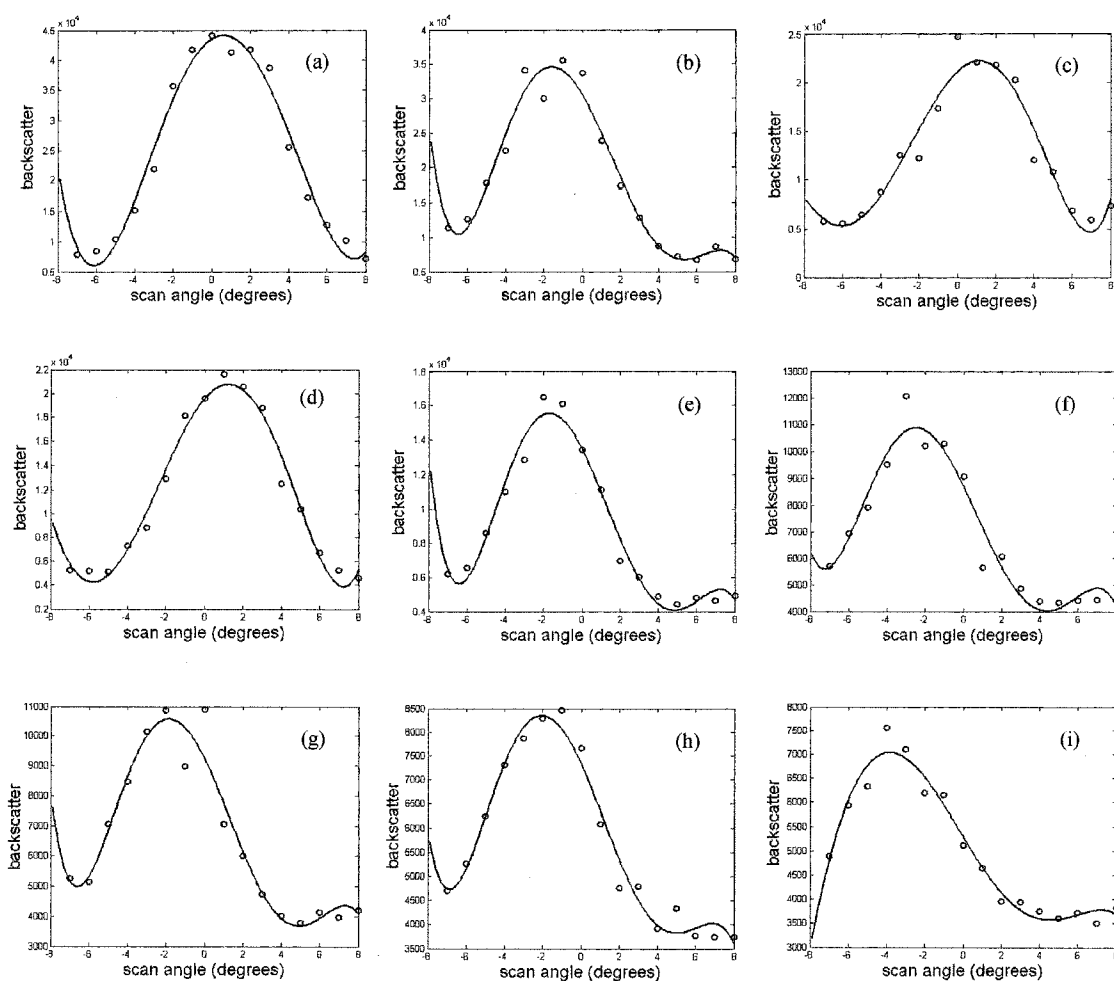


Figure 4.5 Backscatter plots of pole No.1 when the sensor is (a) 100cm (b) 125cm (c) 150cm (d) 175cm (e) 200cm (f) 225cm (g) 250cm (h) 275cm (i) 300cm from the pole.

Figure 4.6 shows the backscatter plots of tree No.14, a 19 cm diameter tree which has a relatively smooth surface. Nine scans are shown from different distances along with their 5th order polynomial interpolations. Some of these are symmetric (bell-shaped) and some are not, which is characteristic of a round smooth-barked tree.

Figure 4.7 shows the backscatter plots of tree No.18, which is a 30 cm diameter tree with a rough bark surface. Nine scans are shown, from different distances, along with their 5th order polynomial interpolations. Only a few of the rough-bark scans are symmetric (bell-shaped) while most are not, which is characteristic of a rough and/or non-circular tree. We also did the same procedure for trees No.15-17, 19-20 and poles No.2-3, 9-10. We find that if all the plots are symmetric bell-shaped it can be confidently identified as a smooth circular pole. If some are symmetric bell-shaped while some are not, it can be identified as a tree.

Of course our goal is to have the computer distinguish trees from poles automatically based on the shapes of backscatter plots. The feature vector \mathbf{x} we choose contains two elements: asymmetry and deviation. If we let x_1 represent asymmetry and x_2 represent deviation, the feature vector can be written as $\mathbf{x}=[x_1, x_2]$. For example, Figure 4.8 is the backscatter plot of pole No.1 when the distance is 200cm. For x_1 , we use Full-Width Half Maximum (FWHM) to define asymmetry. We cut the full width half-maximum into two to get the left width L_1 and right width L_2 . The asymmetry is defined as the difference between L_1 and L_2 divided by FWHM, which is $|L_1-L_2|/|L_1+L_2|$.

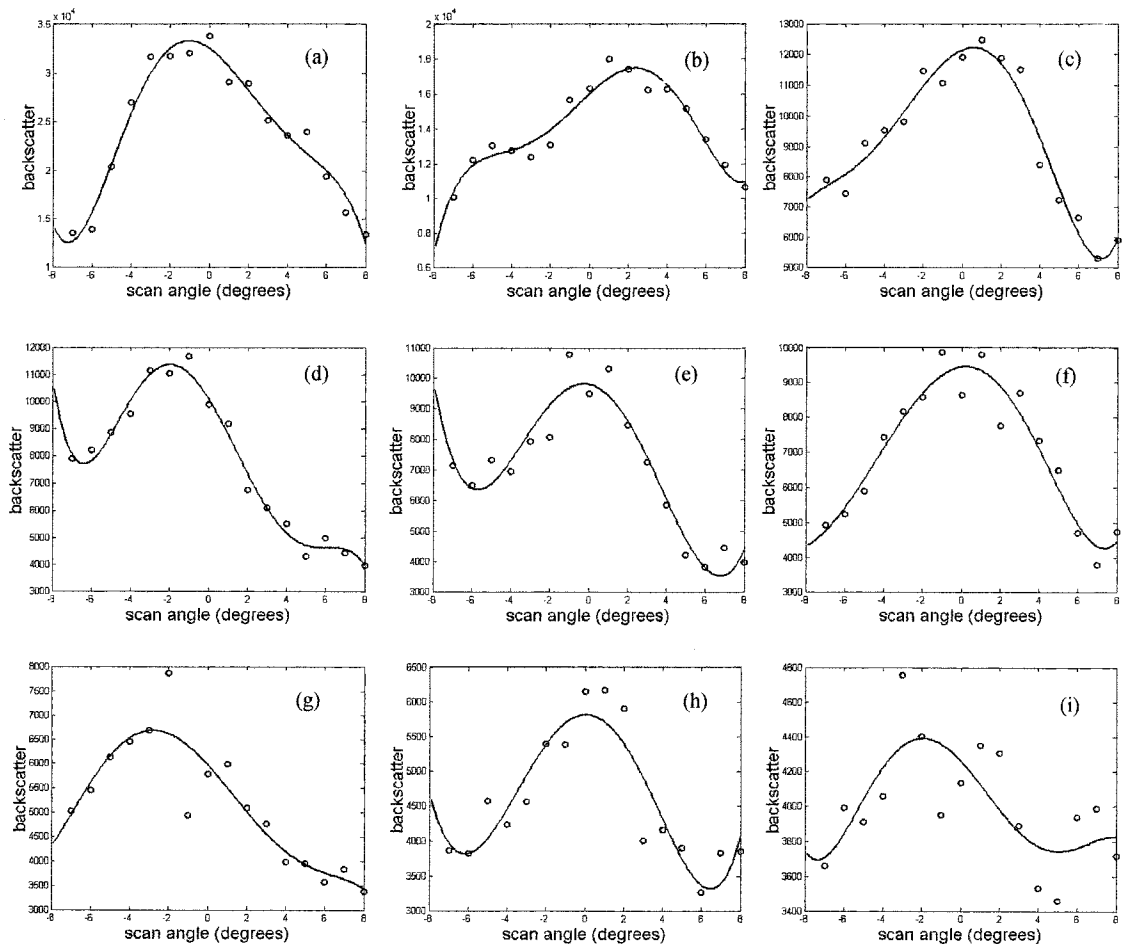


Figure 4.6 Backscatter plots of tree No.14 when the sensor is (a) 75cm (b) 100cm (c) 125cm (d) 150cm (e) 175cm (f) 200cm (g) 225cm (h) 250cm (i) 275cm from the tree.

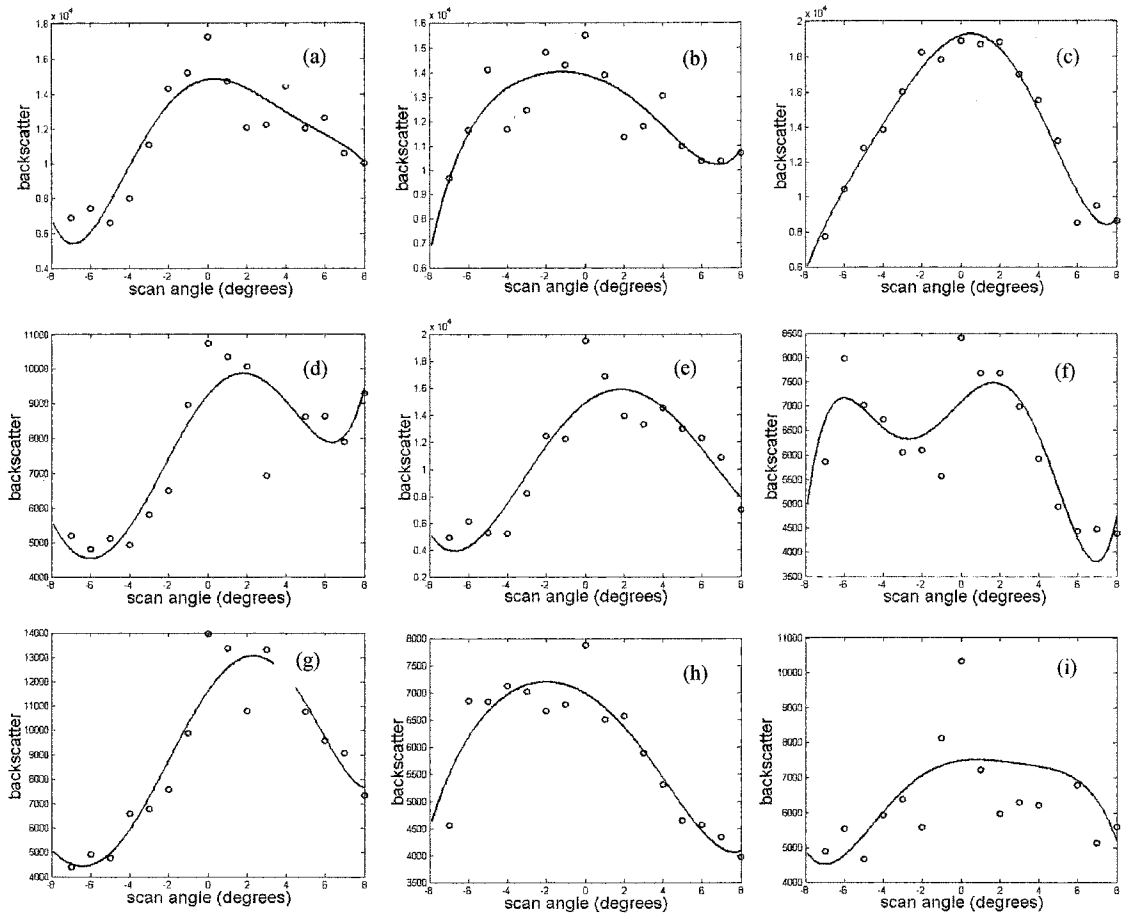


Figure 4.7 Backscatter plots of tree No.18 when the sensor is (a) 100cm (b) 120cm (c) 140cm (d) 150cm (e) 160cm (f) 170cm (g) 180cm (h) 190cm (i) 200cm from the tree.

For the deviation x_2 , we define it as the average Euclidean distance from the experimental data point to the fitted data point at the same scan angle, divided by the total height of the fitted curve H . In this case, there are 16 experimental data points, so

$$Dev = (|d1| + |d2| + \dots + |d16|) / (HX16).$$

For the plot in Figure 4.8, *asymmetry* is 0.0333, which means the degree of asymmetry is small. The *deviation* is 0.0467, which means the degree of deviation is also small.

For the backscatter plot of tree No.14 in Figure 4.9, its asymmetry will be bigger than the more bell-shaped plots. Trees No.1-4, 8,11-13 have rough surfaces (tree group No.1) while trees No.5-7, 9-10 have smooth surfaces (tree group No.2). The pole group contains poles No.4-8. Each tree has two sweeps of scans while each pole has four sweeps of scans.

We plot the Asymmetry-Deviation phase plane in Figure 4.10. Circles represent the pole group while dots represent tree group No.1 and stars represent tree group No.2. We find circles usually within $[0, 0.2]$ in the Asymmetry area. Dots representing the rough surface trees (tree group No.1) spread widely from minimum Asymmetry 0 to maximum 1. Stars representing the smooth surface trees (tree group No.2) are also within $[0,0.2]$ of the Asymmetry area. Two scans per tree may be good enough to tell a rough tree from a pole, but not to distinguish a smooth tree from a pole.

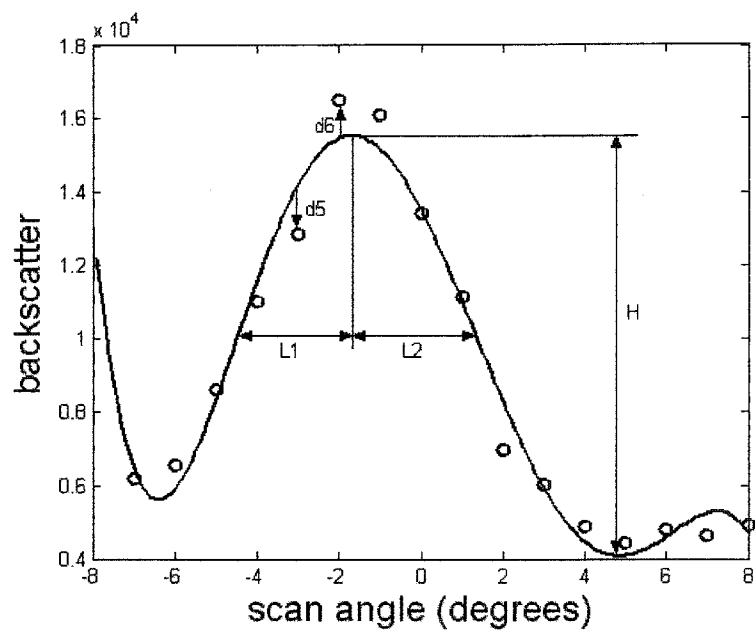


Figure 4.8 Backscatter plot of Pole No.1 at a distance of 200cm.

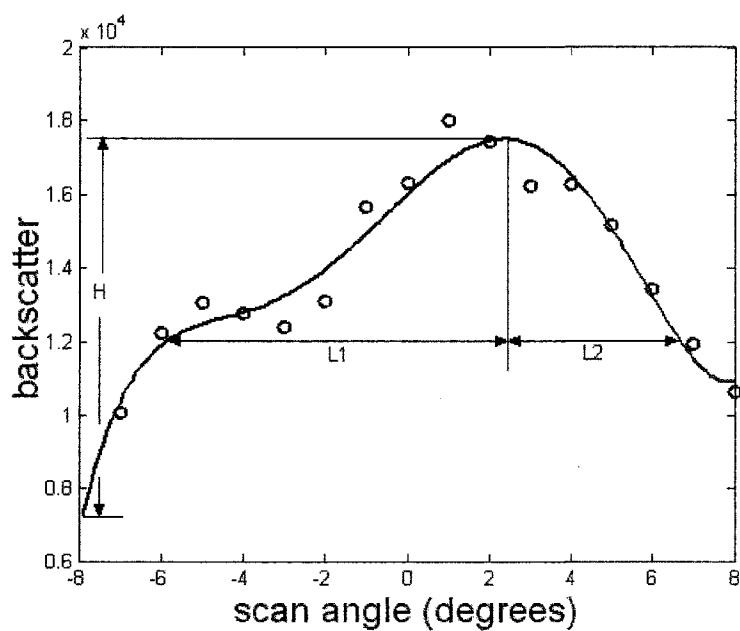


Figure 4.9 Backscatter plot of tree No.14 at a distance of 100cm.

We next acquire a series of scans (≥ 9) from different locations relative to the object, find the backscatter plots from the data, extract the asymmetry and deviation features from each sweep of backscatter plots and then plot them in the phase plane. If all of the data points are located within a small asymmetry area, we say it's a smooth circular pole. If some of the results are located in the small asymmetry area, some are located in the large asymmetry area, we can say it's a tree. If all the data points are located in the large asymmetry area, we say it's a tree with rough surface. The boundary between small asymmetry area and large asymmetry area is set to be 0.2.

Figure 4.11 is the Asymmetry-Deviation phase plane of Pole No.1 and Tree No.14 with the individual scan results shown. Circles "o" mean Pole No.1. Stars "*" mean Tree No.14. We find, as to the circle group, both its asymmetry feature and its deviation feature lie within a small area near the origin. For the star group, both its asymmetry feature and its deviation feature lie in a big area. So the star group means a tree while the circle group means a pole.

Our purpose is to classify the unknown cylindrical objects by the location of their feature vectors in the phase plane. We want to find some boundary to separate the tree group and pole group. Therefore, further feature extraction is needed. First, for a series of points of one object in the Asymmetry- Deviation graph, we calculate the average point of the series of points and find the average squared Euclidean distance from the points to the average point. In Figure 4.12, stars mean the Asymmetry-Deviation feature

of the series of scans for tree No.14. The triangle means the average value of these points. The vector d_1, d_2 etc. are the Euclidean distances of each point to the average point.

We then calculate the average squared Euclidean distance from the points to the average point. It is $(d_1^2 + d_2^2 + \dots + d_n^2)/n$ (n is the total number of scans, here in this case, $n=10$). We also extract the Asymmetry value of the average data point and call it Average Asymmetry. These two features are then combined into a new feature vector and plotted it into the Average Asymmetry-Average Squared Euclidean Distance phase plane. Tree No.14 is then represented by a single point. Applying the same procedure to trees No.15-20, and poles No.1-3, 9-10, we obtain the following phase plane of these objects, shown in Figure 4.13. Star means the trees and circle means poles. We find the pole group clusters in the small area near the origin $(0, 0)$, while the tree group spread widely away from the origin.

Hence, in the Average Asymmetry-Average Squared Euclidean Distance phase plane, if an object's feature vector is located in the small area near the origin, which is within $[0, 0.1]$ in Average Asymmetry and within $[0, 0.02]$ in Average Squared Euclidean Distance, we can say it's a pole. If it is located in the area away from the origin, which is beyond the set area, it's classified as a tree.

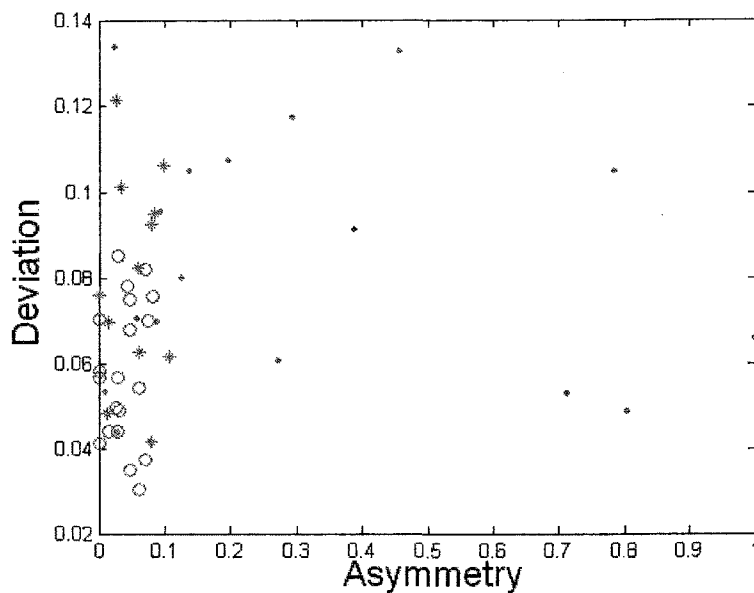


Figure 4.10 Asymmetry-Deviation phase plane of the pole group and two tree groups.

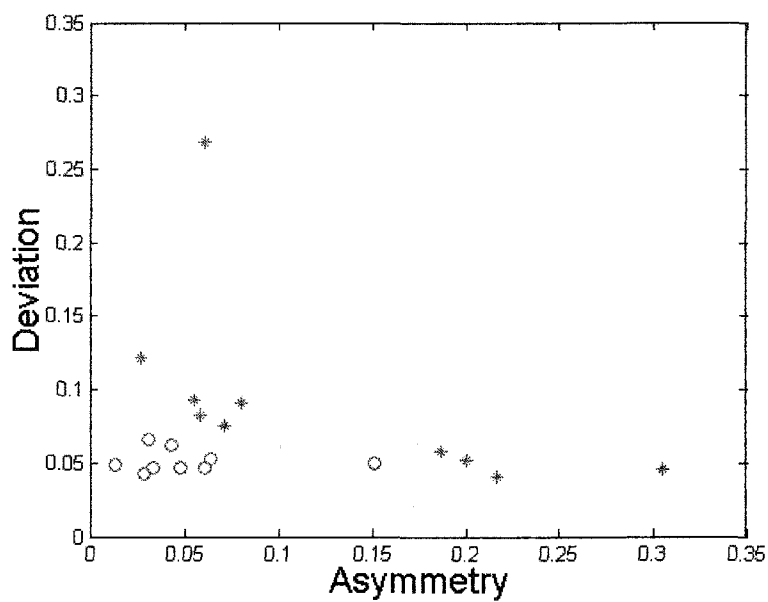


Figure 4.11 Asymmetry-Deviation phase plane of pole No.1 and tree No.14.

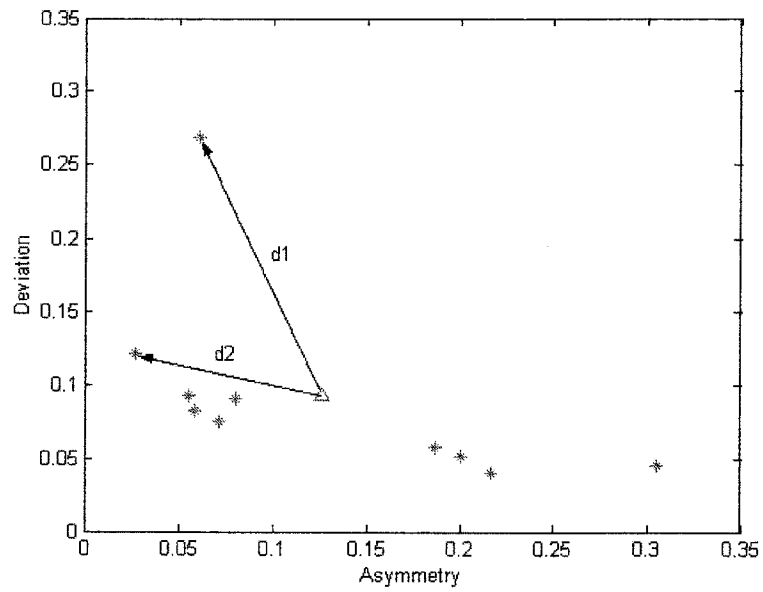


Figure 4.12 Asymmetry-Deviation phase plane for tree No.14

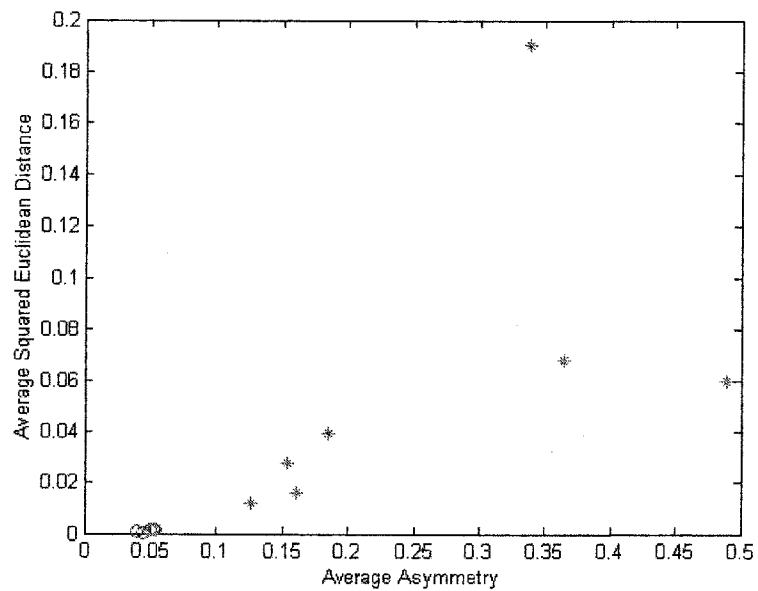


Figure 4.13 Average Asymmetry-Average Squared Euclidean Distance phase plane of trees No.14-20 and poles No.1-3, 9-10.

4.4 Conclusion

In this chapter, we present an algorithm to tell trees from smooth circular poles based on analysis of backscattered sonar echoes. The echo data are acquired by a mobile robot which has a 3D air-coupled ultrasound scanning system packaged as sensor head. Four major steps are conducted. First, a series of scans (≥ 9) for one object are done by the mobile sonar system. Second, the backscatter vs. scan angle plots are constructed by a 5th order polynomial fit. Then, Asymmetry and Deviation features are extracted from the interpolation plots. Finally, further feature extractions are conducted based on the Asymmetry-Deviation graph. Average Asymmetry-Average Squared Euclidean Distance phase plane is segmented to tell a tree from a pole by the location of the data points for the objects interested. Results located in the small area near the origin represent poles while those located away from the origin represent trees.

Chapter 5

Analysis of Extended Objects Outside

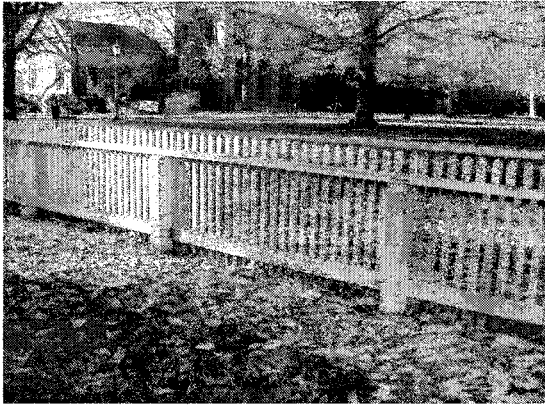
In this chapter, our goal is to distinguish seven extended objects commonly found on the campus — picket fence (inside part and outside part), metal fence with brick pillar, flat brick wall, hedge, brick wall with thin buttress, and brick wall with thick buttress. Figure 5.1 (a)-(f) shows each of these objects. They look similar and are hard to distinguish in low visibility conditions such as night time.

This chapter describes how we scan the objects, analyze the data and develop algorithms [112]-[125] to distinguish the objects. Four major steps are conducted. Step I is object scanning and data analysis. Step II is deformable template matching. Step III is Feature Extraction and Step IV is objects categorization.

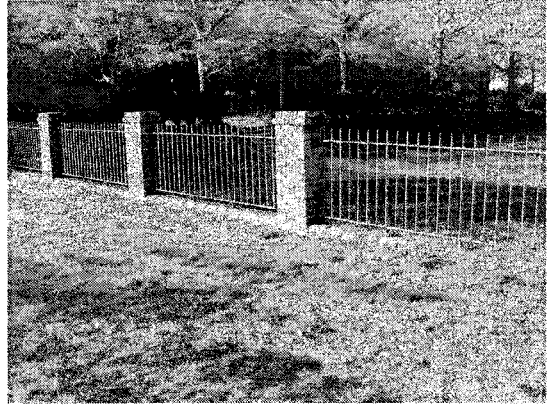
5.1 Object Scanning and Data Analysis

5.1.1 Object Scanning

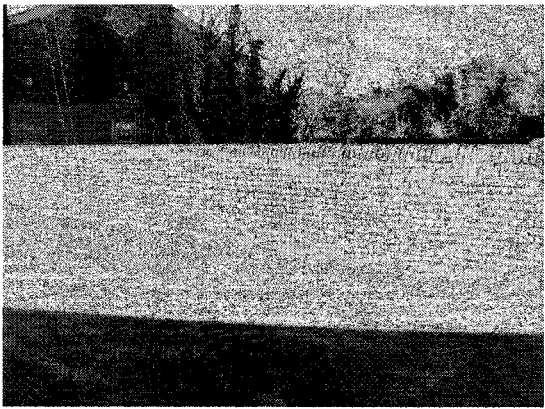
Since all the objects are extended, it's not realistic to scan each part of them. Instead, we defined one unit for each object and had the robot conduct five scans with equal distances which typically cover a spatial unit. For a picket fence, the unit begins at one fence post and ends at the next post. For the metal fence with brick pillar, it begins at one brick pillar and ends at the next. For a brick wall with thin buttress and the brick



(a)



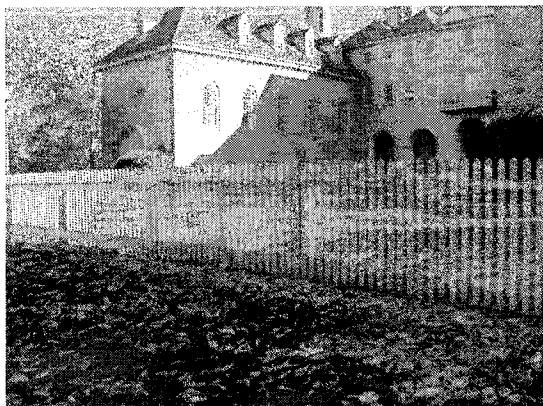
(b)



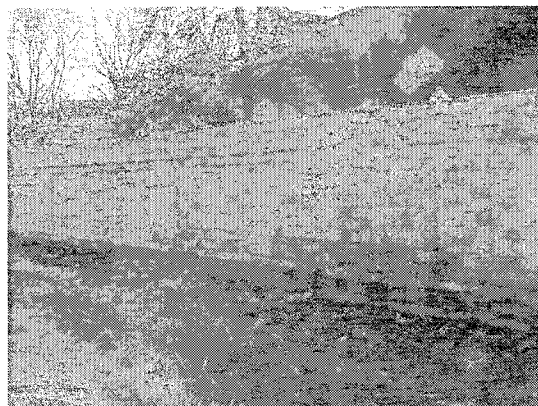
(c)



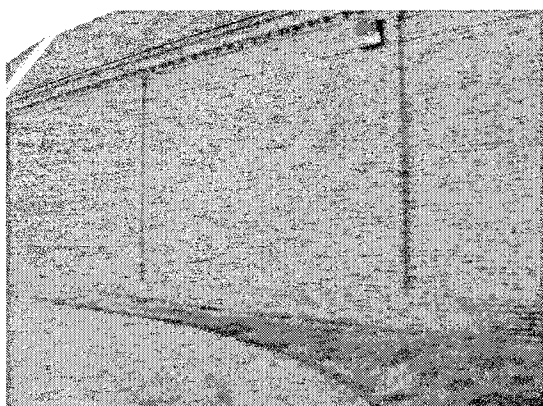
(d)



(e)



(f)



(g)

Figure 5.1 (a) Picket fence scanned from inside (b) Metal fence with brick pillar (c) Flat brick wall (d) Hedge (e) Picket fence scanned from outside (f) Brick wall with thin buttress (g) Brick wall with thick buttress.

wall with thick buttress, the unit begins at one buttress, ends at the next. For those without such periodic features, such as flat brick wall and hedge, we scanned any part of them around the same size of one unit of other objects under investigation.

We took five scans with the robot facing five different locations of the unit from left to right at a distance of $1m$. For example, Figure 5.2 describes how the robot scanned the picket fence from inside. The first scan was taken with the robot facing the left post. The second was obtained with the robot facing the part between the left post and the center of the unit. The third scan data was acquired with the robot facing the center part of the unit. The fourth was obtained with the robot facing the part between the center of the unit and the right post. The fifth scan data was taken with the robot facing the right post.

5.1.2 Data Analysis

Figure 5.4 shows the five corresponding backscatter plots of a unit of the picket fence scanned from inside. Each data point was obtained by low-pass filtering, rectifying, and integrating over the window corresponding to the echo from the object to calculate the backscatter. In our experiments, some data sets obtained previously have 96 horizontal scan steps. Other data sets acquired later have 100 horizontal scan steps. In Figure 5.4, for each location, 96 data points were calculated as the beam was swept across it. The 96 data points are from the 96 horizontal scan steps. Each step represents 1° of scan angle.

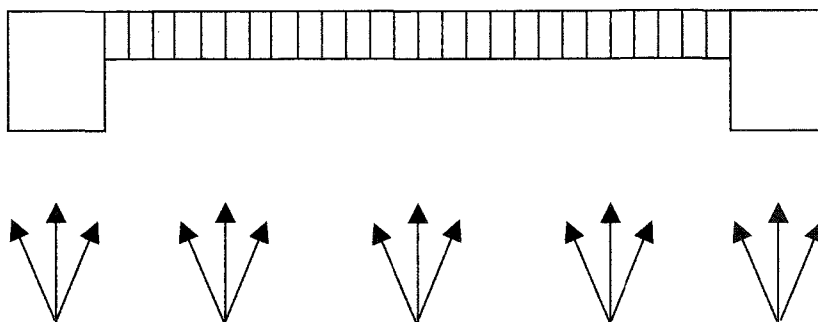


Figure 5.2 Five scanning locations for one unit of the fence.
The two blocks at the ends are posts for the picket fence.

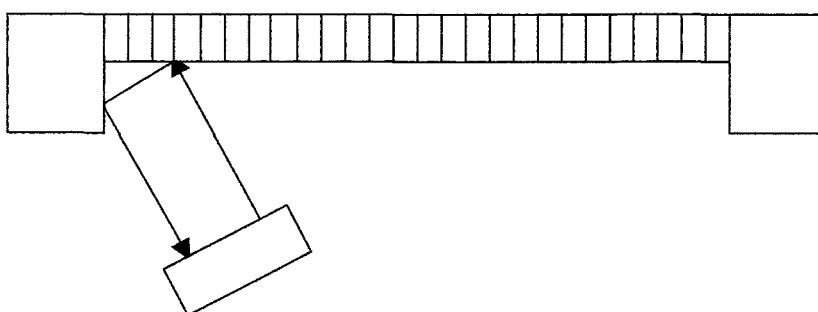


Figure 5.3 The retro-reflection phenomenon happens at the second scan.

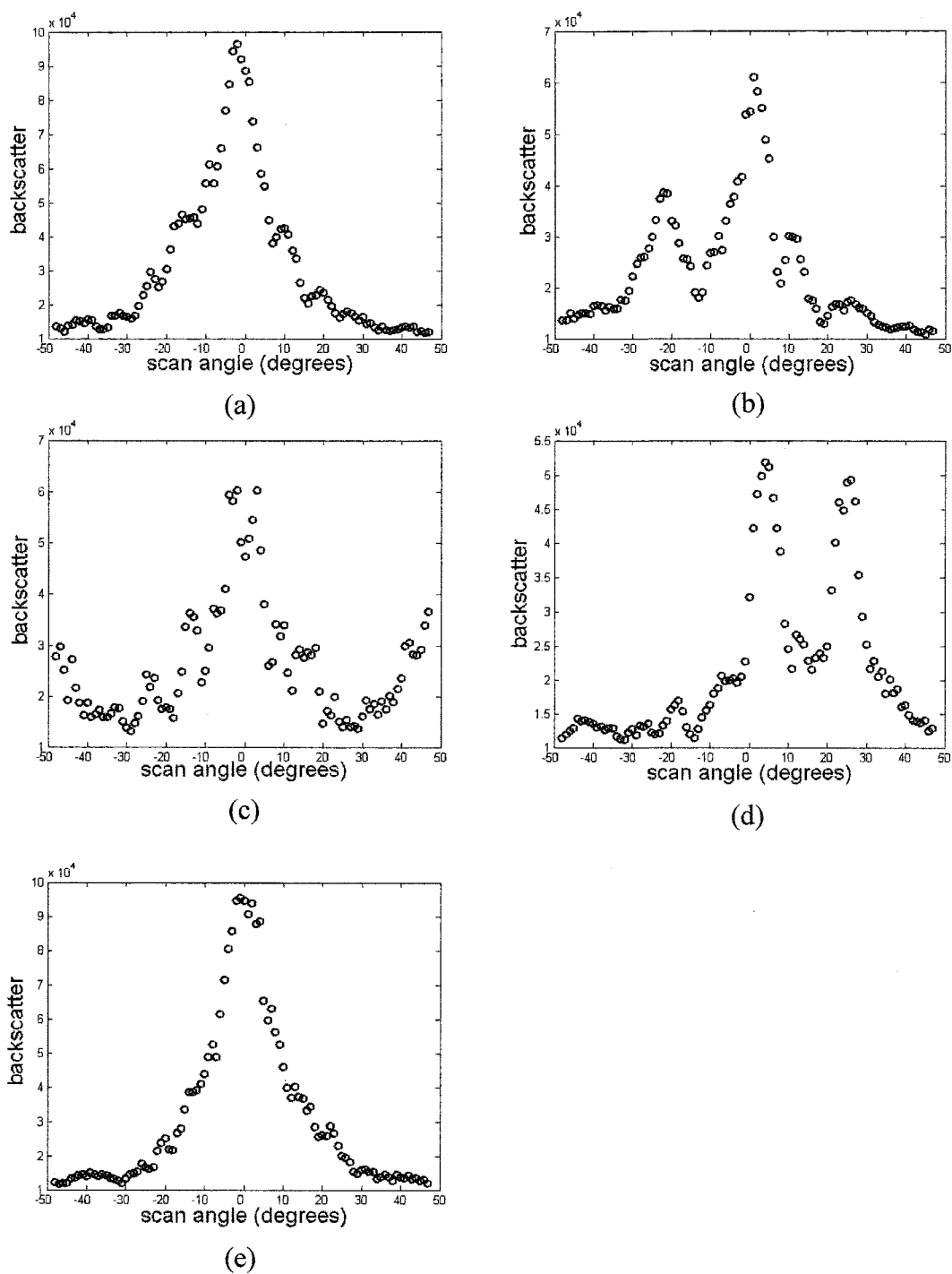


Figure 5.4 Backscatter plots of a unit of the picket fence scanned from inside with the robot facing five different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan.

For those data sets with 100 scan steps, 100 data points were calculated as the beam was swept across the object. Each step represents 1° of scan angle.

Figure 5.4(a) indicates a central peak, obtained with the robot facing the post. Figure 5.4(b) reveals not only a central peak but also a side peak on its right hand side. The central peak is from the fence while the side peak is from a retro-reflector, which is the right angle made by the side surface of the post and the adjacent fence.

As demonstrated in Figure 5.3, when the sound wave interacts with a retro-reflector, their original direction is changed by 180 degrees and then return back to the transducer. The transducer can receive strong echoes from a retro-reflector regardless of the angle of incidence. Consequently, a side peak shows up when the transducer is facing the corner.

Figure 5.4(c) also displays a significant central peak. However, its shape is a little different from the first data set. Here, when the scan angle deviates from the central, the backscatter increases. It indicates a retro-reflection is coming soon. Actually, when the scan angle is deviating from the central, the sensor is facing the retro-reflector —the corner made by the side surface of the post and the fence nearby.

Figure 5.4(d) reveals a central peak and a side peak on its left hand side. It can be explained the same way as the second data set. Figure 5.4(e) displays a significant central peak because it's obtained with the robot facing the second fence post.

Figure 5.5 are the five backscatter plots of a unit of the metal fence with brick pillars with the robot facing five different locations of the unit from left to right at a distance of *100cm*. Figure 5.5(a), (e) display a significant central peak respectively. They are obtained with the robot facing the brick pillars of the metal fence. Figure 5.5(b), (c) and (d) indicate no apparent peaks. They are obtained with the robot facing the metal fence part.

Figure 5.6 displays the backscatter plots of a unit of flat brick wall with the robot facing five different locations of the unit from left to right at a distance of *100cm*. Each plot indicates a significant central peak, which is characteristic of a flat brick wall. Figure 5.7 shows the backscatter plots of a hedge with the robot scanning at the distance of *100cm*. There are no prominent sharp peaks in the plots for a hedge.

Figure 5.8 demonstrates the five backscatter plots of a unit of the picket fence scanned from outside with the robot scanning from left to right at a distance of *100cm*. Figure 5.8 (a) (c) (e) indicate a prominent central peak respectively. Figure (b) reveals a central peak and a side peak on its right hand side. Figure (d) displays a central peak and a side peak on its left hand side.

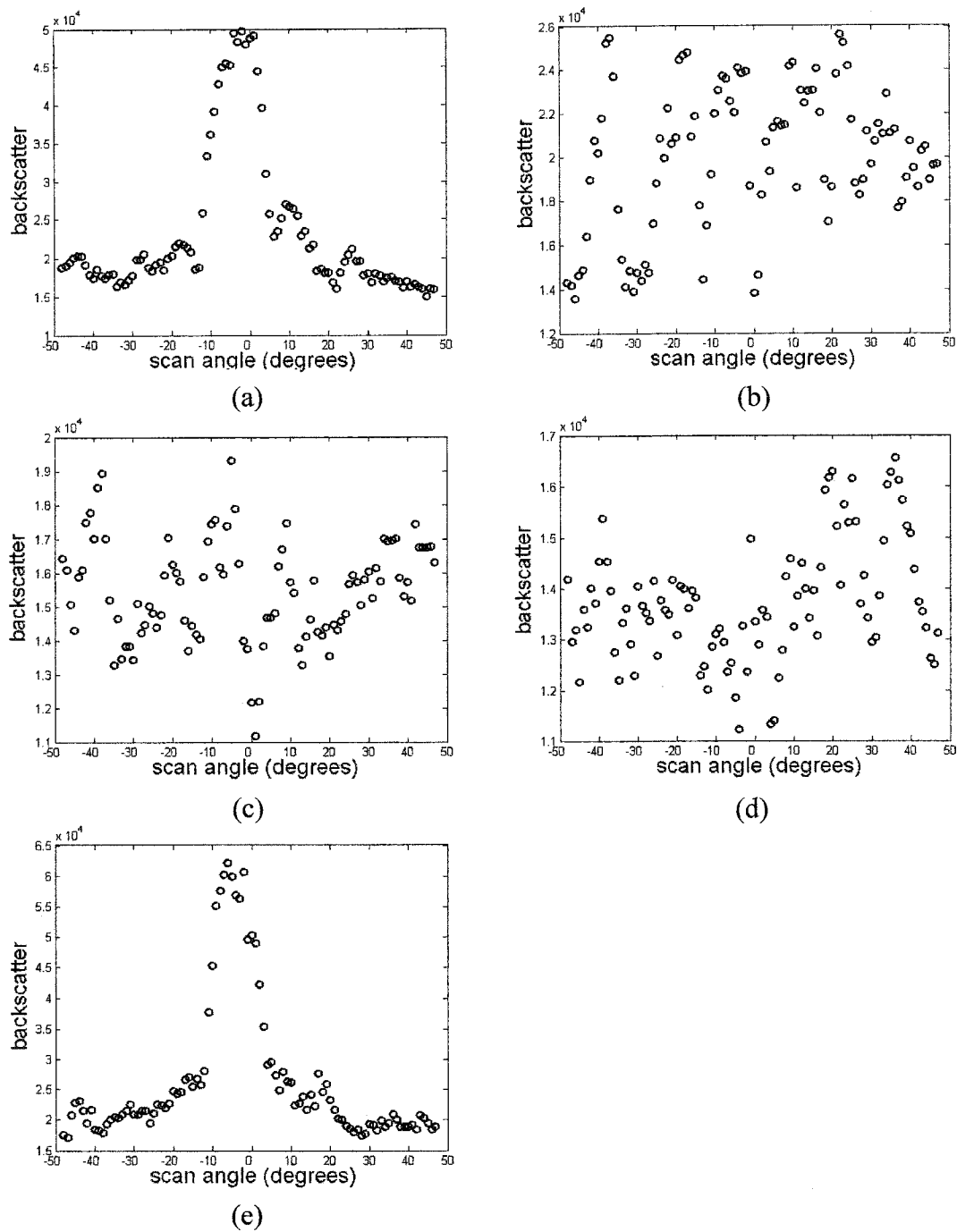


Figure 5.5 Backscatter plots of a unit of the metal fence with brick pillar with the robot facing different locations from left to right at a distance of 100cm.(a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan.

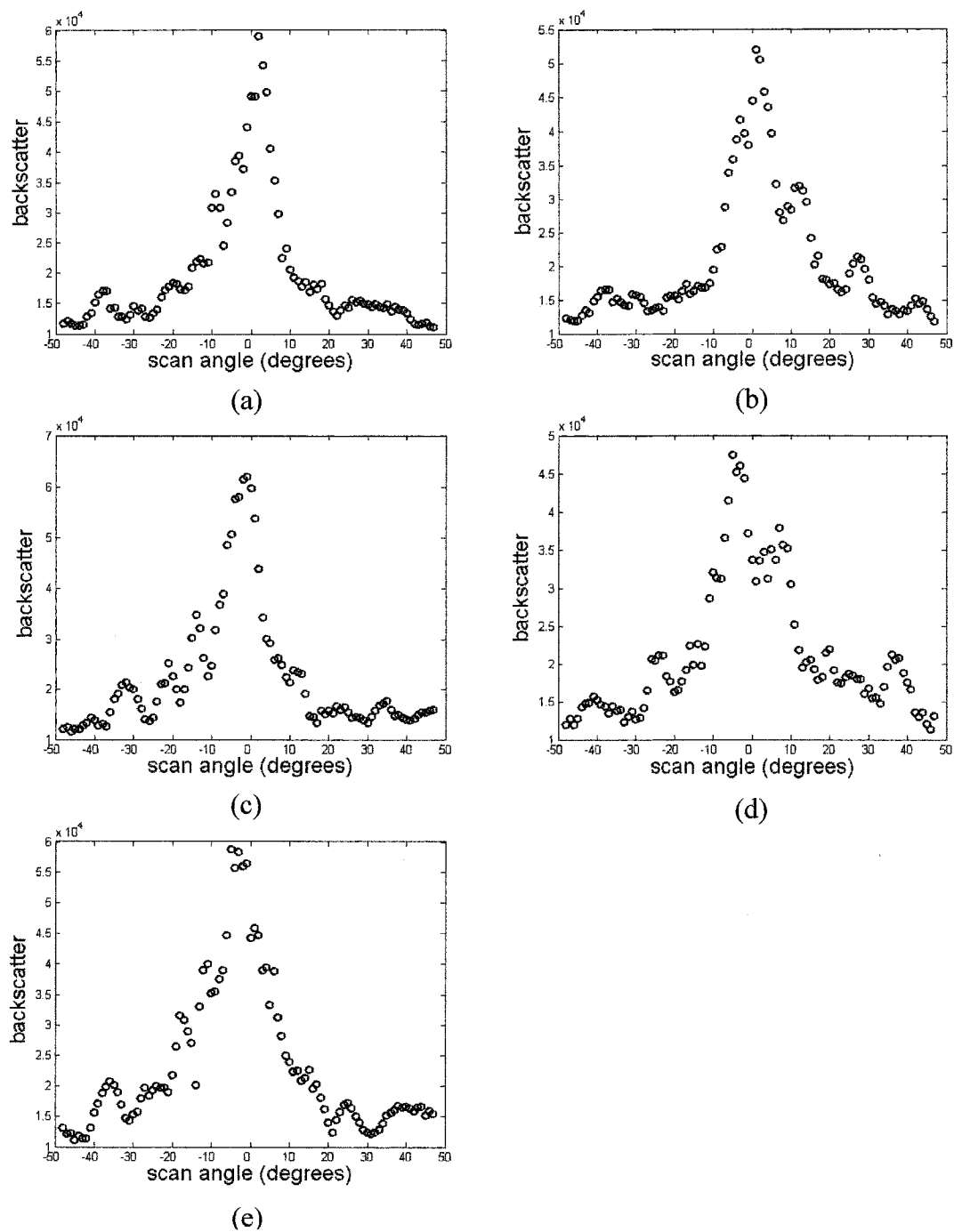


Figure 5.6 Backscatter plots of a unit of flat brick wall with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan.

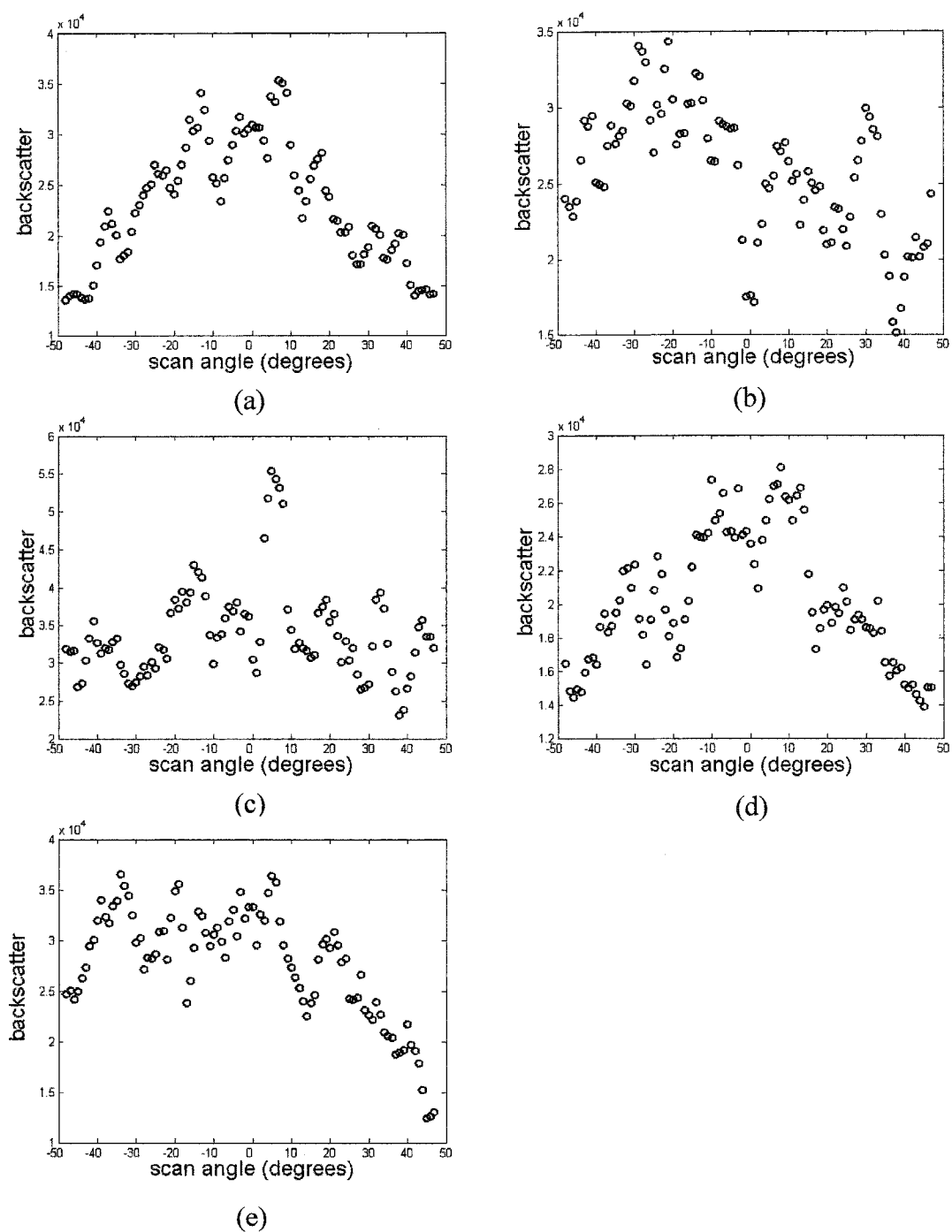


Figure 5.7 Backscatter plots of a unit of hedge with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan.

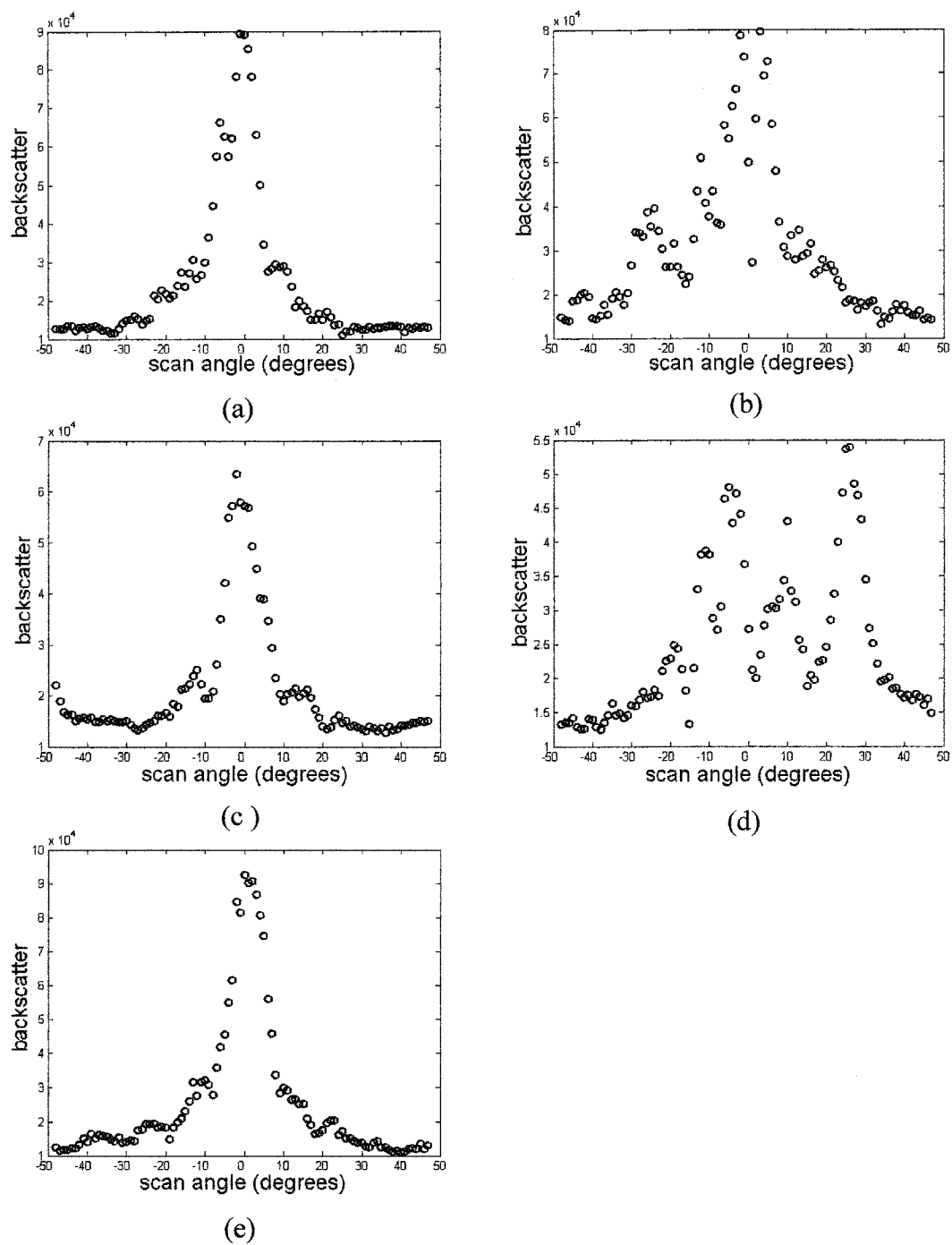


Figure 5.8 Backscatter plots of a unit of picket fence scanned from outside with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan

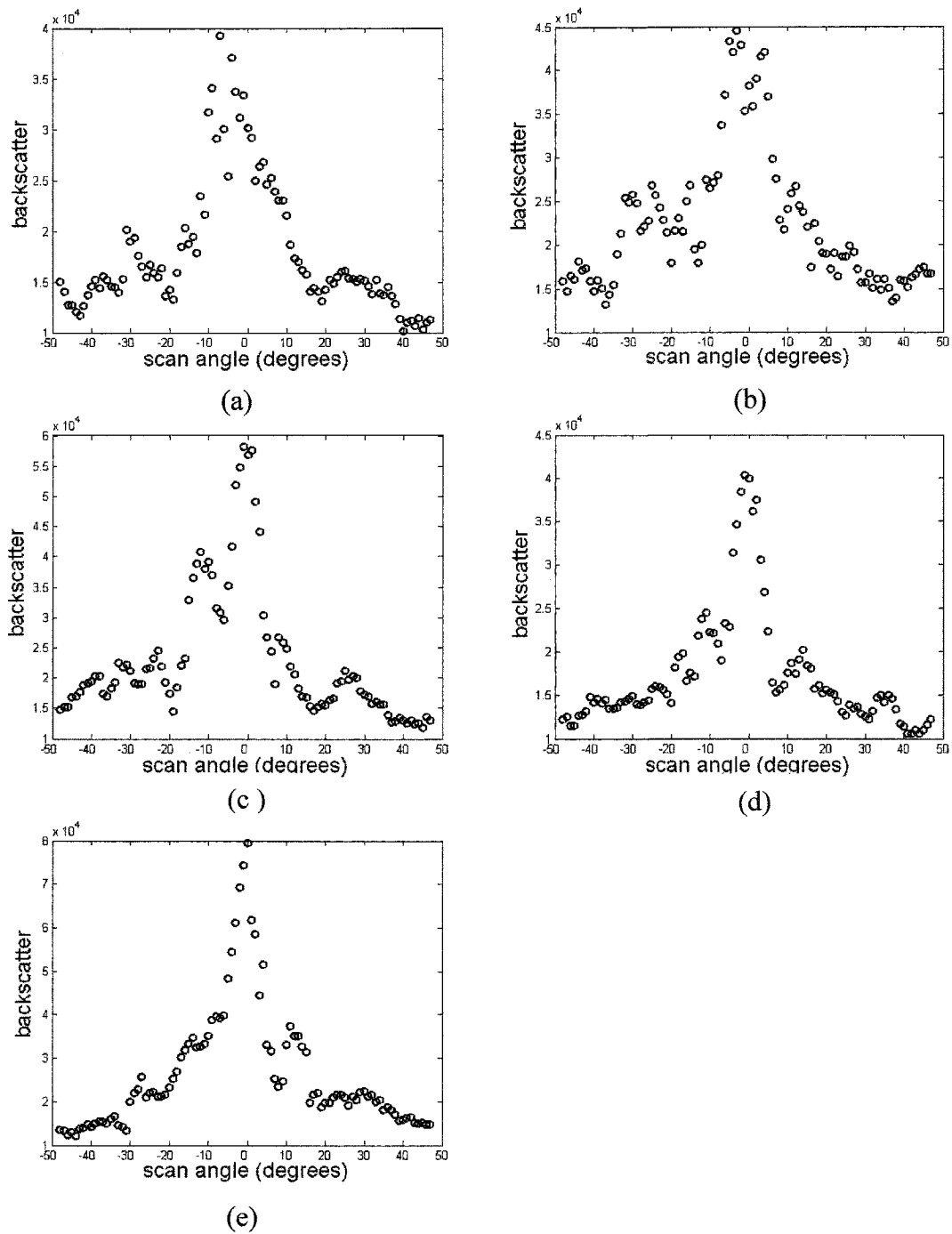


Figure 5.9 Backscatter plots of a unit of brick wall with thin buttress with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan

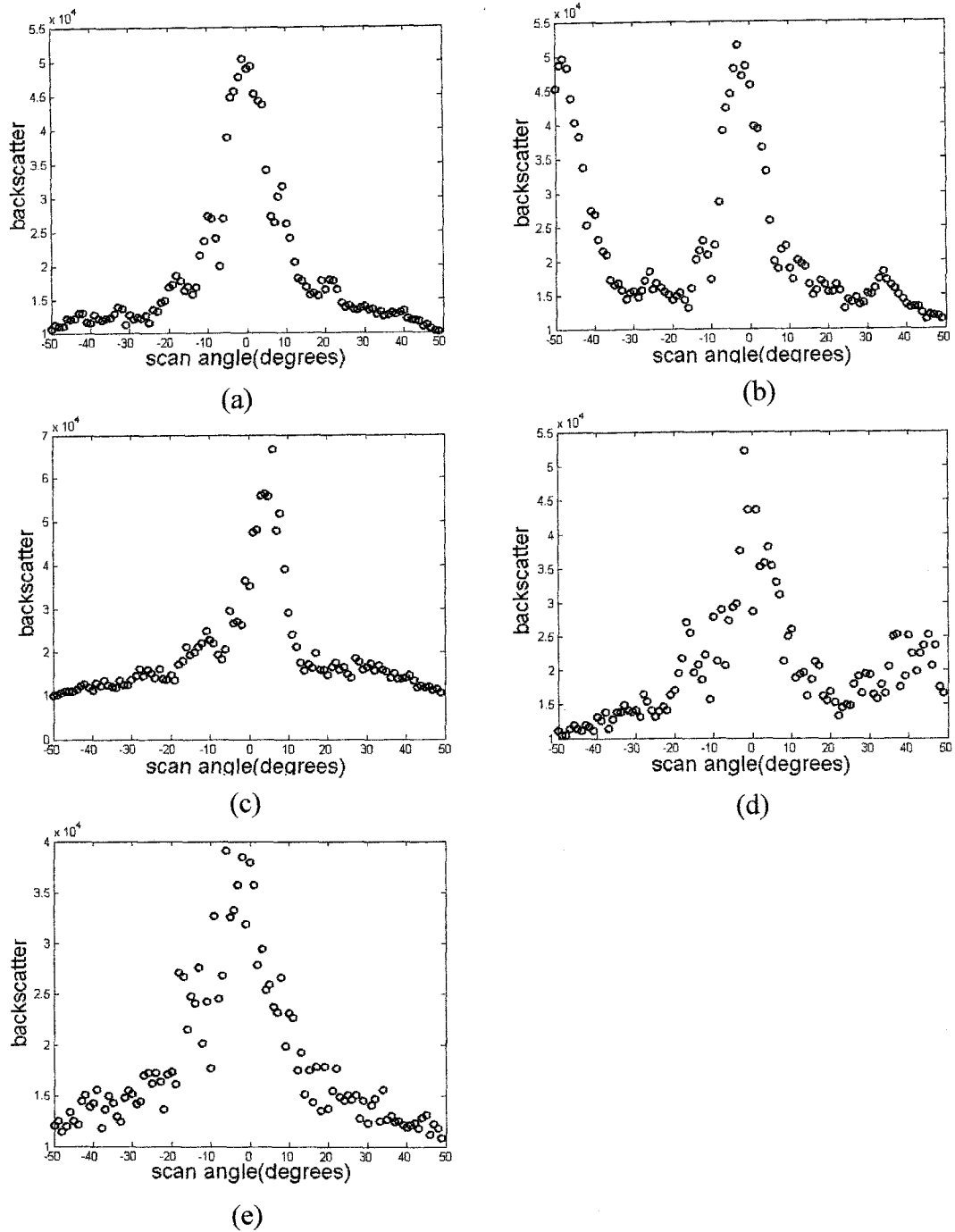


Figure 5.10 Backscatter plots of a unit of brick wall with thick buttress with the robot facing different locations from left to right at a distance of 100cm. (a) the first scan (b) the second scan (c) the third scan (d) the fourth scan (e) the fifth scan.

Figure 5.9 are the backscatter plots of a unit of a brick wall with thin buttress with the robot facing its different locations from left to right at a distance of 100cm. Each of the plots displays a prominent central peak. Figure 5.10 displays the backscatter plots of a unit of a brick wall with thick buttress with the robot scanning it from left to right at a distance of 100cm. Figure 5.10 (a) (c) (e) indicate a significant central peak respectively. Figure 5.10(b) reveals a prominent central peak and a significant side peak. Figure 5.10 (d) displays a prominent central peak and a small side peak.

5.2 Deformable Template Matching

The backscatter plots have their own patterns. Each plot has one, two, or no significant peaks. A series of deformable N -peak templates ($N=1, 2, 3 \dots \dots N_{max}$) are introduced to describe their patterns. N_{max} is the maximum number of the major local peaks in the backscatter plot.

A deformable template is a template that can fit itself with the data of interest without losing its general characteristics. For example, for a one-peak deformable template, its peak location may change when fitting to different data. However, it always preserves its one peak shape characteristic. In the following part, we will describe how to create a series of deformable N -peak templates ($N=1, 2, 3 \dots \dots N_{max}$) for a backscatter plot. It includes how to pick the peak locations and form the shapes of peaks.

For a particular backscatter plot, we need to determine the apices of all its major local peaks. They will be used later as the apices of peaks of the N -peak-template ($N=1, 2, 3 \dots \dots N_{max}$). For example, in Figure 5.11, the backscatter plot is from a picket fence. We first extract the local maxima shown as stars in Figure 5.11. Those local maxima have the largest backscatter value comparing to their immediate neighbors. Therefore, data points larger than both its left and right immediate neighbors are extracted as local maximum points. We then do the same procedure to the local maximum points. That is, again, to extract the local maxima from the local maximum points plot. Those second-step local maxima are the major apices of the backscatter plot. They are shown as squares in Figure 5.11.

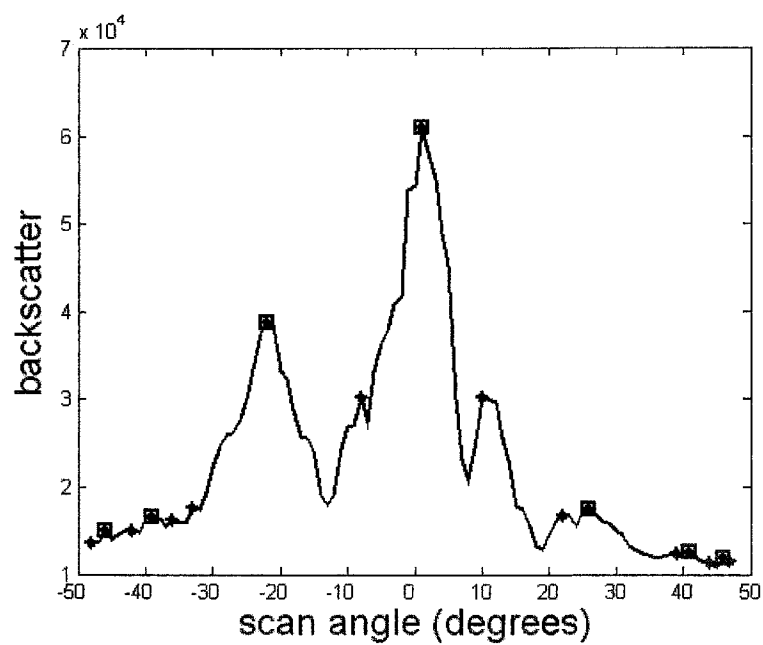


Figure 5.11 Backscatter plot for a picket fence and all its local and major apices, stars '*' mark all the local apices and squares '□' mark all the major apices.

In this case, we have seven major local peaks. The apex with the highest value of backscatter is selected as the apex of the one-peak template. Apices with the highest and second highest values of backscatter are selected as the two apices of the double-peak template. The apices with the highest, second highest and third highest values of backscatter are selected as the three apices of the triple-peak template. The similar procedure is conducted until all the apices are selected as the N_{max} apices of the N_{max} -peak template.

After the apices of the N -peak templates have been selected, the shapes of the templates need to be determined. The total horizontal scan step (or the total scan view angle) of a particular data set is x_T ($x_T=100$ or 96 according to the data set).

For the one-peak template, the selected apex divides the one-peak template into two parts. The left part of the peak template is actually the right half part of the quintic function $y = c_1(x - x_S)^5 + S(x_S)$, which passes through the apex point $(x_p, S(x_p))$. We define $S(n)$ as the value of backscatter when scan angle value is n . x_S is the start scan angle value. For example, in Figure 5.11, the total scan view angle is 96° . The start scan angle value x_S is -47° . x_p is the scan angle value of the apex point and $S(x_p)$ is the

backscatter value of the apex point. C_1 can then be calculated as $c_1 = \frac{S(x_p) - S(x_S)}{(x_p - x_S)^5}$.

Therefore, the one-peak template function on the left part

$$\text{is } y = \frac{S(x_p) - S(x_s)}{(x_p - x_s)^5} (x - x_s)^5 + S(x_s).$$

The right part of the one peak template is actually the left half part of the quintic function $y = c_2(x - x_E)^5 + S(x_E)$, Which pass through the apex point $(x_p, S(x_p))$. x_E is the end of scan angle value. For example, in Figure 5.11, it is $+48^\circ$. C_2 can be calculated as

$$c_2 = \frac{S(x_p) - S(x_E)}{(x_p - x_E)^5}. \text{ Therefore, the one-peak template function of the right part}$$

$$\text{is } y = \frac{S(x_p) - S(x_E)}{(x_p - x_E)^5} (x - x_E)^5 + S(x_E). \text{ Figure 5.12(a) shows the backscatter plot for a}$$

picket fence and its one-peak template.

For the double-peak template, the two selected apexes $(x_{p1}, S(x_{p1}))$, $(x_{p2}, S(x_{p2}))$ and the point with the minimum value of backscatter within the area between the two apexes $(x_b, S(x_b))$ divides the double-peak template into four regions. p means *peak* and b stands for *bottom*. We set $x_{p1} < x_{p2}$, therefore, it's always true that $x_{p1} < x_b < x_{p2}$.

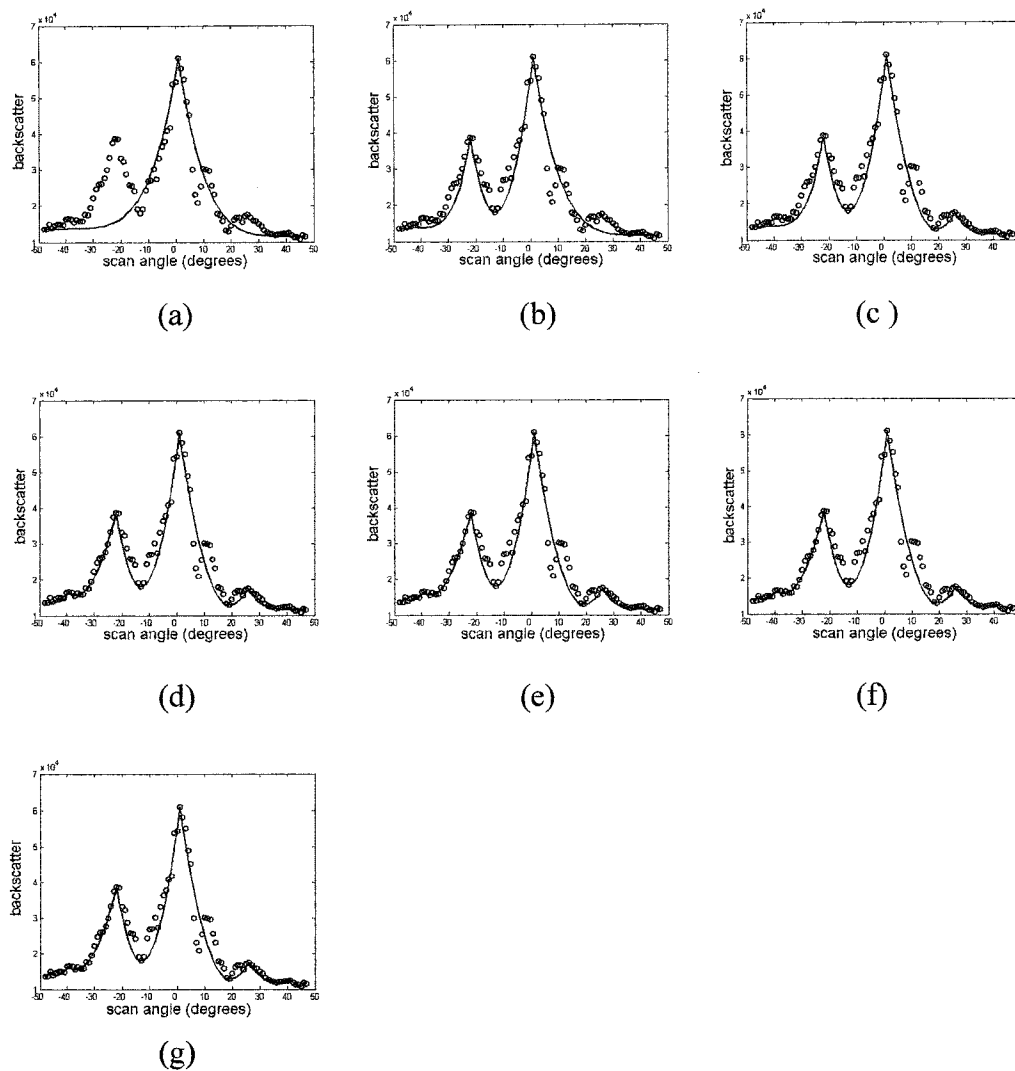


Figure 5.12 Backscatter plots for a picket fence and its n -peak templates. (a) Backscatter plot and its one-peak template. (b) Backscatter plot and its two-peak template. (c) Backscatter plot and its triple-peak template. (d) Backscatter plot and its four-peak template. (e) Backscatter plot and its five-peak template. (f) Backscatter plot and its six-peak template. (g) Backscatter plot and its seven-peak template.

The double peak template in the total area is

$$y = \begin{cases} \frac{S(x_{p1}) - S(x_S)}{(x_{p1} - x_S)^5} (x - x_S)^5 + S(x_S) & x_S \leq x \leq x_{p1} \\ \frac{S(x_{p1}) - S(x_b)}{(x_{p1} - x_b)^2} (x - x_b)^2 + S(x_b) & x_{p1} \leq x \leq x_b \\ \frac{S(x_{p2}) - S(x_b)}{(x_{p2} - x_b)^2} (x - x_b)^2 + S(x_b) & x_b \leq x \leq x_{p2} \\ \frac{S(x_{p2}) - S(x_E)}{(x_{p2} - x_E)^5} (x - x_E)^5 + S(x_E) & x_{p2} \leq x \leq x_E \end{cases} \quad (5.1)$$

In the two middle regions, shapes of quadratic functions are more similar to the backscatter plots. Therefore, quadratic functions are chosen to form the template instead of quintic functions to make better fit. Figure 5.12 (b) shows the backscatter plot for a picket fence and its double-peak template.

For the triple-peak template, the three selected apexes $(x_{p1}, S(x_{p1}))$, $(x_{p2}, S(x_{p2}))$, $(x_{p3}, S(x_{p3}))$ and two bottom points in between $(x_{b1}, S(x_{b1}))$, $(x_{b2}, S(x_{b2}))$ divides the triple-peak template into six regions. We set $x_{p1} < x_{p2} < x_{p3}$, therefore, it's always true that

$$x_{p1} < x_{b1} < x_{p2} < x_{b2} < x_{p3}.$$

The triple-peak template in the total area is

$$y = \begin{cases} \frac{S(x_{p1}) - S(x_S)}{(x_{p1} - x_S)^5} (x - x_S)^5 + S(x_S) & x_S \leq x \leq x_{p1} \\ \frac{S(x_{p1}) - S(x_{b1})}{(x_{p1} - x_{b1})^2} (x - x_{b1})^2 + S(x_{b1}) & x_{p1} \leq x \leq x_{b1} \\ \frac{S(x_{p2}) - S(x_{b1})}{(x_{p2} - x_{b1})^2} (x - x_{b1})^2 + S(x_{b1}) & x_{b1} \leq x \leq x_{p2} \\ \frac{S(x_{p2}) - S(x_{b2})}{(x_{p2} - x_{b2})^2} (x - x_{b2})^2 + S(x_{b2}) & x_{p2} \leq x \leq x_{b2} \\ \frac{S(x_{p3}) - S(x_{b2})}{(x_{p3} - x_{b2})^2} (x - x_{b2})^2 + S(x_{b2}) & x_{b2} \leq x \leq x_{p3} \\ \frac{S(x_{p3}) - S(x_E)}{(x_{p3} - x_E)^5} (x - x_E)^5 + S(x_E) & x_{p3} \leq x \leq x_E \end{cases} \quad (5.2)$$

Figure 5.12 (c) shows the backscatter plot for a picket fence and its triple-peak template.

The four, five, N_{max} -peak template building follows the same procedure. Figure 5.12 (d)-(g) show the backscatter plot for a picket fence and its four, five, six, seven templates respectively. Here, N_{max} is seven.

5.3 Feature Extraction

5.3.1 First Step of Feature Extraction— Deviation vs. N (Template)

Plots

The next step is feature extraction. *Deviation 1* is extracted to describe the similarity of the backscatter plot and its one-peak template. It's defined as the average Euclidean distance from the backscatter data to the template at the same scan angle divided by the

total height of the backscatter plots as shown in Figure 5.13(a). $Dev_{-1} = \frac{1}{H \times N} \sum_{n=1}^N |dn|$,

where N is the total view angle. The difference value dn is the Euclidean distance from the experimental data to the fitted data at the same view angle. H is the total height of the plot.

Deviation 2 is extracted to describe the similarity of the backscatter plot and its two-peak template. It's defined as the average Euclidean distance from the backscatter data to the template at the same scan angle divided by the total height of the backscatter plots as

can be seen in Figure 5.13(b). $Dev_{-2} = \frac{1}{H \times N} \sum_{n=1}^N |dn|$, where N is the total view angle.

The difference value dn is the Euclidean distance from the experimental data to the fitted data at the same view angle. H is the total height of the plots.

Deviation 3, 4, 5, 6, 7 calculation follows the same procedure. They are extracted to describe the similarity of the backscatter plots and its three, four, five, six, seven-peak template respectively. The *Deviation vs. n* (template) plot and its 4th order polynomial

fitting are then displayed by Figure 5.14. It carries critical information about the shape of the original backscatter plot.

There are five scans for each object. Each scan corresponds to one Deviation vs. N plot. These five Deviation vs. N plots are then drawn in a single graph, fitted with 4th order polynomials as shown in Figure 5.15 (a). Circles 'o' represent the first scan. Symbols 'x' represent the second scan. Plus '+' represent the third scan. Stars '*' represent the fourth scan. Squares '□' represent the fifth scan.

Figure 5.15 (a)-(g) display the typical five *Deviation vs. N (Template)* plots for a picket fence scanned from inside, a metal fence with brick pillar, a flat brick wall, a hedge, a picket fence scanned from outside, a brick wall with thin buttress and a brick wall with thick buttress respectively. It demonstrates that different kinds of extended objects have different groups of Deviation vs. N (Template) plots. Therefore, it is possible to tell one object from another from their Deviation vs. N (Template) plots.

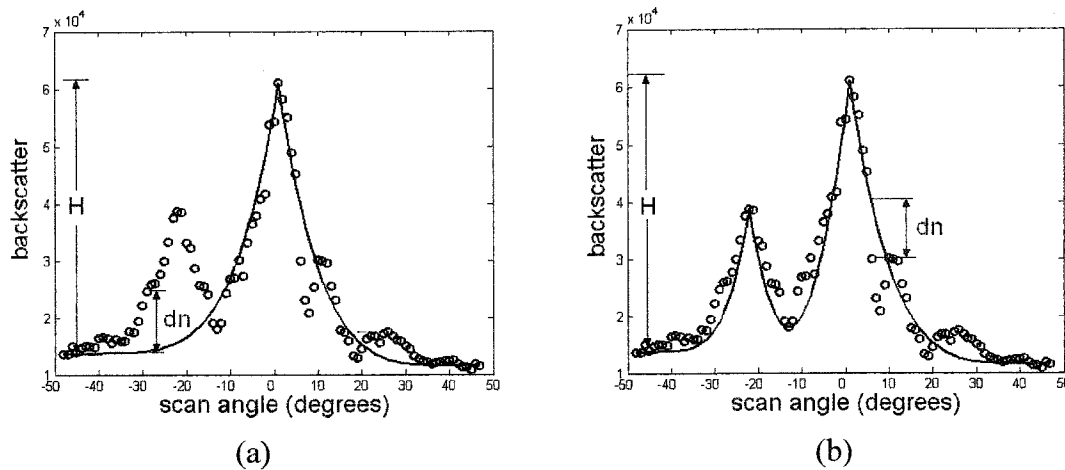


Figure 5.13 Backscatter plots of the second scan of a picket fence and its (a) one-peak template (b) two-peak template

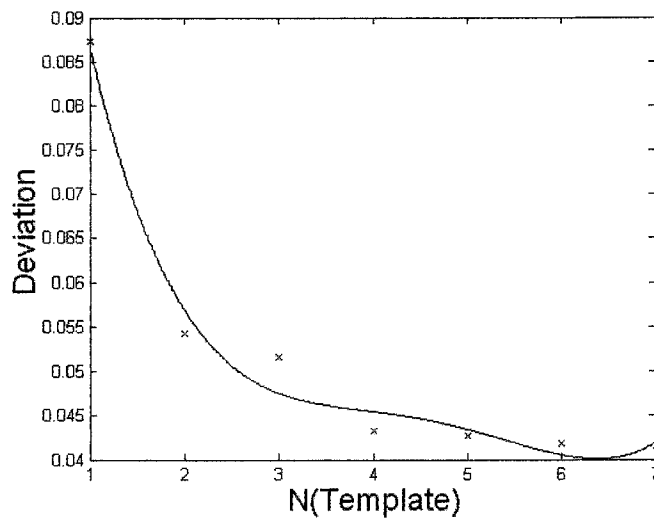
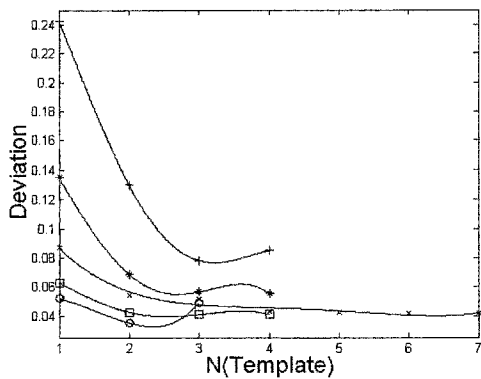
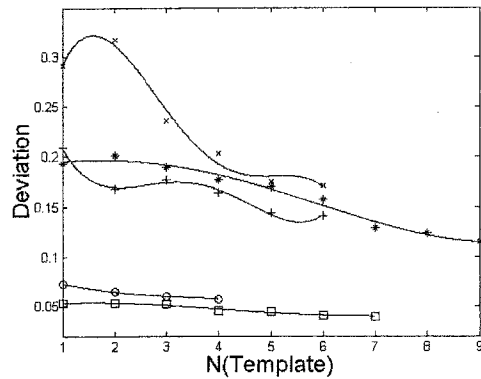


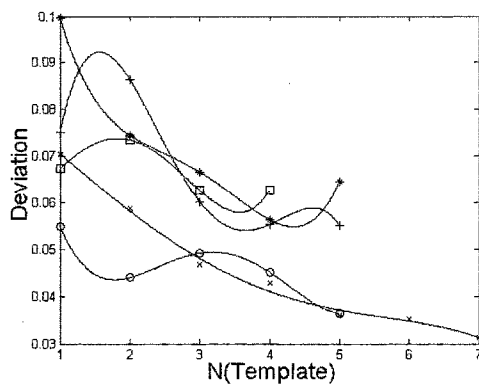
Figure 5.14 Deviation vs. N (Template) plot of the second scan for the picket fence scanned from inside.



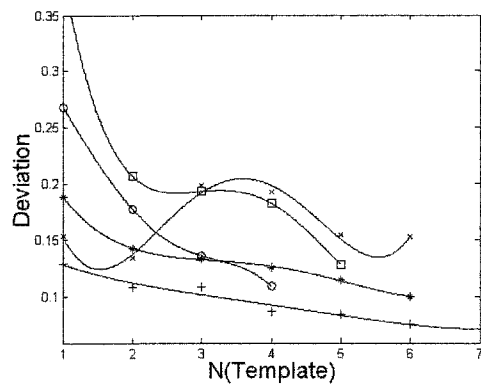
(a)



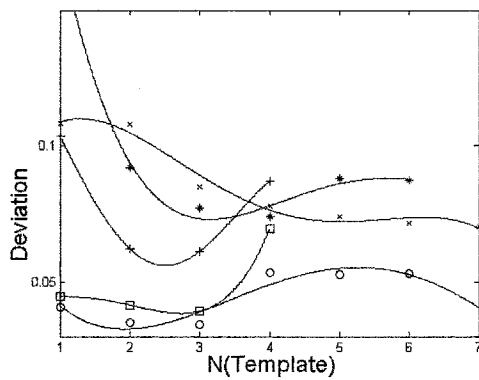
(b)



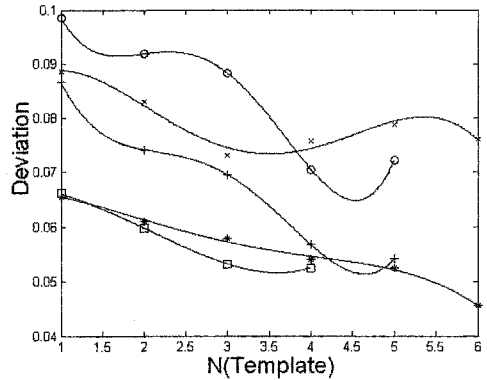
(c)



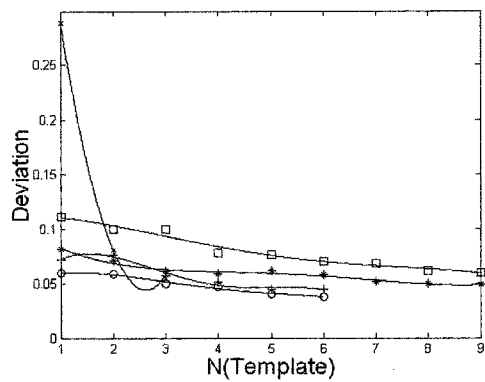
(d)



(e)



(f)



(g)

Figure 5.15 Deviation vs. N (Template) plots (a) picket fence scanned from inside (b) metal fence with brick pillar (c) flat brick wall (d) hedge (e) picket fence scanned from outside (f) brick wall with thin buttress (g) brick wall with thick buttress. Circles 'o' represent the first scan. Symbols 'x' represent the second scan. Plus '+' represent the third scan. Stars '*' represent the fourth scan. Squares '□' represent the fifth scan.

5.3.2 Second Step of Feature Extraction— Best Fit Template and its Deviation

The interesting features to be extracted next are the number of peaks for the best fit template and its corresponding deviation based on the *Deviation vs. N (Template)* plots. The slope (derivative) of the 4th order polynomial fitting between the $n=1, 2, 3, \dots, N_{max}$ point is investigated to describe how the fitting changes. If the slope increases or flattens out at $n = b + \frac{1}{2}$, then the $n=b$ template is chosen as the best fit. According to this, a threshold value -0.01 is chosen based on our experience. Then, $derivative\left(1\frac{1}{2}\right)$, $derivative\left(2\frac{1}{2}\right) \dots derivative\left(N_{max} - \frac{1}{2}\right)$ are investigated respectively. The n -peak-template is chosen as the best fit template when $derivative\left(n + \frac{1}{2}\right)$ is bigger than the threshold value -0.01 for the first time.

For example, $derivative\left(1\frac{1}{2}\right)$ is the derivative plot value when $n = 1\frac{1}{2}$. If it's bigger than -0.01 , it means that changing from a one-peak-template to a two-peak-template doesn't improve the fits much. Therefore, the best fit template is a one-peak-template. If it's smaller than -0.01 , that means a two-peak-template fit is better than a one-peak-template. We then check the derivative plot value when $n = 2\frac{1}{2}$. If it's bigger than -0.01 , it means from a two-peak-template to a three peak template, the fitting doesn't improve much. Therefore, the best fit template is taken to be a two-peak-template. If it's smaller

than -0.01 , it indicates from a two-peak-template to a three template, it improves.

Therefore, a three peak template fitting is better than a two-peak-template. Then we

check the derivative plot value when $n = 3\frac{1}{2}$ and so on. We continue to check until for

the first time derivative $\left(n + \frac{1}{2}\right)$ is bigger than -0.01 . Then, the best fit template is

considered as an n -peak-template. If the derivative $\left(n + \frac{1}{2}\right)$ is always smaller than -0.01

until the maximum number of peaks the 4^{th} order polynomial plots is reached, then the

maximum number of peaks of the 4^{th} order polynomial plots N_{max} is considered as the

number of peaks for the best fitting template.

Since we obtain the number of major reflectors based on the number of peaks, it is very important to pick the right number of peaks. We also set several auxiliary rules to help to pick the right number of peaks.

First is, in a backscatter plot, if the height ratio of a particular peak to the highest peak is less than 0.2 , it is not counted as a peak. Most peaks with the height ratio to the highest peak less than 0.2 are caused by small scatterers of rough surface of the objects and side lobes of the sound beam, not by a major reflector such as a retro-reflector or a normal reflector.

Second is, if the horizontal separation of two peaks is less than 15° , we merge them into one peak. Most of the peaks with horizontal separation less than 15° are actually caused by the same major reflector.

And third, when the maximum number of peaks N_{max} in the Deviation vs. N (Template) plot is less than 4, instead of the 4th order fitting, the $N_{max}-1$ order fitting is applied. Fitting three data points with a 4th order polynomial will cause bad fitting.

Then, for a single object, a sequence of five scans is conducted. For each scan, two features are extracted. One is number of peaks of the best fit template. The second is its corresponding deviation value. Therefore, we can form two 5-dimensional feature vectors for an object.

The first 5-dimensional feature vector is a sequence of five correct number of peaks $N=[$ number of peaks for the 1st scan, number of peaks for the 2nd scan, number of peaks for the 3rd scan, number of peaks for the 4th scan, number of peaks for the 5th scan]. The second 5-dimension feature vector is a sequence of the corresponding deviations multiplied by 10 for ease of interpretation. $\mathbf{D}=[10 \times \text{Deviation}1, 10 \times \text{Deviation}2, 10 \times \text{Deviation}3, 10 \times \text{Deviation}4, 10 \times \text{Deviation}5]$.

For example, consider the sequence of five scans for a picket fence. The 1st Scan has 1 peak and its corresponding deviation is 0.052. The 2nd Scan has 2 peaks with the deviation 0.0543. The 3rd Scan has 3 peaks with the deviation 0.0782. The 4th Scan has 2

peaks with the deviation 0.0686 . The last Scan has 1 peak with the deviation 0.0631 .

Therefore, the two 5-dimension feature vectors of that sequence of five scans for a picket fence are $N=[1,2,3,2,1]$ and $D=[0.520, 0.543, 0.782, 0.686,0.631]$. For a sequence of five scans for a flat brick wall, its two 5-dimension feature vectors are $N=[1,1,1,1,1]$ and $D=[0.549,0.704,0.752,0.998,0.673]$.

5.4 Objects Categorization

The next step is to get the computer to recognize an unknown object based on its two 5-dimension feature vectors. In my previous research, there are only two or three dimensions in a feature vector. A two-dimension feature vector $x=[x_1, x_2]$ can be displayed in a 2D phase plane. Different classifications correspond to different locations in the 2D phase plane. A 3-dimension feature vector $x=[x_1, x_2, x_3]$ can be displayed in a 3D phase space. Different categories correspond to different regions in the 3D phase space.

Feature vectors with higher dimensions (>3) are difficult to display visually. However, we can still work with them in a hyper space. The Euclidean distance of the feature vector of the unknown object to the feature vector of the object we already know in the hyper space is calculated and used to determine if the unknown object belongs to the kind of object we already know.

For the two 5-dimensional feature vectors of an unknown object, we first calculate their Euclidean distances from the two template feature vectors of a picket fence n_1 and d_1 . n_1 is the Euclidean distance between the N vector of the unknown object $N_{unknown}$ to the N vector of the picket fence $N_{picketfence}$,

$$n_1 = \left| \vec{N}_{unknown} - \vec{N}_{picketfence} \right| = \sqrt{\sum_{i=1}^5 (n_{i_unknown} - n_{i_picketfence})^2}, \quad (5.3)$$

$n_{i_unknown}$ is the number of peaks for the i^{th} scan of the unknown object ($i=1, 2, 3, 4, 5$).

$n_{i_picketfence}$ means number of peaks for the i^{th} scan of a picket fence. d_1 is the Euclidean distance between the \mathbf{D} vector of the unknown object $\mathbf{D}_{unknown}$ to the \mathbf{D} vector of the picket fence $\mathbf{D}_{picketfence}$.

$$d_1 = \left| \vec{D}_{unknown} - \vec{D}_{picketfence} \right| = \sqrt{\sum_{i=1}^5 (d_{i_unknown} - d_{i_picketfence})^2} \quad (5.4)$$

$d_{i_unknown}$ means deviation value for the i^{th} scan of the unknown object. $d_{i_picketfence}$ means deviation value for the i^{th} scan of a picket fence.

We then calculate the distances between the two feature vectors of a flat brick wall n_2 , d_2 . n_2 is the Euclidean distance between the \mathbf{N} vector of the unknown object $\mathbf{N}_{unknown}$ to the \mathbf{N} vector of the flat brick wall $\mathbf{N}_{flatbrickwall}$.

$$n_2 = \left| \vec{N}_{unknown} - \vec{N}_{flatbrickwall} \right| = \sqrt{\sum_{i=1}^5 (n_{i_unknown} - n_{i_flatbrickwall})^2} \quad (5.5)$$

$n_{i_unknown}$ is the number of peaks for the i^{th} scan of the unknown object. $n_{i_flatbrickwall}$ is the number of peaks for the i^{th} scan of a picket fence. d_1 is the Euclidean distance between the \mathbf{D} vector of the unknown object $\mathbf{D}_{unknown}$ to the \mathbf{D} vector of the flat brick wall $\mathbf{D}_{flatbrickwall}$.

$$d_2 = \left| \vec{D}_{unknown} - \vec{D}_{flatbrickwall} \right| = \sqrt{\sum_{i=1}^5 (d_{i_unknown} - d_{i_flatbrickwall})^2} \quad (5.6)$$

$d_{i_unknown}$ is the deviation value for the i^{th} scan of the unknown object. $d_{i_flatbrickwall}$ is the deviation value for the i^{th} scan of a flat brick wall.

The unknown object belongs to the kinds of objects whose two feature vectors are near to the two feature vectors of the unknown object. This means both n and d are small.

For example, Figure 5.16 is a bar chart showing the distances of two feature vectors of an unknown object to the two feature vectors of seven known objects. The horizontal axis shows the different objects. "1" represents picket fence scanned from inside. "2" represents flat brick wall. "3" represents hedge. "4" represents brick wall with thin buttress. "5" represents brick wall with thick buttress. "6" represents metal fence with brick pillar. "7" represents picket fence scanned from outside. The vertical axis shows the Euclidean distances between feature vectors of an unknown object and of the 7 objects respectively. The height of the black bar at object *No.1* is n_1 . It represents the Euclidean Distance between the N vector of the unknown object and of the picket fence scanned from inside. The height of gray bar at object *No.1* is d_1 . It represents the Euclidean Distance of the D vector between the unknown object and of the picket fence scanned from inside. The height of black bar at object *No.2* is n_2 . It represents the Euclidean Distance of the N vector between the unknown object and of the flat brick wall. The height of gray bar at object *No.2* is d_2 . It represents the Euclidean Distance of the D vector between the unknown object and of the flat brick wall and so on. Both the black bar and gray bar are the shortest when it comparing to N, D vectors of picket fence scanned from inside. Therefore, it is most probable that the unknown object is a picket fence scanned from inside. This conclusion is true.

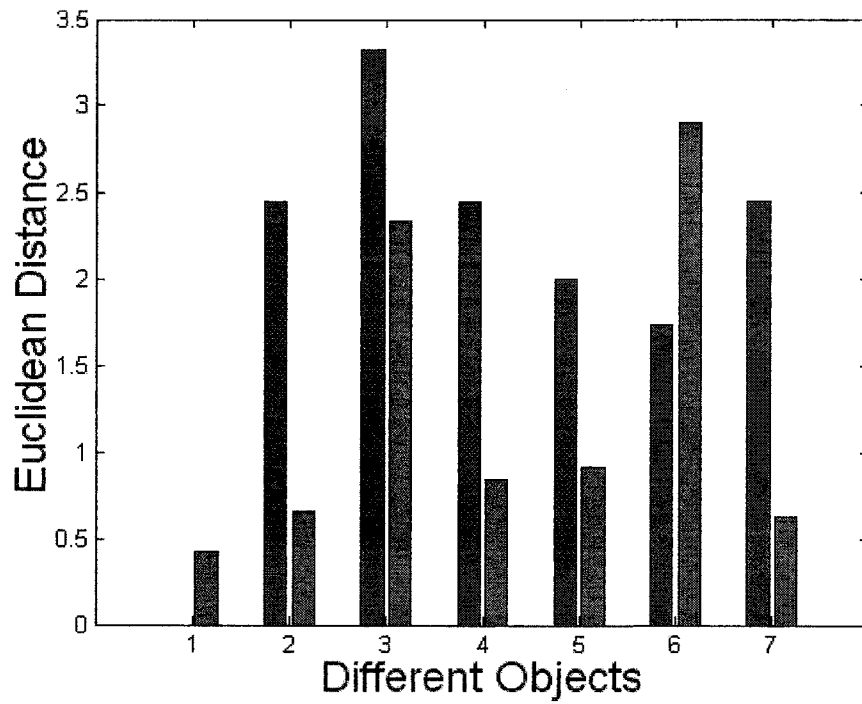
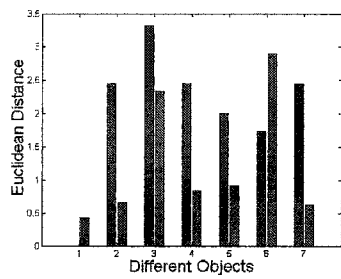
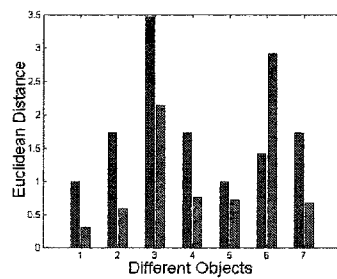


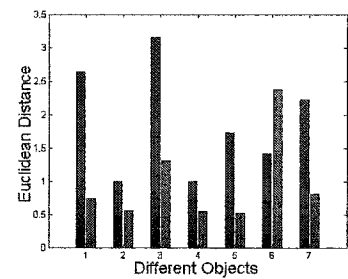
Figure 5.16 Bar chart of the Euclidean distances of the two feature vectors of an unknown object to the objects we already know. Black bar shows the Euclidean distance of their N vector. Gray bar shows the Euclidean distance of their D vector.



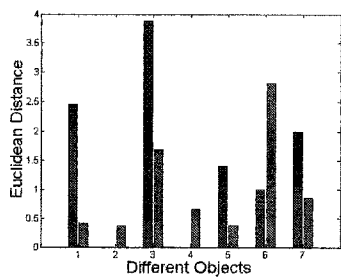
(a)



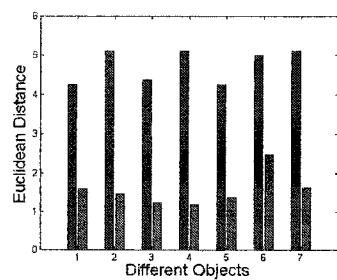
(b)



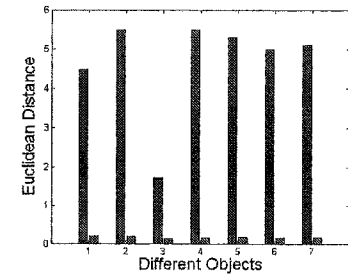
(c)



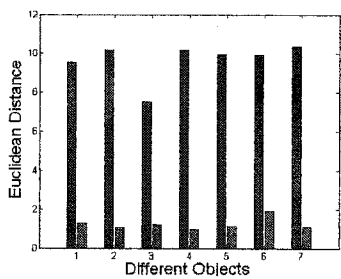
(d)



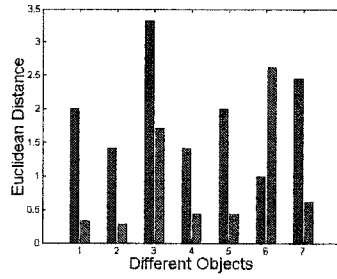
(e)



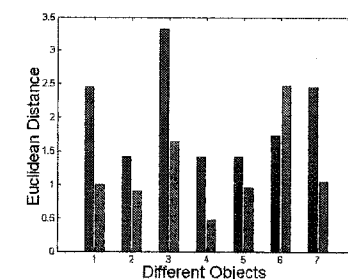
(f)



(g)



(h)



(i)

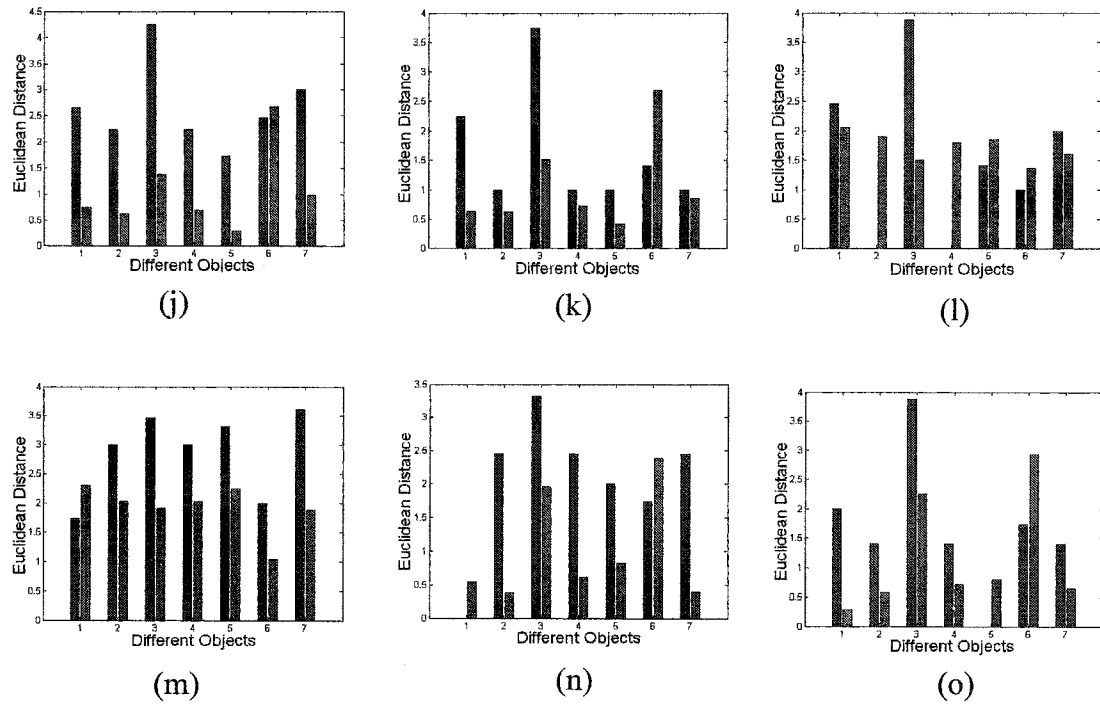


Figure 5.17 Bar chart of the Euclidean distances of the two feature vectors of fifteen unknown objects to the seven known objects. Black bar shows the Euclidean distance of their N vector. Gray bar shows the Euclidean distance of their D vector. Table 5.1 describes each case.

Figure 5.17 displays the bar charts of all the unknown objects to the seven objects we already know. The bar chart in Figure 5.17 (b) shows both the black bar and gray bar are the shortest when comparing the N, D vectors of object *No. 1*—picket fence scanned from inside. It means the unknown object is most probable a picket fence scanned from inside. Actually, it is a picket fence scanned from inside.

The bar chart in Figure 12(c) shows both the black bar and gray bar are short when comparing to N, D vectors of object *No. 2* —flat brick wall and object *No. 4* —brick wall with thin buttress. That means the most probable kinds of the unknown object are flat brick wall or brick wall with thin buttress. Actually it is a flat brick wall.

Table 5.1 displays the categorizing results of all the unknown objects. Data sets *a* and *b* are from picket fences scanned from inside. They are correctly categorized as object *No. 1*— picket fence scanned from inside. Data sets *c* and *d* are from flat brick walls. They are properly categorized as object *No. 2* — flat brick wall or object *No. 4* — brick wall with thin buttress. Data sets *e, f*, and *g* are from hedges. *e* is properly categorized as object *No. 1* — picket fence scanned from inside, object *No. 3* — hedge or object *No. 5* — brick wall with thick buttress. *f* and *g* are correctly categorized as object *No. 3* — hedges. Data sets *h* and *i* are from brick walls with thin buttress. *h* is properly categorized as object *No. 2* —flat brick wall or object *No. 4* — brick wall with thin buttress. *i* is properly categorized as object *No. 4* — brick wall with thin buttress. Data sets *j* and *k* are from

brick walls with thick buttress. They are correctly categorized as object *No.5* — brick wall with thick buttress. Data sets *l* and *m* are from metal fences with brick pillar. *l* is properly categorized as object *No.2* — flat brick wall, object *No.4* — brick wall with thin buttress or object *No.6* — metal fence with brick pillar. *m* is correctly categorized as object *No.1* — picket fence scanned from inside or object *No.6* — metal fence with brick pillar. Data sets *n* and *o* are from picket fences scanned from outside. *N* is mistaken as object *No.5* — brick wall with thick buttress. *o* is mistaken as object *No.1* — picket fence scanned from inside. Of the fifteen unknown data set, thirteen are correctly categorized, two are categorized wrong. Both of the wrong categorized data sets are from picket fence scanned from outside due to its complicated backscattering.

Table 5.1 Categorization results of the 15 data sets of the unknown objects

<i>Unknown objects</i>	<i>Object</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>
(a)		×						
(b)		×						
(c)			×		×			
(d)			×		×			
(e)		×		×		×		
(f)				×				
(g)				×				
(h)			×		×			
(i)					×			
(j)						×		
(k)						×		
(l)			×		×			×
(m)		×						×
(n)		×						
(o)						×		

“×” represents the possible choice of the unknown objects.

5.5 Conclusion

In this chapter, we successfully distinguished seven objects on campus by taking a sequence scans of each object, obtaining the corresponding backscatter vs. scan angle plots, applying deformable template matching, extracting interesting feature vectors and then categorizing them in a hyper-plane. Preliminary results show that this method correctly categorizes objects 86% of this time.

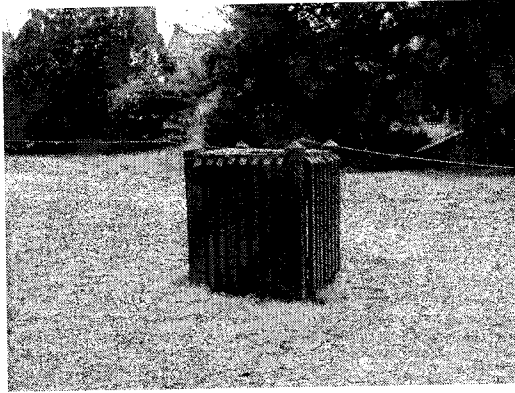
Chapter 6

Analysis of Outside Rectangular and Cylindrical Objects

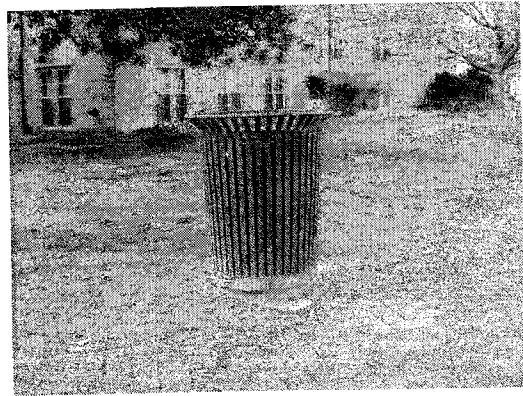
In this chapter, we will continue our research on mobile robot recognizing outside objects on campus. We will focus on outside rectangular and cylindrical objects. They are common shapes in the real world. It will be useful if the mobile robot can distinguish rectangular objects from cylindrical ones. In addition, the diameters of the cylindrical objects and the widths of the rectangular objects are their basic features based on which the robot can make a reasonable decision about what the object is. For example, a slim cylindrical object may indicate a round lamppost.

6.1 Object Scanning

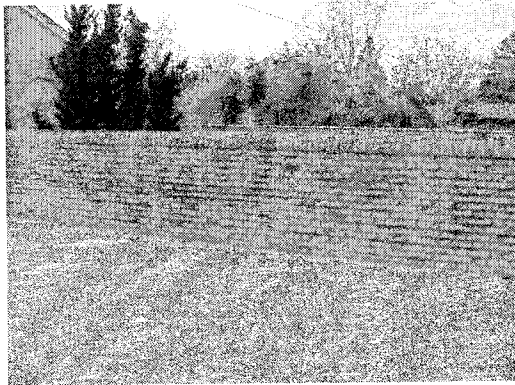
The purpose of the research in this chapter is to distinguish pairs of cylindrical and rectangular object about the same size. Three pairs are selected. The first is rectangular trashcan vs. cylindrical trashcan on campus. The second pair is flat brick wall vs. convex brick wall. The third one is square lamppost vs. round lamppost. Figure 6.1 shows the six objects. As we can see, the two objects in each pair have similar shapes. It will be difficult for the video camera on the unmanned robot to distinguish them in times of low visibility.



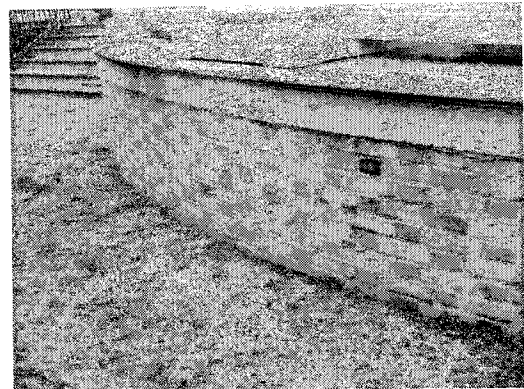
(a)



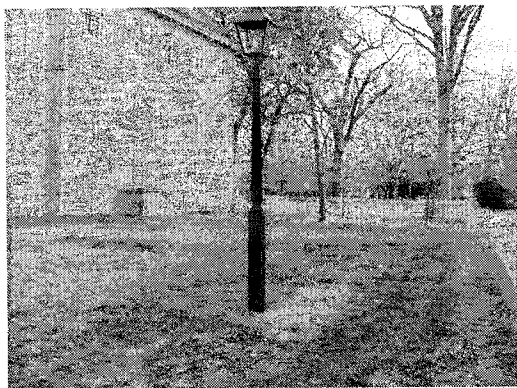
(b)



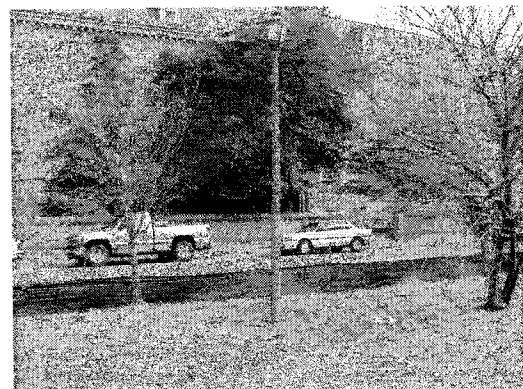
(c)



(d)



(e)



(f)

Figure 6.1 (a) Rectangular trashcan (b) Cylindrical trashcan (c) Flat brick wall (d) Convex brick wall (e) Square lamppost (f) Round lamppost.

The round lamppost, the cylindrical trashcan and the convex brick wall are symmetrical about the center axis. However, they have different radii of curvatures. The square lamppost, the rectangular trashcan and the flat brick wall are symmetrical about their center x, y -planes. They have different sizes.

At the frequency range of interest both the shape of objects and their distances from the scanner affect the back-scattered echo. This makes it possible for us to distinguish them by taking a series of scans at different distances for each object. Eight scans were conducted with the robot facing the object at different distances from three feet to ten feet. The first scan was taken with the robot three feet away from the object. The second scan data was acquired with the robot four feet away from the object and so on. In each scan, the beam was swept across each object for a range of elevation angles and the RF echoes corresponding to the horizontal fan were digitized and recorded for off-line analysis.

6.2 Data Analysis and Feature Extraction

Backscatter vs. scan angle plot might not be the appropriate plot for analysis due to the similar shapes of backscatter plots of these objects. However, backscatter plot is not the only parameter that can be used to describe the echoes. Signal energy and amplitude of the echo are also important features that carry critical information about the backscatter echoes. Here, we consider *amplitude vs. scan angle* plots, which are good at picking the echoes when the incident wave is oblique to the front surface of the object. We will then fit the *amplitude vs. scan angle* plots with 4th order polynomials. Interesting features will be extracted based on the *amplitude vs. scan angle* plots and its fittings.

Figure 6.2 shows the *amplitude vs. scan angle* plots of a rectangular trashcan when the robot is three, fourten feet away from the trashcan. Each amplitude of data point was computed by low-pass filtering, rectifying, and selecting the maximum absolute value within the window corresponding to the echo from the object. All the data sets acquired have 100 horizontal scan steps, each step representing 1° of scan angle. Figures 6.3-6.7 show the *amplitude vs. scan angle* plots of a cylindrical trashcan, a flat brick wall, a convex brick wall, a square lamppost and a round lamppost respectively.

The *amplitude vs. scan angle* plots are divided into three parts — the left region, the central region, and the right region. Generally, the central part is a relatively flat top (see Figure 6.8(a)). It consists of data points in the central region with the amplitude value around 1000. It is usually acquired by the robot when it is near the object (see Figure 6.2-6(a)). A threshold $|Amplitude_{ScanAngle} - 1000| \leq 50$ is set to find the boundaries that divides

the central region from the left and right regions. For example, in Figure 6.8 (a), the amplitude value of each data point is checked one by one starting at *scan angle* = -50 from left to right. At *scan angle* = -23, for the first time, $|Amplitude_{-23} - 1000| \leq 50$. Therefore, the left region and the central region are divided by the vertical line *scan angle* = -23. Similarly, the data points were then checked one by one starting at *scan angle* = 50 from right to left. At *scan angle* = 20, for the first time, $|Amplitude_{-23} - 1000| \leq 50$. Consequently, the straight line *scan angle* = 20 separates the central region from the right one. 4th order polynomials are fitted in the left and right region respectively.

However, as the robot moves away from the object, the width of the flat top first decreases and then vanishes. In this case, when there is no central region (see Figure 6.2(d)), the data point with the maximum amplitude value is selected to divide the *amplitude vs. scan angle* plot into two regions — the left and the right. Similarly, 4th order polynomials are fitted in the two regions respectively.

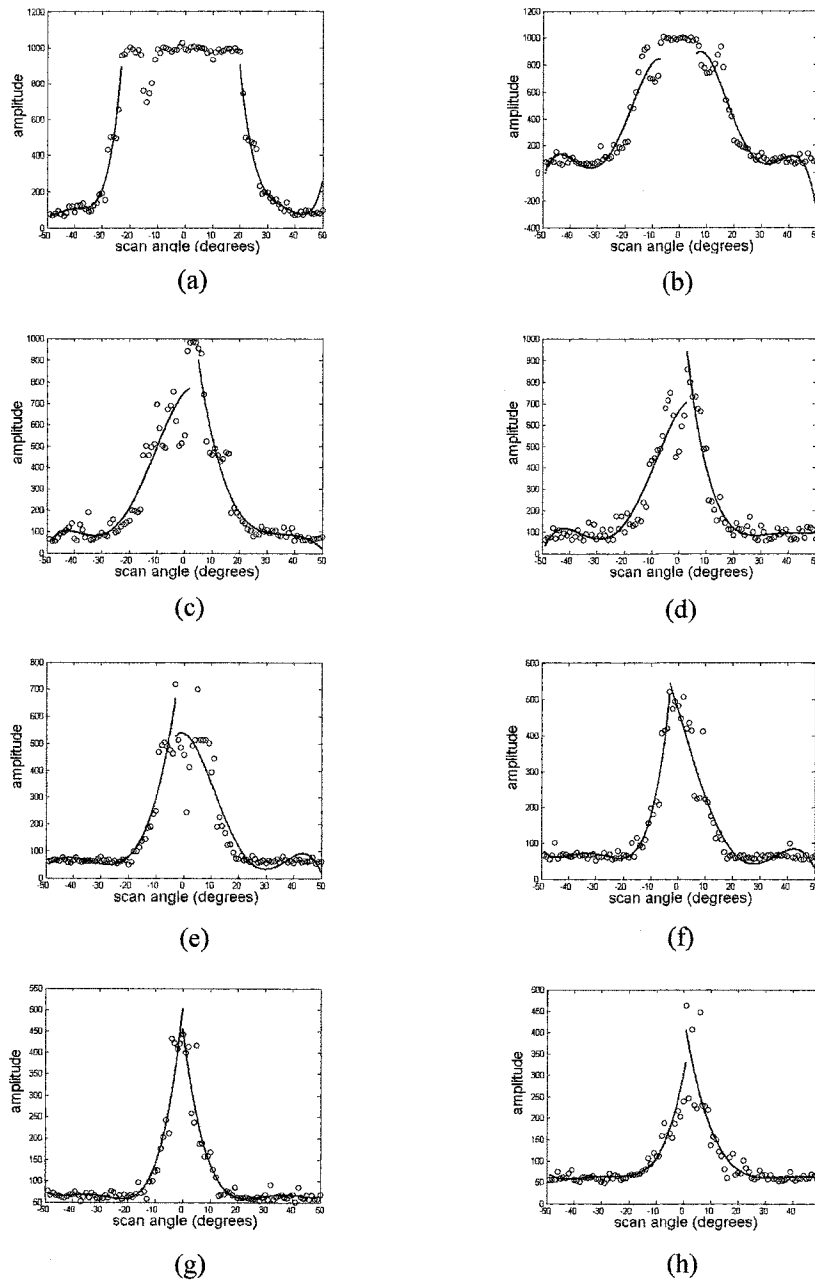


Figure 6.2 Amplitude vs. scan angle (degrees) plots of a rectangular trashcan when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the trash can.

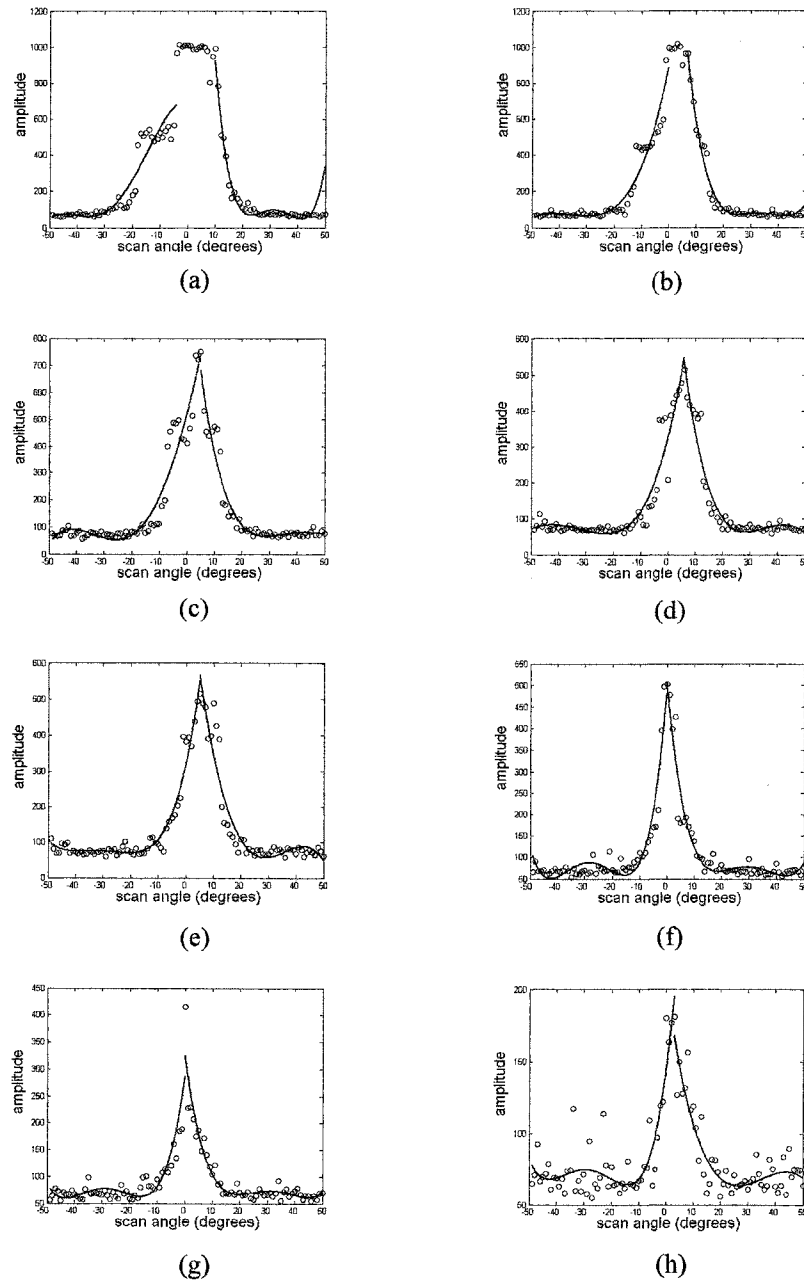


Figure 6.3 Amplitude vs. scan angle (degrees) plots of a cylindrical trashcan when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) six feet (f) eight feet (g) nine feet (h) ten feet away from the trash can.

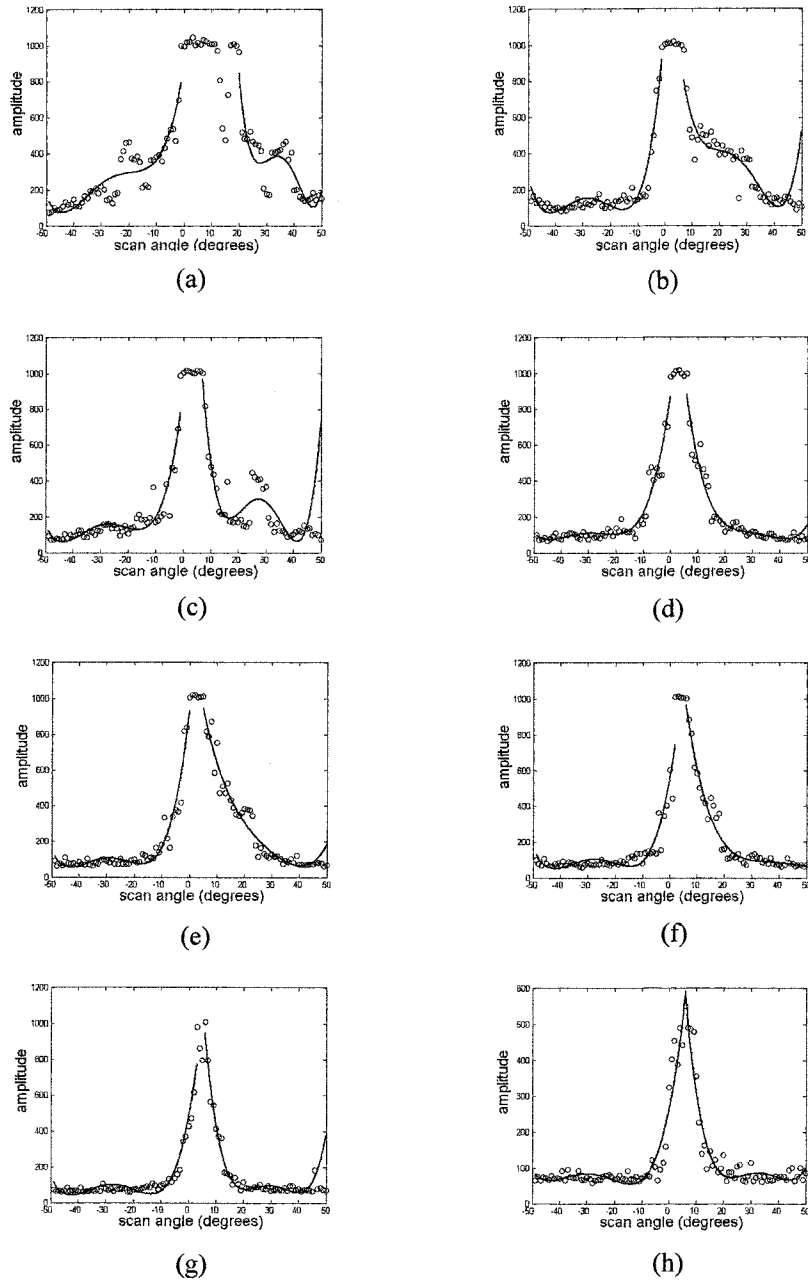


Figure 6.4 Amplitude vs. scan angle (degrees) plots of a flat brick wall when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the wall.

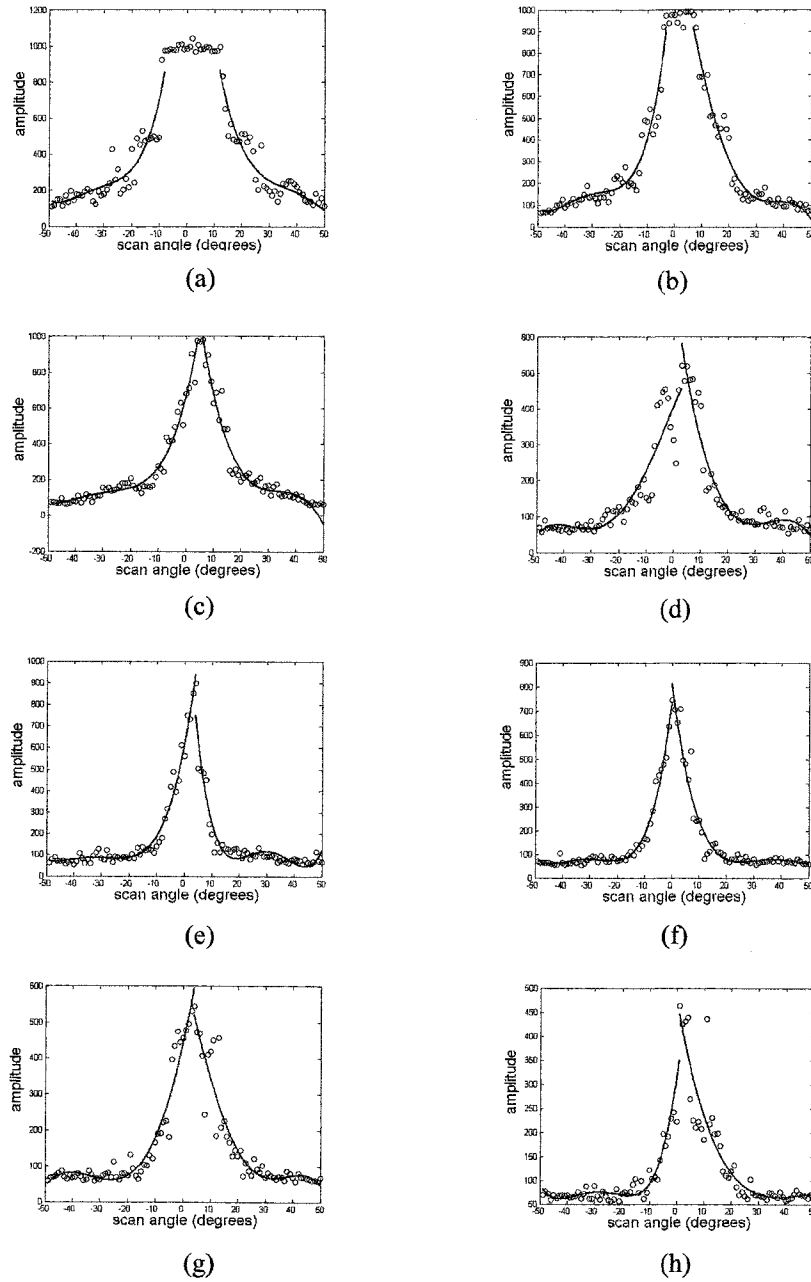


Figure 6.5 Amplitude vs. scan angle (degrees) plots of a convex brick wall when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the wall.

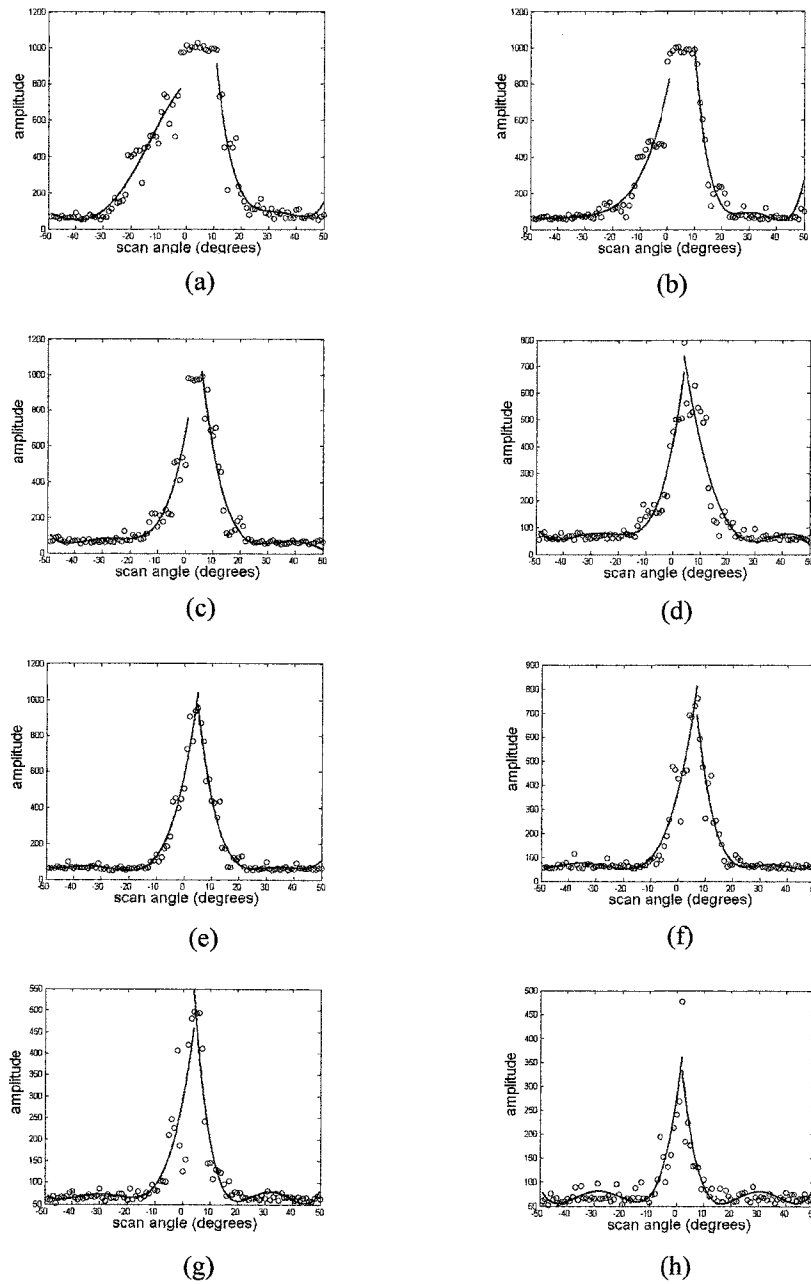


Figure 6.6 Amplitude vs. scan angle (degrees) plots of a square lamppost when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the lamppost.

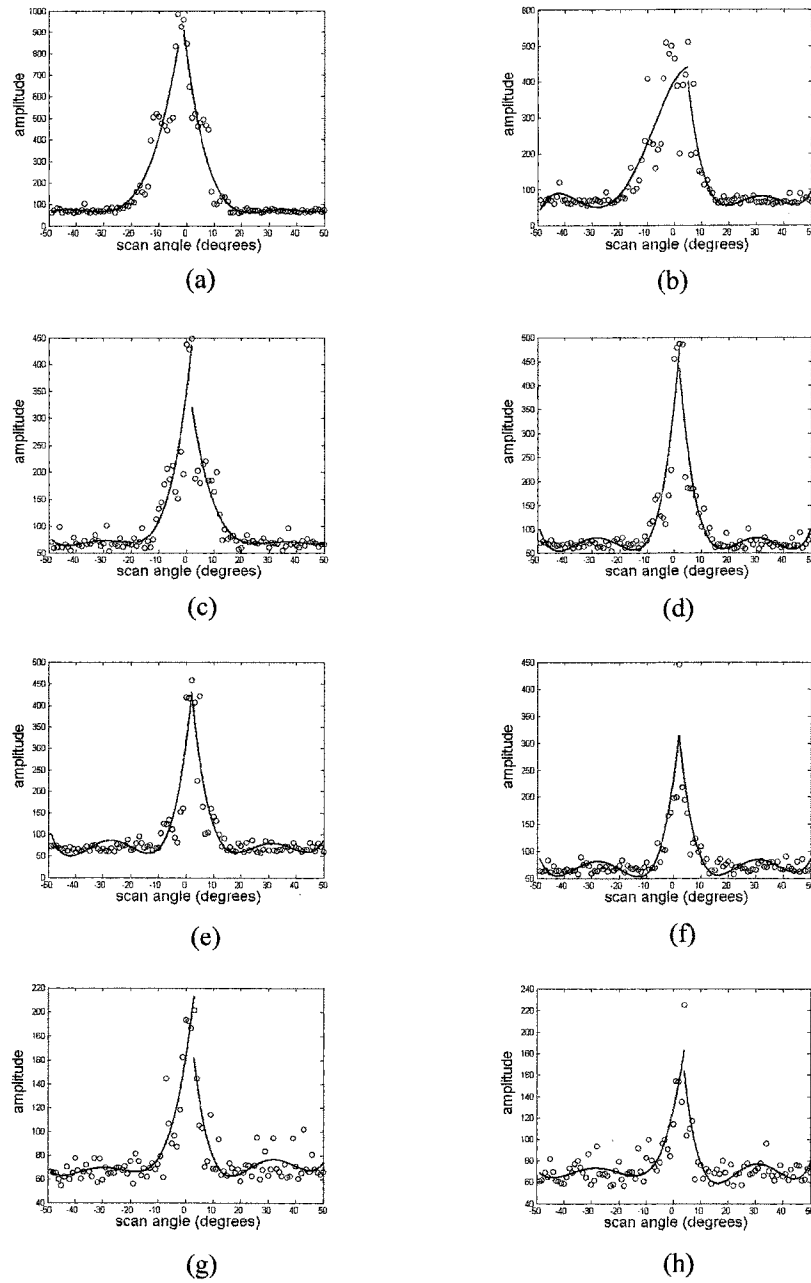


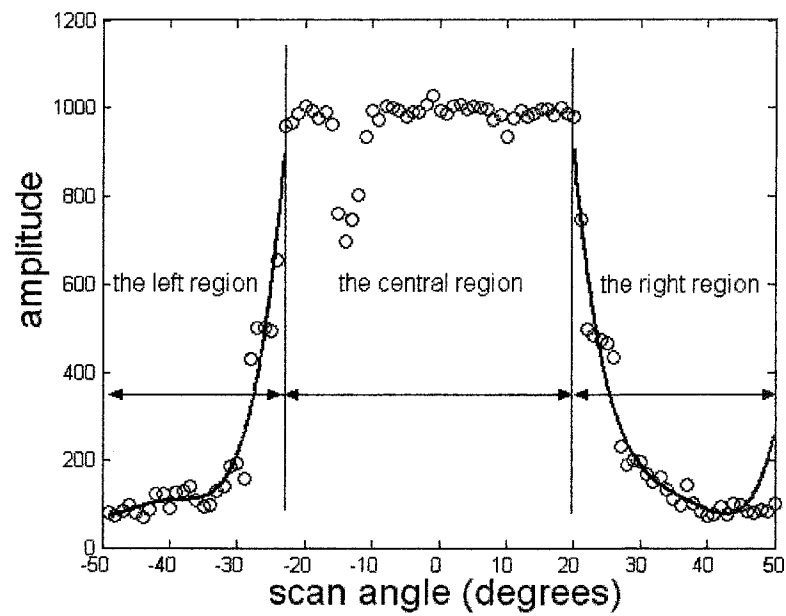
Figure 6.7 Amplitude vs. scan angle (degrees) plots of a round lamppost when the robot is (a) three feet (b) four feet (c) five feet (d) six feet (e) seven feet (f) eight feet (g) nine feet (h) ten feet away from the lamppost.

The first feature we extract is the width of the central top. It's actually the total scan angle of the central region. It is defined as *Width_1*. For example, it's 44 in Figure 6.8 (b).

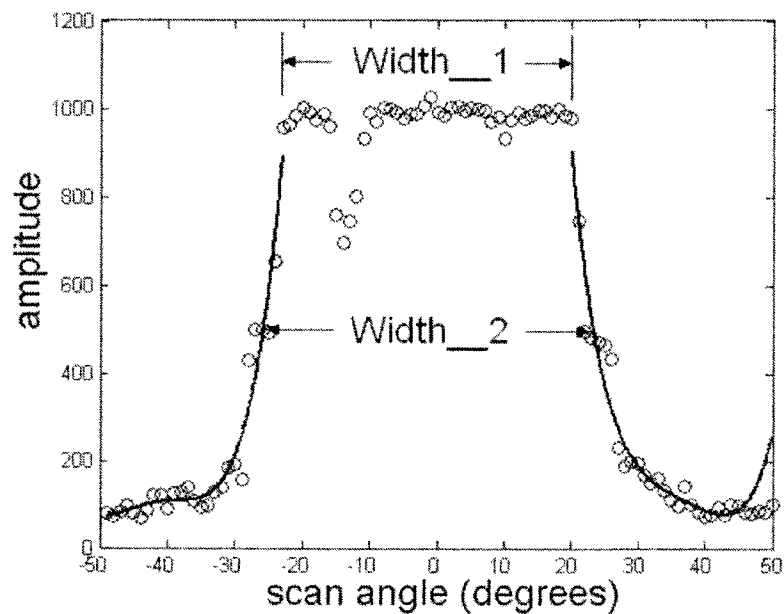
The second feature we extract is the width of the central peak at amplitude=500. It is defined as *Width_2*. It is the difference of scan angle values for the two 4th order fittings at the amplitude value 500 (see Figure 6.8(b)). It's actually the same value as the Full-Width Half Maximum (FWHM) of the central peak in the *amplitude vs. scan angle* plot when the height of the central peak is around 1000. Generally, it happens when the robot is near the object. However, when the robot moves away from the object, *Width_2* decreases (see Figure 6.2(b)-(d)). When the height of the central peak begins to decrease (see Figure 6.2(c)-(h)), *Width_2* is no longer the *FWHM* of the central peak. When the height of the central peak is less than 500, *Width_2* has no physical meaning any more.

The third feature we extract is the location of the echo in the original A-scan. It is proportional to the distance of the object and the robot. It is an important feature to which both of the previous features are related.

Table 6.1 lists the three features extracted from the *amplitude vs. scan angle* plots for the rectangular trash can in Figure 6.1. Table 6.2-6 list the features of cylindrical trashcan, flat brick wall, convex brick wall, square lamppost and round lamppost based on Figure 6.2-6 respectively.



(a)



(b)

Figure 6.8 The amplitude vs. scan angle plot of a rectangular trashcan when the robot is three feet away from the trash can. (a) The amplitude vs. scan angle is divided into three parts — the left region, the central region and the right region. (b) *Width_1* is the width of the central top while *Width_2* is the width of the central peak at the amplitude value 500.

Table 6.1 Features for the rectangular trashcan

<i>Features</i>	<i>Width_1</i>	<i>Width_2</i>	<i>Echo location</i>
<i>Distance(feet)</i>			
3	44	50.1	551
4	14	36.2	727
5	4	21.7	899
6	1	15.5	1098
7	1	10.2	1258
8	1	3.1	1430
9	1	1.1	1610
10	1	1	1744

Table 6.2 Features for the cylindrical trashcan

<i>Features</i>	<i>Width_1</i>	<i>Width_2</i>	<i>Echo location</i>
<i>Distance(feet)</i>			
3	15	25.6	577
4	8	18.1	756
5	1	9.2	922
6	1	2.8	1114
6	1	3.3	1109
8	1	1.1	1473
9	1	1	1675
10	1	1	1860

Table 6.3 Features for the flat brick wall

<i>Features</i>	<i>Width_1</i>	<i>Width_2</i>	<i>Echo location</i>
<i>Distance(feet)</i>			
3	22	29.4	579
4	9	17.7	745
5	9	14.9	922
6	7	15.4	1073
7	6	18.2	1303
8	5	13.3	1463
9	4	10.1	1647
10	1	3.3	1808

Table 6.4 Features for the convex brick wall

<i>Features</i>	<i>Width_1</i>	<i>Width_2</i>	<i>Echo location</i>
<i>Distance(feet)</i>			
3	21	33	559
4	11	23.8	742
5	3	17	927
6	1	2.8	1165
7	1	9.1	1279
8	1	8.2	1466
9	1	4	1656
10	1	1	1834

Table 6.5 Features for the square lamppost

<i>Distance(feet)</i>	<i>Features</i>	<i>Width_1</i>	<i>Width_2</i>	<i>Echo location</i>
3		14	28.3	591
4		10	19.1	728
5		6	14.4	896
6		1	7.7	1081
7		1	11.4	1295
8		1	7.5	1461
9		1	1.6	1608
10		1	1	1832

Table 6.6 Features for the round lamppost

<i>Distance(feet)</i>	<i>Features</i>	<i>Width_1</i>	<i>Width_2</i>	<i>Echo location</i>
3		3	13	599
4		1	1	792
5		1	1	955
6		1	1	1174
7		1	1	1318
8		1	1	1504
9		1	1	1671
10		1	1	1824

6.3 Further Feature Extraction and Object Classification

Both $Width_1$ and the $Width_2$ change differently for different kinds of objects. Further feature extraction is conducted to describe these changes. For each object, $Width_1$ decreases when the distance increases. When it vanishes, we set the value to be 1. Feature $Slope\ 1$ is extracted to describes how $Width_1$ changes according to the echo location until it vanishes. First, $Width_1$ vs. $echo\ location$ plot is fit to a straight line (1^{st} order polynomial). Then, the absolute value of slope of the straight line is extracted as $Slope\ 1$. For example, in Table 6.1, when the $echo\ location$ is 551, $Width_1$ is 44. When the $echo\ location$ is 1098, $Width_1$ decreases to be 1. Therefore, instead of the total eight data points, the first four data points (44, 551), (14, 727), (4, 899) and (1, 1098) are selected and fit to a straight line. Its absolute value of slope — $Slope\ 1$ is 0.0758.

Similarly, for each object, $Width_2$ also decreases when the distance increases. When the peak value of $amplitude\ vs.\ scan\ angle$ plot is less than 500 (see Figure 6.2(h)), $Width_2$ has no physical meaning. In this case, we set its value to be 1. $Slope\ 2$ is extracted to describe how $Width_2$ changes according to the echo location until it doesn't have any physical meaning. First, $Width_2$ vs. $echo\ location$ data is fit to a straight line. Then, the absolute value of the slope of the straight line is extracted as $Slope\ 2$. For example, in Table 6.2, when the echo location is 577, $Width_2$ is 25.6. When the echo location is 1675, it decreases to be 1. Consequently, the first seven data points (25.6, 577), (18.1, 756)(1, 1675) are selected and fitted with a straight line. Its absolute value of the slope — $Slope\ 2$ is 0.0219.

Slope 1 vs. *Slope 2* of each object is plotted on a phase plane (see Figure 6.9). The large value of *Slope 1&2* (*Slope 1* = 0.0758, *Slope 2*=0.0395) for the rectangular trashcan indicates that both *Width_1* and *Width_2* decrease dramatically as the robot moves away. On the other side, for the cylindrical trashcan, *Slope 1* is 0.0406 and *Slope 2* is 0.0219. It means both *Width_1* and *Width_2* decreases gradually as the robot moves away. This is a way to distinguish a rectangular trashcan from a cylindrical trashcan.

However, the brick wall pair is different. For the flat brick wall, *Slope 1*=0.0122, *Slope 2*= 0.0147. It indicates neither *Width_1* nor the *Width_2* decreases much for the flat brick wall when the robot moves away. While for the convex brick wall, its *Slope 1*=0.0332, *Slope 2*=0.0230. Comparing the flat brick wall, both of the widths for convex brick wall decrease faster when the robot moves away.

For the lamppost pair, the *Slope 1* of the square lamppost is 0.0262 and the *Slope 1* of the round lamppost is 0.0104. It indicates *Width_1* for the square lamppost changes gradually when the robot moves away while *Width_1* for the round lamppost doesn't change much. The *Slope 2* of the square lamppost is 0.0194 and the *Slope 2* of the round lamppost is 0.0622. Therefore, *Width_2* doesn't change much for the square lamppost while decreases dramatically for the round lamppost. This can be used to distinguish a square lamppost from a round lamppost.

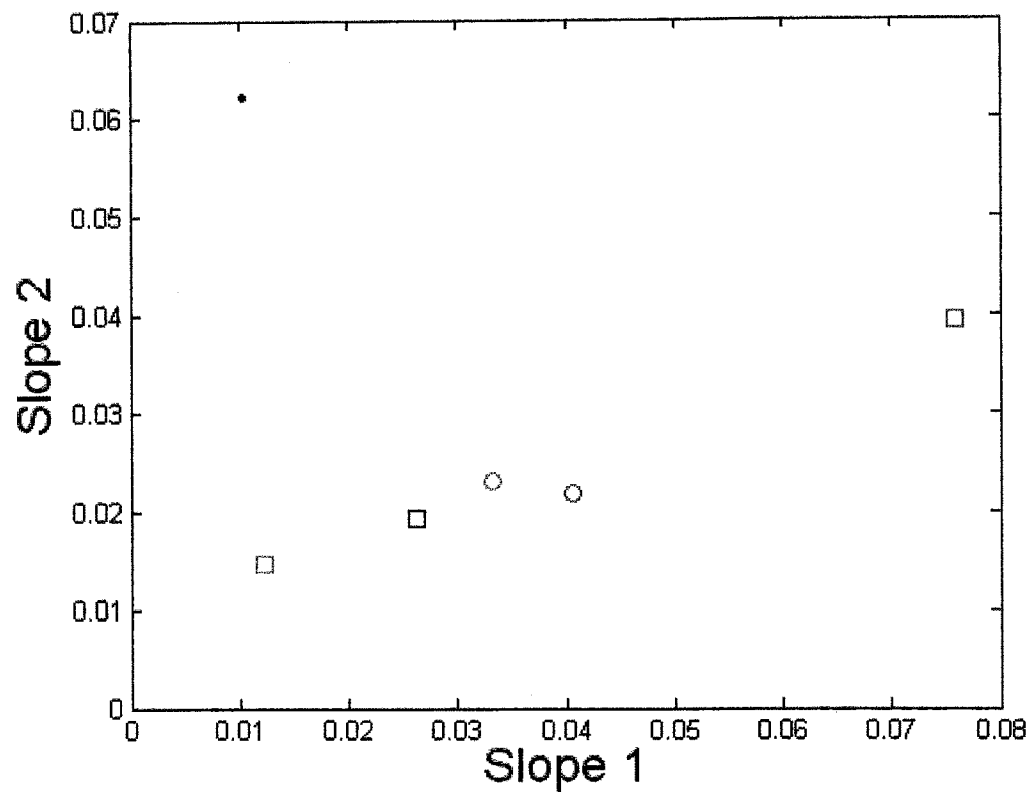


Figure 6.9 Slope 1 vs. Slope 2 phase plane of rectangular trashcan, cylindrical trashcan, flat brick wall, convex brick wall, square lamppost and round lamppost. The blue square represents the rectangular trashcan. The blue circle represents the cylindrical trashcan. The red square and circle represent the flat and convex brick wall respectively. The black square and circle represent the square and round lamppost respectively.

6.4 Conclusion

In this chapter, we present an algorithm to allow the robot to distinguish three pairs of outside objects — a rectangular trashcan vs. a cylindrical trashcan, a flat brick wall vs. a convex brick wall, and a square lamppost vs. a round lamppost. Multiple scans are conducted while the robot moves away from each object. In each scan, the beam sweeps across the object and the corresponding *amplitude vs. scan angle* plots are obtained. Three groups of features — *Width_1*, *Width_2* and *echo location* are extracted based on the *amplitude vs. scan angle* plots.

Further feature extraction is conducted based on the three groups of features. *Slope 1&2* describe how *Width_1* and *Width_2* changes when the robot moves away from the object. The *Slope 1 vs. Slope 2* phase plane separates a rectangular trashcan from a cylindrical trashcan. It also separates a flat brick wall from a convex brick wall, a square lamppost from a round lamppost. Results show this methodology is promising.

Chapter 7

Conclusions and Future Work

We have studied the physics of full-field sensor interaction with materials, structures and components in order to develop the brains for useful robots. Four generations of sonar systems have been developed in this project to automatically interpret surrounding environment. The first two are stationary 3D air-coupled ultrasound scanning systems and the last two are packaged as sensor heads for mobile robots.

All the robotic scanning systems consist of data-acquisition and motion control systems. From the LeCroy 9310M digital oscilloscope in the first generation to the Gage CompuScope 12100 data acquisition board in the second generation, to the Bitscope in the third generation, the data acquisition system has been updated for each generation. The motion control system has also been upgraded from the big DMC-1000 motion controllers to the 8"× 8" ×2" motion control system set as a sensor head for the mobile robot-RWilliam.

AT120 transducer, AR50-8 narrow beam transducer, and the modified AR50-8 narrow beam transducers are tested. R7, the modified AR-50-8 narrow beam transducer with a concave front face (radius of curvature=7.3 inches) is selected as the sensor head for the mobile robot — rWilliam.

Objects of interest have been shown in the B-scan, C-scan and 3D images. Some are displayed clearly, while most have poor spatial resolution. Imaging comes to a dead end because of the large beam size. More sophisticated techniques are needed to classify the objects.

We analyzed the echoes from indoor objects after the second generation robotic scanning system had been developed. Three methodologies were discussed to classify different materials. The first methodology compares the waveforms of echoes from three different materials visually. Materials with low acoustic impedance give weak echoes while materials with high acoustic impedance give strong echoes. Tails in the echoes are used to distinguish different materials with high acoustic impedance. Echoes from aluminum plate have short tails while echoes from cardboard have longer tails.

The second methodology applied wavelet transforms to A-scans from different objects. Different patterns in the 2D wavelet transform correspond to different A-scans. By converting 1D signal into 2D wavelet transform, it magnifies the difference and makes the differentiation easier.

More sophisticated pattern recognition technique is applied as the third methodology. Template matching analysis was conducted to compare the tested echo waveform from the unknown material with the reference echoes (template) to find the best matches to identify the material.

During comparison, the location of the reference echo is fixed, while the tested echoes are moved step by step. The difference of the two echoes is obtained at each step, and a *difference vs. step* curve is obtained wherein a small minimum difference value and a large curvature indicates a good match.

Then, the difference curves are fitted with 2^{nd} order polynomials. The reciprocal of the 2^{nd} order coefficient of the polynomials $1/a$ and its minimum value $(4ac-b^2)/4a$ are extracted and plotted on a phase plane. Smaller $1/a$ and $(4ac-b^2)/4a$ near the origin $(0,0)$ indicates a good match. Based on this, the computer automatically determines the best choices of the tested echoes.

Finally, statistical analysis is applied by adding additional target waveforms to compare against. This addition greatly improves the accuracy of the decision method. Among the ten groups of tested echoes, 50% are on the first choice and 40% are on the second choice.

However, the research in Chapter 3 was limited primarily to the analysis of indoor objects. Our goal is to give robots the ability to deal with real world issues such as “Go down to the big tree and turn left.” The mobile robot — rWilliam with the robotic scanning system packaged as the sensor head gave us the ability to do research on outside objects.

Two classes of outside cylindrical objects — trees and smooth circular poles were first studied. An algorithm has been present to distinguish trees from smooth circular poles based on analysis of backscattered sonar echoes. All data was acquired at 50 kHz via the mobile apparatus. The beam was swept across each object for a range of elevation angles and the RF echoes corresponding to the horizontal fan were digitized and recorded for off-line analysis.

The backscatter of the echo is calculated for each angle in the horizontal sweep and fitted with 5^{th} order polynomials. Smooth circular poles always give a symmetric (bell-shaped) central response whereas rough and/or irregular objects often give response less symmetric about the central peak. The deviation of backscatter from its 5^{th} order polynomials fitting and the asymmetry value of the 5^{th} order fitting curve are extracted and plotted in a phase plane. In the Asymmetry-Deviation phase plane, circles represent poles usually lie with $[0, 0.2]$ in the Asymmetry area. Dots represent the rough surface trees spread widely from minimum Asymmetry 0 to maximum 1 . Stars represent the smooth surface trees are also within $[0, 0.2]$ of the Asymmetry area.

One sweep over an object is not enough to tell a tree from a pole. Multiple scans for each object are then conducted. It is equivalent to a robot scanning an object repeatedly as it approaches. Accordingly, further feature extractions are done based on the Asymmetry-Deviation graph. Average Asymmetry-Average Squared Euclidean Distance phase plane is segmented to tell a tree from a pole by the location of the data points for

the objects interested. Results located in the small area near the origin represent poles while those located away from the origin represent trees.

We also successfully distinguished seven extended objects on campus — picket fence (inside part and outside part), metal fence with brick pillar, flat brick wall, hedge, brick wall with thin buttress and brick wall with thick buttress. We defined one unit for each object and had the robot scan its five different locations. The corresponding five *backscatter vs. scan angle* plots are obtained and fitted with N -peak deformable template matching. Deviations $I-N$ are extracted to describe the similarity of the backscatter plot and its $I-N$ peak templates.

In the first step of feature extraction, the *Deviation vs. N (Template)* curves are plotted and fitted with 4^{th} order polynomials. In the second step of feature extraction, the number of peaks for the best fit template and its corresponding deviation are extracted based on the *Deviation vs. N (Template)* plots. Two 5-dimensional feature vectors \mathbf{N} , \mathbf{D} are extracted for each object. \mathbf{N} is the feature vector describing the sequence of five right numbers of peaks and \mathbf{D} is the feature vector describing the corresponding deviation. Finally, the Euclidean distance of the feature vectors of the unknown object $\mathbf{N}_{unknown}$, $\mathbf{D}_{unknown}$ to the feature vectors of the known object \mathbf{N}_{known} , \mathbf{D}_{known} in the hyper space is calculated respectively and used to identify the unknown object. Of the fifteen unknown data sets, thirteen were correctly categorized and two were not. Results indicated that this approach for distinguishing different objects is promising.

We then successfully taught the robot to distinguish three pairs of outside objects — a rectangular trash can vs. a cylindrical trashcan, a flat brick wall vs. a convex brick wall, and a square lamppost vs. a round lamppost. Multiple scans are conducted at different distances. Instead of the *backscatter vs. scan angle* plots, *amplitude vs. scan angle* plots of each scan is obtained for analysis. Three groups of features are extracted first based on the *amplitude vs. scan angle* plots. Further feature extraction combines the three groups of features and creates a *Slope 1 vs. Slope 2* phase plane. It not only separates a rectangular trashcan from a cylindrical trashcan, but also separates a flat brick wall from a convex brick wall, a square lamppost from a round lamppost.

From Chapter 4 to Chapter 6, multiple scans of the outside objects are conducted in three different ways. The cylindrical objects are scanned by the robot passing by the objects in Chapter 4. However, in Chapter 5, different locations of the extended objects are scanned by the robot from a certain distance. In Chapter 6, the cylindrical and rectangular objects are scanned by the robot moving away from the object.

Future work will be done based on the work in this dissertation. It includes updating the current robotic scanning system. The fourth generation of robotic scanning system is current under construction. It will be packaged as a sensor head for another mobile robot — rMary. The AR 50-8 narrow beam transducer will be replaced by a phased array system for better performance. A robotic arm will be built to hold the sensor system, allowing more freedom to move the sensor. Instead of the laptop, an embedded computer will be used to control its movement and acquire the data. Future work also includes

exploring other applications such as security screening system to detect concealed weapons.

Appendix A

Air-coupled Ultrasound

Before Air-coupled systems are practical for robotics applications, we must overcome the poor reflectance and wide sound beams. The high reflectance from the transducer face is a major obstacle for air-coupled ultrasound technique. When the incident wave propagates from the transducer into the air, the reflectance from the transducer face can be expressed as

$$R = \frac{(Z_1 - Z_2)^2}{(Z_1 + Z_2)^2} \quad (\text{A.1})$$

Z_1 and Z_2 are the acoustic impedance ($Z = \rho c$) of the transducer and the air respectively. The acoustic impedance of the air Z_2 is small compared to the acoustic impedance of the transducer Z_1 . This will cause the reflectance R to be about 1. This means that instead of propagating into the air, most of the energy is reflected back to the transducer. Usually, a $\lambda/4$ matching layer is attached to the front face of the transducer to reduce the reflectance. Figure A.1 shows how it works. Two different materials are layered to create the transducer. Material 0 is the transducer, material 1 is the matching layer and material 2 the air.

Now with the matching layer, when the incident wave propagates from the transducer into the air perpendicularly, the combined reflectance can be expressed as

$$R = \frac{R_{10} + R_{12} + 2\sqrt{R_{10}R_{12}} \cos(2k_1d)}{1 + R_{10}R_{12} + 2\sqrt{R_{10}R_{12}} \cos(2k_1d)} \quad (\text{A.2})$$

Where R_{10} is the reflectance at the transducer-matching layer interface and R_{12} is the reflectance at the matching layer-air interface. They are given by

$$R_{10} = \frac{(Z_1 - Z_0)^2}{(Z_1 + Z_0)^2} \qquad R_{12} = \frac{(Z_2 - Z_1)^2}{(Z_2 + Z_1)^2} \qquad (\text{A.3})$$

Where Z_0 , Z_1 , Z_2 are the acoustic impedance of the transducer, matching layer and air respectively. When the thickness of the matching layer is $\lambda_l/4$, $\cos(2k_l d) = -1$, the combined reflectance reaches minimum. Here $k_1 = \frac{2\pi}{\lambda_1}$, λ_1 is the wavelength of the acoustic wave in the matching layer. It gives the maximum transmission of sound's propagation through the air.

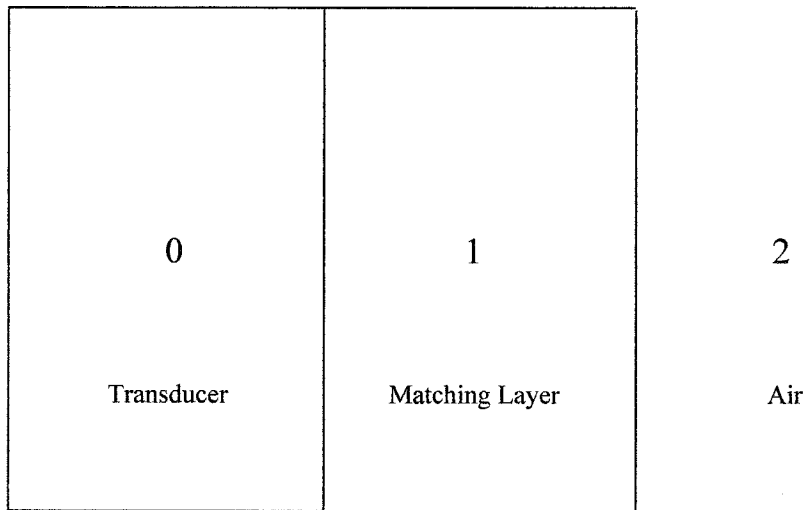


Figure A.1 0:transducer; 1: matching layer; 2: air.

Wide beam width is another obstacle for air-coupled ultrasound technology due to its poor lateral resolution.

The divergence of the ultrasonic beam δ is defined as the half diverge angle of an ultrasound beam (see Figure A.2). It's given by

$$\delta = \arcsin\left(\frac{1.22\lambda}{D}\right) \quad (\text{A.4})$$

Where λ is the wavelength in the medium and D is the diameter of the emitter.

The Airmar AR50-8 narrow beam 50kHz transducer that we use as the sensor head for rWilliam has a diameter D of 6.6cm. The wavelength λ and divergence δ are

$$\lambda = \frac{c}{f} = \frac{340(m/s)}{50000(1/s)} = 0.0068m = 0.68cm \quad (\text{A.5})$$

$$\delta = \arcsin\left(\frac{1.22\lambda}{D}\right) = \arcsin\left(\frac{1.22 \times 0.68cm}{6.6cm}\right) = \arcsin(0.1257) = 7.22^\circ \quad (\text{A.6})$$

To narrow the beam, we can choose a transducer with bigger diameter and higher frequency. However, the diameter and frequency can not been increased without limitation. Instead, lens is a more practical solution to narrow the beam here. Several different methodologies are suggested for future work to narrow the beam.

Methodology No.1 is to make an air-coupled transducer with a curved concave piezoelectric disk as shown in Figure A.3. Its focusing length f is equal to the radius R . Another way is to bond a plastic concave lens with a central thickness $\lambda/4$ to the flat-face-transducer as shown in Figure A.4.

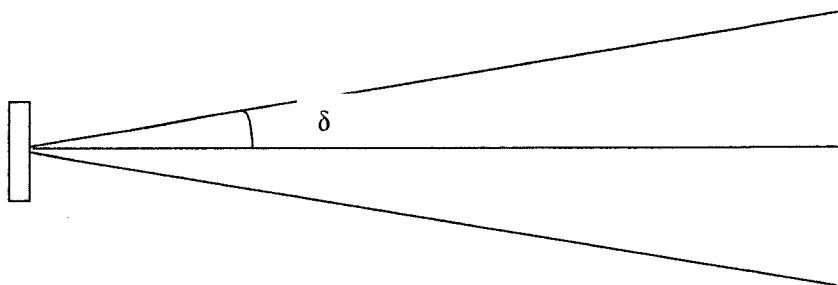


Figure A.2 An ultrasonic beam with the divergence δ .

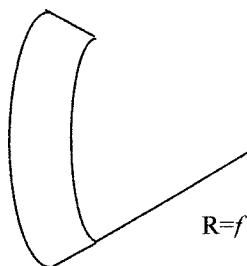


Figure A.3 A curved transducer with the focusing length f equal to the radius of curvature R .

From Snell's law, the geometrical focal length f is derived as

$$f = R \left[1 + \frac{\frac{c_{air}}{c_L} \cdot \frac{r}{R}}{\sin \left(\arcsin \frac{r}{R} - \arcsin \left(\frac{c_{air}}{c_L} \cdot \frac{r}{R} \right) \right)} \right] \quad (A.7)$$

When a concave lens is used to focus the energy, the geometrical focal length is not the same as the acoustic focal length because all transducers have a "natural" focus known as the near field or Fresnel zone. It is defined as

$$l_0 = \frac{D^2}{4\lambda} \quad (A.8)$$

Their relationship can be described by Eq.A.9 and Figure A.5. The acoustic focal length is always shorter than the near field length of the transducer. In the near field, the acoustic field is basically cylindrical, with a diameter slightly less than the diameter of the emitter, and the intensity of the acoustic waves oscillates along the axis of the transducer. The zone lying beyond l_0 is called the far field. In the far field, the intensity of the acoustic waves diverges as the inverse of the square of the distance from the transducer and small oscillations appear in the radial direction. Most of the acoustic energy is contained in the near field [126].

$$\frac{F}{l_0} = \frac{1}{1 - \frac{F_{ac}}{l_0}} \left[\frac{F_{ac}}{l_0} - 0.635 \left(\frac{F_{ac}}{l_0} \right)^2 + 0.2128 \left(\frac{F_{ac}}{l_0} \right)^3 \right] \quad (A.9)$$

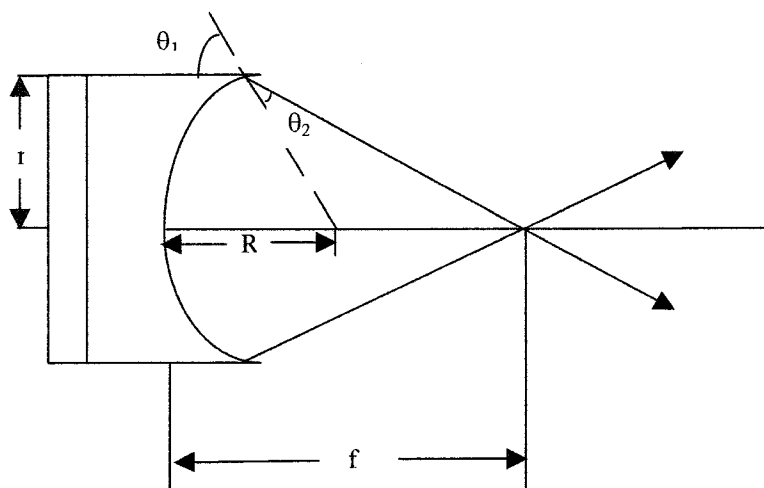


Figure A.4 Plastic convex lens bonds to a flat face transducer.

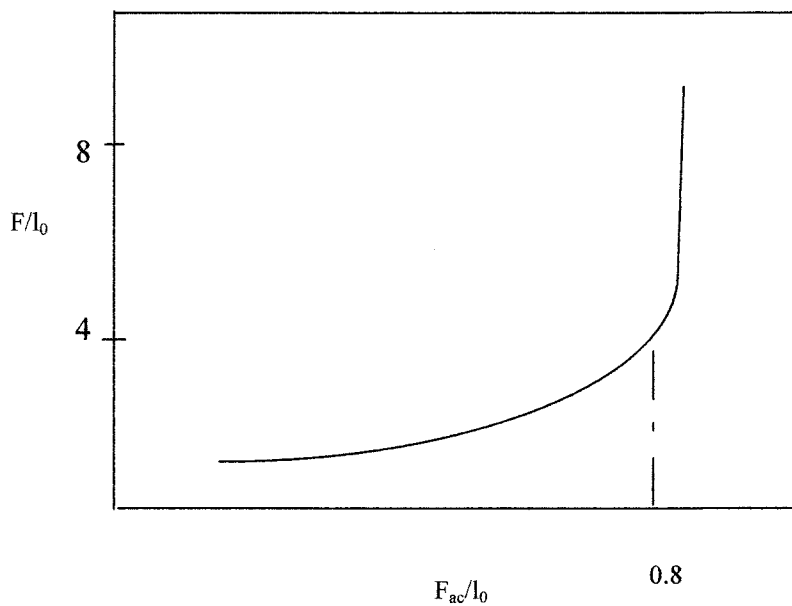


Figure A.5 Relationship between F/l_0 and F_{ac}/l_0 .

We can also bond a plastic Axicon lens to the flat-face-transducer (see Figure A.6) or use a conical face lens (see Figure A.7). It enlarges the effective diameter and reduces the divergence δ .

We can use a right angle conical and two parabolic reflectors to redirect the sound waves. In Figure A.8, parallel sound waves in the near field meet with a right angle conical face and are redirected perpendicular to the original path. Two parabolic reflectors are placed in their paths to reflect and focus them to one point. The focal length can be determined by adjusting the orientations of the parabolic mirrors.

We can even focus the sound beam with an air-coupled Fresnel zone-plate. As shown in Figure A.9, ultrasonic plane waves are created by an air-coupled transducer and arrive in-phase over the back surface of the zone-plate aperture. The aperture consists of alternating rings or zones of solid material and air, as indicated by the black and white regions. The solid (white) regions do not allow the incident air waves to pass, and so may be considered opaque, whereas the regions open to air (black) freely allow the waves to pass, and so may be considered transparent. The various dimensions are specially chosen according to well-known principles so that all waves emanating from the various transparent zones arrive in-phase at a point along the zone-plate axis at a predetermined focal length Z_0 . The fact that all transmitted waves arrives in phase at the focus leads to constructive interference with a dramatic rise in ultrasonic intensity. Since the zone-plate structure is axially symmetric, the ultrasonic intensity reaches an appreciable level only

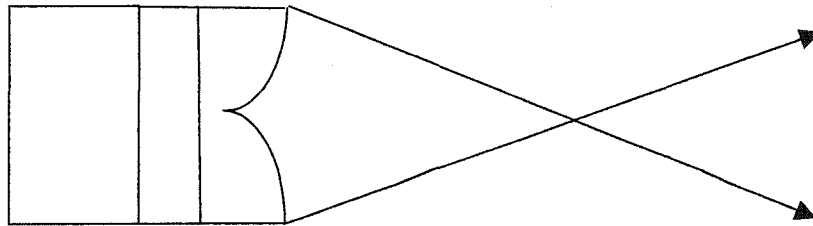


Figure A.6 A plastic Axicon lens attached to a flat face transducer.

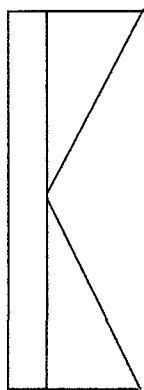


Figure A.7 A conical lens attached to the flat face transducer.

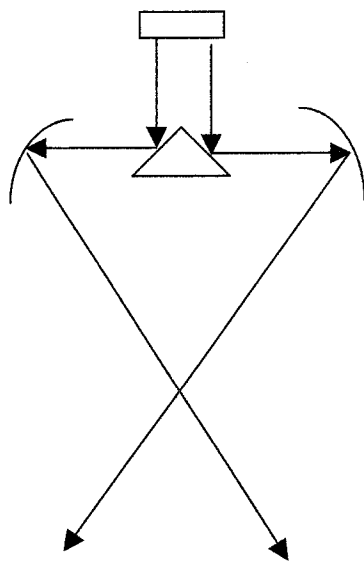


Figure A.8 A right angle conical and two parabolic reflectors to redirect the sound waves.

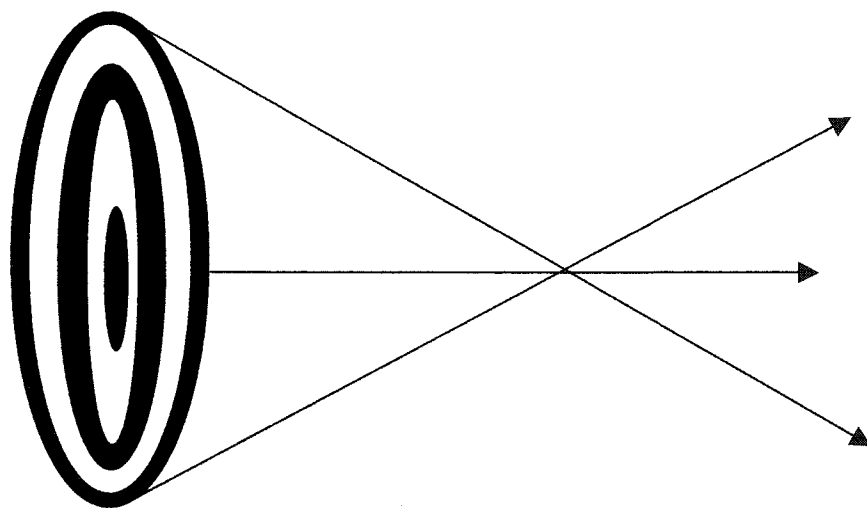


Figure A.9 An air-coupled Fresnel zone-plate.

on axis at Z_0 . This creates a well-focused ultrasonic field that is ideal for imaging applications in air.

Another methodology is to use a double conical lens as shown in Figure 2.14. The parallel sound waves are redirected twice when they reach the face of the conical lens. They are focused at a point out of the conical lens. The distance H can be calculated by Equation A.10

$$H = 2r \frac{\sin \beta \sin(\beta - 2\beta_0)}{\sin 2(\beta - \beta_0)} \quad (\text{A.10})$$

Where β and β_0 are the angles of the individual conical lenses and the angles of the lenses determines the focal length.

Besides lenses, there is another popular solution to narrow the beam in which electronic focusing or steering is used by delayed excitation of transducers in a phased array. Figure 2.15 displays a phased-array system. A different delayed pulse sequencing of the whole array of elements is introduced to form the unique interference pattern. It creates a highly focused ultrasound beam perpendicular to the transducer face. The depth of the focal zone can be simply changed by varying the delays applied to the individual elements.

Once the high reflectance and wide beam width problems have been overcome, the air-coupled ultrasound scanning system is ready for use.

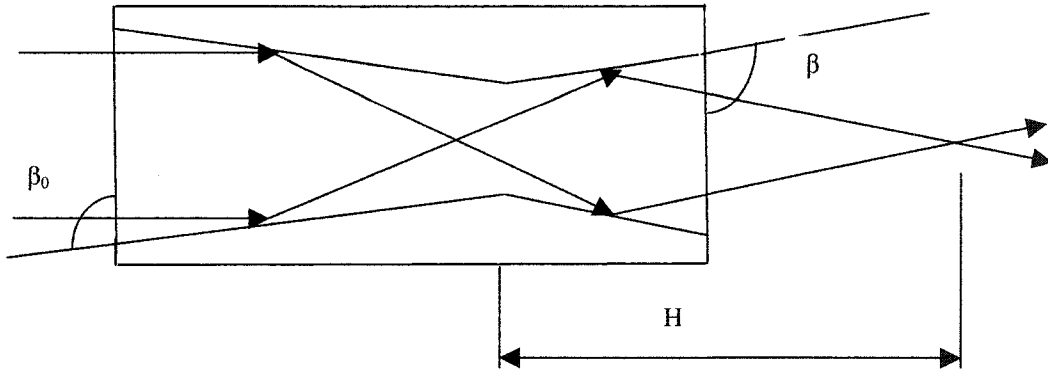


Figure A.10 Double conical lens.

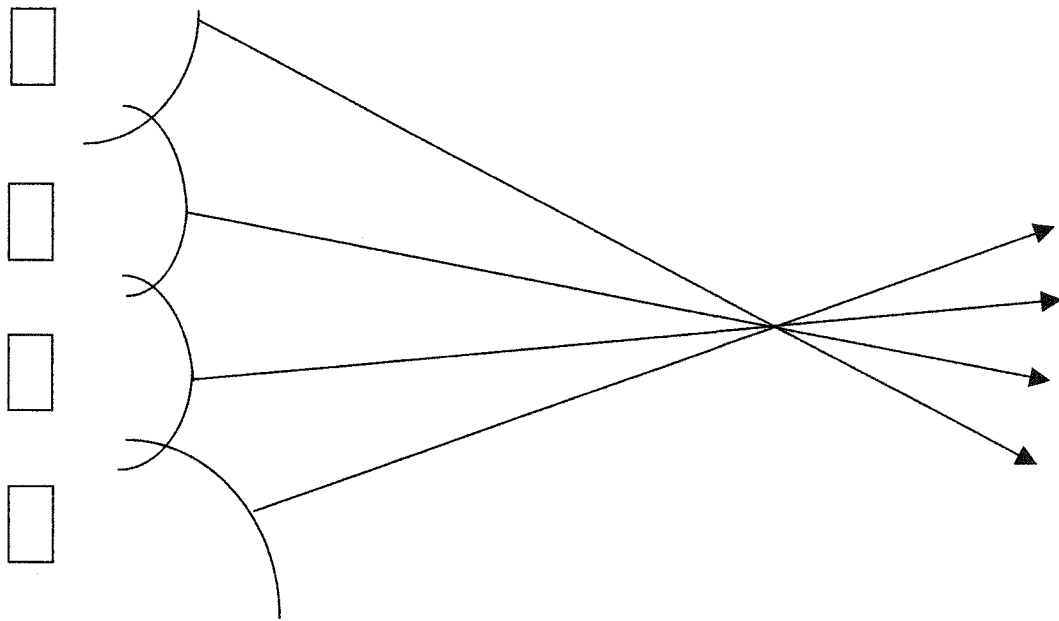


Figure A.11 Electric focusing or steering by an phased array.

Appendix B

Matlab Code for Template Matching Analysis

Try_A1.m

```

x1=load('Nov2002A1');
save NovA1 x1;
x2=load('Nov2002CardBoard');
save NovCardBorad x2;
x3=load('Nov2002FoamBlock');
save NovFoamBlock x3;
x4=load('Nov2002Plant');
save NovPlant x4;
x5=load('Nov2002BrickWall');
save NovBrickWall x5;
x6=load('Nov2002IvyOnBrickWall');
save NovIvyOnBrickWall x6;
x7=load('Nov2002Block');
save NovBlock x7;
x8=load('Nov2002Chair');
save NovChair x8;
x9=load('Nov2002ChairWithBag');
save NovChairWithBag x9;
x10=load('Nov2002ThinCardBoard');
save NovThinCardBoard x10;

%The following is the test signal.
y1=load('Nov2002A1Test');
save NovA1Test y1;

for i=1:400
    x1_echo(i)=x1(i+1100);
    x2_echo(i)=x2(i+1100);
    x3_echo(i)=x3(i+1085);
    x4_echo(i)=x4(i+700);
    x5_echo(i)=x5(i+800);
    x6_echo(i)=x6(i+880);
    x7_echo(i)=x7(i+1230);
    x8_echo(i)=x8(i+570);
    x9_echo(i)=x9(i+610);
    x10_echo(i)=x10(i+775);

```

```
end

f1=fft(x1_echo);
f2=fft(x2_echo);
f3=fft(x3_echo);
f4=fft(x4_echo);
f5=fft(x5_echo);
f6=fft(x6_echo);
f7=fft(x7_echo);
f8=fft(x8_echo);
f9=fft(x9_echo);
f10=fft(x10_echo);

for i=1:5
    f1(i)=0;
    f2(i)=0;
    f3(i)=0;
    f4(i)=0;
    f5(i)=0;
    f6(i)=0;
    f7(i)=0;
    f8(i)=0;
    f9(i)=0;
    f10(i)=0;
end

for i=396:400
    f1(i)=0;
    f2(i)=0;
    f3(i)=0;
    f4(i)=0;
    f5(i)=0;
    f6(i)=0;
    f7(i)=0;
    f8(i)=0;
    f9(i)=0;
    f10(i)=0;
end

x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
x3_echo_improve=ifft(f3);
x4_echo_improve=ifft(f4);
x5_echo_improve=ifft(f5);
```

```

x6_echo_improve=ifft(f6);
x7_echo_improve=ifft(f7);
x8_echo_improve=ifft(f8);
x9_echo_improve=ifft(f9);
x10_echo_improve=ifft(f10);

```

% The following program calculate the envelop of both the reference echo and the echoes of interesting material.

```

x1_echo_improveshape2=envelope(x1_echo_improve);
x2_echo_improveshape2=envelope(x2_echo_improve);
x3_echo_improveshape2=envelope(x3_echo_improve);
x4_echo_improveshape2=envelope(x4_echo_improve);
x5_echo_improveshape2=envelope(x5_echo_improve);
x6_echo_improveshape2=envelope(x6_echo_improve);
x7_echo_improveshape2=envelope(x7_echo_improve);
x8_echo_improveshape2=envelope(x8_echo_improve);
x9_echo_improveshape2=envelope(x9_echo_improve);
x10_echo_improveshape2=envelope(x10_echo_improve);

for i=1:10
error100_1(i)=matchfilter(y1,x1_echo_improveshape2,10*i+680);
error100_2(i)=matchfilter(y1,x2_echo_improveshape2,10*i+680);
error100_3(i)=matchfilter(y1,x3_echo_improveshape2,10*i+680);
error100_4(i)=matchfilter(y1,x4_echo_improveshape2,10*i+680);
error100_5(i)=matchfilter(y1,x5_echo_improveshape2,10*i+680);
error100_6(i)=matchfilter(y1,x6_echo_improveshape2,10*i+680);
error100_7(i)=matchfilter(y1,x7_echo_improveshape2,10*i+680);
error100_8(i)=matchfilter(y1,x8_echo_improveshape2,10*i+680);
error100_9(i)=matchfilter(y1,x9_echo_improveshape2,10*i+680);
error100_10(i)=matchfilter(y1,x10_echo_improveshape2,10*i+680);
end

error1min=min(error100_1);
error2min=min(error100_2);
error3min=min(error100_3);
error4min=min(error100_4);
error5min=min(error100_5);
error6min=min(error100_6);
error7min=min(error100_7);
error8min=min(error100_8);
error9min=min(error100_9);

```

```

error10min=min(error100_10);

%hold off
n=10:10:100;
%plot(n,error100_1,'k')
%xlabel('shifting number of points'); ylabel('difference')
%title('Echo from Aluminum Plate')
%hold on
%plot(n,error100_2,'ok')
%hold on
%plot(n,error100_3,'xk')
%hold on
%plot(n,error100_4,'+k')
%hold on
%plot(n,error100_5,'*k')
%hold on
%plot(n,error100_6,'sk')
%hold on
%plot(n,error100_7,'dk')
%hold on
%plot(n,error100_8,'vk')
%hold on
%plot(n,error100_9,'^k')
%hold on
%plot(n,error100_10,'<k')
%hold on
%legend('Al','Cardboard','Foamblock','Plant','Brickwall','Ivy on
brickwall','Block','Chair','Chairwithbag','Thin cardboard')
%plot(n,error100_1,'-k',n,error100_2,'-k',n,error100_3,'-k',n,error100_4,'-
k',n,error100_5,'-k',n,error100_6,'-k',n,error100_7,'-k',n,error100_8,'-k',n,error100_9,'-
k',n,error100_10,'-k','linewidth',1.5);
%hold off

cof1=polyfit(n,error100_1,2);
cof2=polyfit(n,error100_2,2);
cof3=polyfit(n,error100_3,2);
cof4=polyfit(n,error100_4,2);
cof5=polyfit(n,error100_5,2);
cof6=polyfit(n,error100_6,2);
cof7=polyfit(n,error100_7,2);
cof8=polyfit(n,error100_8,2);
cof9=polyfit(n,error100_9,2);
cof10=polyfit(n,error100_10,2);
ylocation1=(4*cof1(1)*cof1(3)-cof1(2)*cof1(2))/(4*cof1(1));
ylocation2=(4*cof2(1)*cof2(3)-cof2(2)*cof2(2))/(4*cof2(1));
ylocation3=(4*cof3(1)*cof3(3)-cof3(2)*cof3(2))/(4*cof3(1));

```

```

ylocation4=(4*cof4(1)*cof4(3)-cof4(2)*cof4(2))/(4*cof4(1));
ylocation5=(4*cof5(1)*cof5(3)-cof5(2)*cof5(2))/(4*cof5(1));
ylocation6=(4*cof6(1)*cof6(3)-cof6(2)*cof6(2))/(4*cof6(1));
ylocation7=(4*cof7(1)*cof7(3)-cof7(2)*cof7(2))/(4*cof7(1));
ylocation8=(4*cof8(1)*cof8(3)-cof8(2)*cof8(2))/(4*cof8(1));
ylocation9=(4*cof9(1)*cof9(3)-cof9(2)*cof9(2))/(4*cof9(1));
ylocation10=(4*cof10(1)*cof10(3)-cof10(2)*cof10(2))/(4*cof10(1));

```

```

hold off
plot(1/cof1(1),ylocation1,'k');
xlabel('1/a of the function y=ax^2+bx+c'); ylabel('Ymin');
%title('Echo from Aluminum Plate')
%axis([0 5 0 50000]);
hold on
plot(1/cof2(1),ylocation2,'ok');
hold on
plot(1/cof3(1),ylocation3,'xk');
hold on
plot(1/cof4(1),ylocation4,'+k');
hold on
plot(1/cof5(1),ylocation5,'*k');
hold on
plot(1/cof6(1),ylocation6,'sk');
hold on
plot(1/cof7(1),ylocation7,'dk');
hold on
plot(1/cof8(1),ylocation8,'vk');
hold on
plot(1/cof9(1),ylocation9,'^k');
hold on
plot(1/cof10(1),ylocation10,'<k');
%legend('Al','Cardboard','Foamblock','Plant','Brickwall','Ivy on
brickwall','Block','Chair','Chairwithbag','Thin cardboard')
hold off

```

```

R1=sqrt((1/cof1(1))^2+(ylocation1/10000)^2);
R2=sqrt((1/cof2(1))^2+(ylocation2/10000)^2);
R3=sqrt((1/cof3(1))^2+(ylocation3/10000)^2);
R4=sqrt((1/cof4(1))^2+(ylocation4/10000)^2);
R5=sqrt((1/cof5(1))^2+(ylocation5/10000)^2);
R6=sqrt((1/cof6(1))^2+(ylocation6/10000)^2);
R7=sqrt((1/cof7(1))^2+(ylocation7/10000)^2);
R8=sqrt((1/cof8(1))^2+(ylocation8/10000)^2);
R9=sqrt((1/cof9(1))^2+(ylocation9/10000)^2);
R10=sqrt((1/cof10(1))^2+(ylocation10/10000)^2);

```



```
R=[R1 R2 R3 R4 R5 R6 R7 R8 R9 R10];
ssR=sort(R);
```

```
switch ssR(1)
case R1
    fprintf('The first choice is Aluminum plate.\n');
case R2
    fprintf('The first choice is cardboard.\n');
case R3
    fprintf('The first choice is foamblock.\n');
case R4
    fprintf('The first choice is plant.\n');
case R5
    fprintf('The first choice is brick wall.\n');
case R6
    fprintf('The first choice is ivy on brick wall.\n');
case R7
    fprintf('The first choice is block.\n');
case R8
    fprintf('The first choice is chair.\n');
case R9
    fprintf('The first choice is chair with bag.\n');
case R10
    fprintf('The first choice is thin card board.\n');
otherwise
end
```

```
switch ssR(2)
case R1
    fprintf('The second choice is Aluminum plate.\n');
case R2
    fprintf('The second choice is cardboard.\n');
case R3
    fprintf('The second choice is foamblock.\n');
case R4
    fprintf('The second choice is plant.\n');
case R5
    fprintf('The second choice is brick wall.\n');
case R6
    fprintf('The second choice is ivy on brick wall.\n');
case R7
    fprintf('The second choice is block.\n');
case R8
    fprintf('The second choice is chair.\n');
```

```
case R9
    fprintf('The second choice is chair with bag.\n');
case R10
    fprintf('The second choice is thin card board.\n');
otherwise
end
```

```
switch ssR(3)
case R1
    fprintf('The third choice is Aluminum plate.\n');
case R2
    fprintf('The third choice is cardboard.\n');
case R3
    fprintf('The third choice is foamblock.\n');
case R4
    fprintf('The third choice is plant.\n');
case R5
    fprintf('The third choice is brick wall.\n');
case R6
    fprintf('The third choice is ivy on brick wall.\n');
case R7
    fprintf('The third choice is block.\n');
case R8
    fprintf('The third choice is chair.\n');
case R9
    fprintf('The third choice is chair with bag.\n');
case R10
    fprintf('The third choice is thin card board.\n');
otherwise
end
```

```
switch ssR(4)
case R1
    fprintf('The fourth choice is Aluminum plate.\n');
case R2
    fprintf('The fourth choice is cardboard.\n');
case R3
    fprintf('The fourth choice is foamblock.\n');
case R4
    fprintf('The fourth choice is plant.\n');
case R5
    fprintf('The fourth choice is brick wall.\n');
case R6
    fprintf('The fourth choice is ivy on brick wall.\n');
case R7
```

```

    fprintf('The fourth choice is block.\n');
case R8
    fprintf('The fourth choice is chair.\n');
case R9
    fprintf('The fourth choice is chair with bag.\n');
case R10
    fprintf('The fourth choice is thin card board.\n');
otherwise
end

```

```

switch ssR(5)
case R1
    fprintf('The fifth choice is Aluminum plate.\n');
case R2
    fprintf('The fifth choice is cardboard.\n');
case R3
    fprintf('The fifth choice is foamblock.\n');
case R4
    fprintf('The fifth choice is plant.\n');
case R5
    fprintf('The fifth choice is brick wall.\n');
case R6
    fprintf('The fifth choice is ivy on brick wall.\n');
case R7
    fprintf('The fifth choice is block.\n');
case R8
    fprintf('The fifth choice is chair.\n');
case R9
    fprintf('The fifth choice is chair with bag.\n');
case R10
    fprintf('The fifth choice is thin card board.\n');
otherwise
end

```

envelope.m

```

function x_echo_improveshape2=envelope(x_echo_improve)
j=1;
for i=2:399
    if ((x_echo_improve(i)-x_echo_improve(i-1))>0)&((x_echo_improve(i+1)-
x_echo_improve(i))<0)
        x_echo_improveshape2(j)=abs(x_echo_improve(i));
        j=j+1;
    else
    end
end
end

```

matchfilter.m

% testsignal is the original 2000 points y1 signal while samplesignal is the improved 400 points xn signals.

```
function error=matchfilter(testsignal,samplesignal,shiftnumber)
    for i=1:400
        y1_echo(i)=testsignal(i+shiftnumber);
    end
    f11=fft(y1_echo);
    for i=1:5
        f11(i)=0;
    end
    for i=396:400
        f11(i)=0;
    end
    y1_echo_improve=ifft(f11);
    y1_echo_improveshape2=envelope(y1_echo_improve);
    error=zerosize(samplesignal, y1_echo_improveshape2);
```

zeropad.m

% zero padding for the shorter one among the tested and the original signal

```
function error=zeropad(x_echo_improveshape2, y_echo_improveshape2)
if length(y_echo_improveshape2)>length(x_echo_improveshape2)
    for i=(length(x_echo_improveshape2)+1):length(y_echo_improveshape2)
        x_echo_improveshape2(i)=0;
    end
elseif length(y_echo_improveshape2)<length(x_echo_improveshape2)
    for i=(length(y_echo_improveshape2)+1):length(x_echo_improveshape2)
        y_echo_improveshape2(i)=0;
    end
else
end

error=0;
for i=1:length(y_echo_improveshape2)
    error=error+(y_echo_improveshape2(i)-x_echo_improveshape2(i)).^2;
end

error=error/length(y_echo_improveshape2);
```

try_AI_net.m

```

x1=load('Nov2002AI');
save NovAI x1;
x2=load('Nov2002CardBoard');
save NovCardBorad x2;
x3=load('Nov2002FoamBlock');
save NovFoamBlock x3;
x4=load('Nov2002Plant');
save NovPlant x4;
x5=load('Nov2002BrickWall');
save NovBrickWall x5;
x6=load('Nov2002IvyOnBrickWall');
save NovIvyOnBrickWall x6;
x7=load('Nov2002Block');
save NovBlock x7;
x8=load('Nov2002Chair');
save NovChair x8;
x9=load('Nov2002ChairWithBag');
save NovChairWithBag x9;
x10=load('Nov2002ThinCardBoard');
save NovThinCardBoard x10;

```

%The following are the test signals.

```

y1=load('Nov2002AITest');
save NovAITest y1;
y2=load('Nov2002AITest2');
save NovAITest2 y2;
y3=load('Nov2002AITest3');
save NovAITest3 y3;
y4=load('Nov2002AITest4');
save NovAITest4 y4;
y5=load('Nov2002AITest5');
save NovAITest5 y5;
y6=load('Nov2002AITest6');
save NovAITest6 y6;
y7=load('Nov2002AITest7');
save NovAITest7 y7;
y8=load('Nov2002AITest8');
save NovAITest8 y8;

```

```

for i=1:400

```

```

    x1_echo(i)=x1(i+1100);
    x2_echo(i)=x2(i+1100);
    x3_echo(i)=x3(i+1085);
    x4_echo(i)=x4(i+700);

```

```
x5_echo(i)=x5(i+800);  
x6_echo(i)=x6(i+880);  
x7_echo(i)=x7(i+1230);  
x8_echo(i)=x8(i+570);  
x9_echo(i)=x9(i+610);  
x10_echo(i)=x10(i+775);  
end
```

```
f1=fft(x1_echo);  
f2=fft(x2_echo);  
f3=fft(x3_echo);  
f4=fft(x4_echo);  
f5=fft(x5_echo);  
f6=fft(x6_echo);  
f7=fft(x7_echo);  
f8=fft(x8_echo);  
f9=fft(x9_echo);  
f10=fft(x10_echo);
```

```
for i=1:5  
    f1(i)=0;  
    f2(i)=0;  
    f3(i)=0;  
    f4(i)=0;  
    f5(i)=0;  
    f6(i)=0;  
    f7(i)=0;  
    f8(i)=0;  
    f9(i)=0;  
    f10(i)=0;  
end
```

```
for i=396:400  
    f1(i)=0;  
    f2(i)=0;  
    f3(i)=0;  
    f4(i)=0;  
    f5(i)=0;  
    f6(i)=0;  
    f7(i)=0;  
    f8(i)=0;  
    f9(i)=0;  
    f10(i)=0;
```

```

end
x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
x3_echo_improve=ifft(f3);
x4_echo_improve=ifft(f4);
x5_echo_improve=ifft(f5);
x6_echo_improve=ifft(f6);
x7_echo_improve=ifft(f7);
x8_echo_improve=ifft(f8);
x9_echo_improve=ifft(f9);
x10_echo_improve=ifft(f10);

```

% The following program calculate the envelop of both the reference echo and the echoes of interesting material.

```

x1_echo_improveshape2=envelope(x1_echo_improve);
x2_echo_improveshape2=envelope(x2_echo_improve);
x3_echo_improveshape2=envelope(x3_echo_improve);
x4_echo_improveshape2=envelope(x4_echo_improve);
x5_echo_improveshape2=envelope(x5_echo_improve);
x6_echo_improveshape2=envelope(x6_echo_improve);
x7_echo_improveshape2=envelope(x7_echo_improve);
x8_echo_improveshape2=envelope(x8_echo_improve);
x9_echo_improveshape2=envelope(x9_echo_improve);
x10_echo_improveshape2=envelope(x10_echo_improve);

for i=1:10
error100_1(i)=matchfilter(y1,x1_echo_improveshape2,10*i+680);
error100_2(i)=matchfilter(y1,x2_echo_improveshape2,10*i+680);
error100_3(i)=matchfilter(y1,x3_echo_improveshape2,10*i+680);
error100_4(i)=matchfilter(y1,x4_echo_improveshape2,10*i+680);
error100_5(i)=matchfilter(y1,x5_echo_improveshape2,10*i+680);
error100_6(i)=matchfilter(y1,x6_echo_improveshape2,10*i+680);
error100_7(i)=matchfilter(y1,x7_echo_improveshape2,10*i+680);
error100_8(i)=matchfilter(y1,x8_echo_improveshape2,10*i+680);
error100_9(i)=matchfilter(y1,x9_echo_improveshape2,10*i+680);
error100_10(i)=matchfilter(y1,x10_echo_improveshape2,10*i+680);
end

for i=1:10
error102_1(i)=matchfilter(y2,x1_echo_improveshape2,10*i+1050);
error102_2(i)=matchfilter(y2,x2_echo_improveshape2,10*i+1050);
error102_3(i)=matchfilter(y2,x3_echo_improveshape2,10*i+1050);
error102_4(i)=matchfilter(y2,x4_echo_improveshape2,10*i+1050);

```

```

error102_5(i)=matchfilter(y2,x5_echo_improveshape2,10*i+1050);
error102_6(i)=matchfilter(y2,x6_echo_improveshape2,10*i+1050);
error102_7(i)=matchfilter(y2,x7_echo_improveshape2,10*i+1050);
error102_8(i)=matchfilter(y2,x8_echo_improveshape2,10*i+1050);
error102_9(i)=matchfilter(y2,x9_echo_improveshape2,10*i+1050);
error102_10(i)=matchfilter(y2,x10_echo_improveshape2,10*i+1050);
end

```

```

for i=1:10
error103_1(i)=matchfilter(y3,x1_echo_improveshape2,10*i+1060);
error103_2(i)=matchfilter(y3,x2_echo_improveshape2,10*i+1060);
error103_3(i)=matchfilter(y3,x3_echo_improveshape2,10*i+1060);
error103_4(i)=matchfilter(y3,x4_echo_improveshape2,10*i+1060);
error103_5(i)=matchfilter(y3,x5_echo_improveshape2,10*i+1060);
error103_6(i)=matchfilter(y3,x6_echo_improveshape2,10*i+1060);
error103_7(i)=matchfilter(y3,x7_echo_improveshape2,10*i+1060);
error103_8(i)=matchfilter(y3,x8_echo_improveshape2,10*i+1060);
error103_9(i)=matchfilter(y3,x9_echo_improveshape2,10*i+1060);
error103_10(i)=matchfilter(y3,x10_echo_improveshape2,10*i+1060);
end

```

```

for i=1:10
error104_1(i)=matchfilter(y4,x1_echo_improveshape2,10*i+1060);
error104_2(i)=matchfilter(y4,x2_echo_improveshape2,10*i+1060);
error104_3(i)=matchfilter(y4,x3_echo_improveshape2,10*i+1060);
error104_4(i)=matchfilter(y4,x4_echo_improveshape2,10*i+1060);
error104_5(i)=matchfilter(y4,x5_echo_improveshape2,10*i+1060);
error104_6(i)=matchfilter(y4,x6_echo_improveshape2,10*i+1060);
error104_7(i)=matchfilter(y4,x7_echo_improveshape2,10*i+1060);
error104_8(i)=matchfilter(y4,x8_echo_improveshape2,10*i+1060);
error104_9(i)=matchfilter(y4,x9_echo_improveshape2,10*i+1060);
error104_10(i)=matchfilter(y4,x10_echo_improveshape2,10*i+1060);
end

```

```

for i=1:10
error105_1(i)=matchfilter(y5,x1_echo_improveshape2,10*i+1060);
error105_2(i)=matchfilter(y5,x2_echo_improveshape2,10*i+1060);
error105_3(i)=matchfilter(y5,x3_echo_improveshape2,10*i+1060);
error105_4(i)=matchfilter(y5,x4_echo_improveshape2,10*i+1060);
error105_5(i)=matchfilter(y5,x5_echo_improveshape2,10*i+1060);
error105_6(i)=matchfilter(y5,x6_echo_improveshape2,10*i+1060);
error105_7(i)=matchfilter(y5,x7_echo_improveshape2,10*i+1060);
error105_8(i)=matchfilter(y5,x8_echo_improveshape2,10*i+1060);
error105_9(i)=matchfilter(y5,x9_echo_improveshape2,10*i+1060);
error105_10(i)=matchfilter(y5,x10_echo_improveshape2,10*i+1060);
end

```


end

for i=1:10

```

error106_1(i)=matchfilter(y6,x1_echo_improveshape2,10*i+960);
error106_2(i)=matchfilter(y6,x2_echo_improveshape2,10*i+960);
error106_3(i)=matchfilter(y6,x3_echo_improveshape2,10*i+960);
error106_4(i)=matchfilter(y6,x4_echo_improveshape2,10*i+960);
error106_5(i)=matchfilter(y6,x5_echo_improveshape2,10*i+960);
error106_6(i)=matchfilter(y6,x6_echo_improveshape2,10*i+960);
error106_7(i)=matchfilter(y6,x7_echo_improveshape2,10*i+960);
error106_8(i)=matchfilter(y6,x8_echo_improveshape2,10*i+960);
error106_9(i)=matchfilter(y6,x9_echo_improveshape2,10*i+960);
error106_10(i)=matchfilter(y6,x10_echo_improveshape2,10*i+960);
end

```

for i=1:10

```

error107_1(i)=matchfilter(y7,x1_echo_improveshape2,10*i+960);
error107_2(i)=matchfilter(y7,x2_echo_improveshape2,10*i+960);
error107_3(i)=matchfilter(y7,x3_echo_improveshape2,10*i+960);
error107_4(i)=matchfilter(y7,x4_echo_improveshape2,10*i+960);
error107_5(i)=matchfilter(y7,x5_echo_improveshape2,10*i+960);
error107_6(i)=matchfilter(y7,x6_echo_improveshape2,10*i+960);
error107_7(i)=matchfilter(y7,x7_echo_improveshape2,10*i+960);
error107_8(i)=matchfilter(y7,x8_echo_improveshape2,10*i+960);
error107_9(i)=matchfilter(y7,x9_echo_improveshape2,10*i+960);
error107_10(i)=matchfilter(y7,x10_echo_improveshape2,10*i+960);
end

```

for i=1:10

```

error108_1(i)=matchfilter(y8,x1_echo_improveshape2,10*i+680);
error108_2(i)=matchfilter(y8,x2_echo_improveshape2,10*i+680);
error108_3(i)=matchfilter(y8,x3_echo_improveshape2,10*i+680);
error108_4(i)=matchfilter(y8,x4_echo_improveshape2,10*i+680);
error108_5(i)=matchfilter(y8,x5_echo_improveshape2,10*i+680);
error108_6(i)=matchfilter(y8,x6_echo_improveshape2,10*i+680);
error108_7(i)=matchfilter(y8,x7_echo_improveshape2,10*i+680);
error108_8(i)=matchfilter(y8,x8_echo_improveshape2,10*i+680);
error108_9(i)=matchfilter(y8,x9_echo_improveshape2,10*i+680);
error108_10(i)=matchfilter(y8,x10_echo_improveshape2,10*i+680);
end

```

```

error1min=min(error100_1);
error2min=min(error100_2);
error3min=min(error100_3);
error4min=min(error100_4);

```

```
error5min=min(error100_5);  
error6min=min(error100_6);  
error7min=min(error100_7);  
error8min=min(error100_8);  
error9min=min(error100_9);  
error10min=min(error100_10);
```

```
error1min_2=min(error102_1);  
error2min_2=min(error102_2);  
error3min_2=min(error102_3);  
error4min_2=min(error102_4);  
error5min_2=min(error102_5);  
error6min_2=min(error102_6);  
error7min_2=min(error102_7);  
error8min_2=min(error102_8);  
error9min_2=min(error102_9);  
error10min_2=min(error102_10);
```

```
error1min_3=min(error103_1);  
error2min_3=min(error103_2);  
error3min_3=min(error103_3);  
error4min_3=min(error103_4);  
error5min_3=min(error103_5);  
error6min_3=min(error103_6);  
error7min_3=min(error103_7);  
error8min_3=min(error103_8);  
error9min_3=min(error103_9);  
error10min_3=min(error103_10);
```

```
error1min_4=min(error104_1);  
error2min_4=min(error104_2);  
error3min_4=min(error104_3);  
error4min_4=min(error104_4);  
error5min_4=min(error104_5);  
error6min_4=min(error104_6);  
error7min_4=min(error104_7);  
error8min_4=min(error104_8);  
error9min_4=min(error104_9);  
error10min_4=min(error104_10);
```

```
error1min_5=min(error105_1);  
error2min_5=min(error105_2);  
error3min_5=min(error105_3);  
error4min_5=min(error105_4);  
error5min_5=min(error105_5);  
error6min_5=min(error105_6);
```

```

error7min_5=min(error105_7);
error8min_5=min(error105_8);
error9min_5=min(error105_9);
error10min_5=min(error105_10);

```

```

error1min_6=min(error106_1);
error2min_6=min(error106_2);
error3min_6=min(error106_3);
error4min_6=min(error106_4);
error5min_6=min(error106_5);
error6min_6=min(error106_6);
error7min_6=min(error106_7);
error8min_6=min(error106_8);
error9min_6=min(error106_9);
error10min_6=min(error106_10);

```

```

error1min_7=min(error107_1);
error2min_7=min(error107_2);
error3min_7=min(error107_3);
error4min_7=min(error107_4);
error5min_7=min(error107_5);
error6min_7=min(error107_6);
error7min_7=min(error107_7);
error8min_7=min(error107_8);
error9min_7=min(error107_9);
error10min_7=min(error107_10);

```

```

error1min_8=min(error108_1);
error2min_8=min(error108_2);
error3min_8=min(error108_3);
error4min_8=min(error108_4);
error5min_8=min(error108_5);
error6min_8=min(error108_6);
error7min_8=min(error108_7);
error8min_8=min(error108_8);
error9min_8=min(error108_9);
error10min_8=min(error108_10);

```

```

%hold off
n=10:10:100;
%plot(n,error100_1,'b')
%xlabel('shifting number of points'); ylabel('error')
%title('Echo from Aluminum Plate')
%hold on
%plot(n,error100_2,'ob')
%hold on

```

```

%plot(n,error100_3,'xg')
%hold on
%plot(n,error100_4,'+g')
%hold on
%plot(n,error100_5,'*r')
%hold on
%plot(n,error100_6,'sr')
%hold on
%plot(n,error100_7,'dm')
%hold on
%plot(n,error100_8,'vm')
%hold on
%plot(n,error100_9,'^k')
%hold on
%plot(n,error100_10,'<k')
%hold on
%legend('Al','Cardboard','Foamblock','Plant','Brickwall','Ivy on
brickwall','Block','Chair','Chairwithbag','Thin cardboard')
%plot(n,error100_1,n,error100_2,n,error100_3,n,error100_4,n,error100_5,n,error100_6,n
,error100_7,n,error100_8,n,error100_9,n,error100_10);
%hold off

%hold off
n=10:10:100;
%plot(n,error108_1,'.b')
%xlabel('shifting number of points'); ylabel('error')
%title('Echo from Aluminum Plate')
%hold on
%plot(n,error108_2,'ob')
%hold on
%plot(n,error108_3,'xg')
%hold on
%plot(n,error108_4,'+g')
%hold on
%plot(n,error108_5,'*r')
%hold on
%plot(n,error108_6,'sr')
%hold on
%plot(n,error108_7,'dm')
%hold on
%plot(n,error108_8,'vm')
%hold on
%plot(n,error108_9,'^k')
%hold on
%plot(n,error108_10,'<k')
%hold on

```

```
%legend('Al','Cardboard','Foamblock','Plant','Brickwall','Ivy on
brickwall','Block','Chair','Chairwithbag','Thin cardboard')
%plot(n,error108_1,n,error108_2,n,error108_3,n,error108_4,n,error108_5,n,error108_6,n
,error108_7,n,error108_8,n,error108_9,n,error108_10);
%hold off
```

```
cof1_1=polyfit(n,error100_1,2);
cof2_1=polyfit(n,error100_2,2);
cof3_1=polyfit(n,error100_3,2);
cof4_1=polyfit(n,error100_4,2);
cof5_1=polyfit(n,error100_5,2);
cof6_1=polyfit(n,error100_6,2);
cof7_1=polyfit(n,error100_7,2);
cof8_1=polyfit(n,error100_8,2);
cof9_1=polyfit(n,error100_9,2);
cof10_1=polyfit(n,error100_10,2);
ylocation1_1=(4*cof1_1(1)*cof1_1(3)-cof1_1(2)*cof1_1(2))/(4*cof1_1(1));
ylocation2_1=(4*cof2_1(1)*cof2_1(3)-cof2_1(2)*cof2_1(2))/(4*cof2_1(1));
ylocation3_1=(4*cof3_1(1)*cof3_1(3)-cof3_1(2)*cof3_1(2))/(4*cof3_1(1));
ylocation4_1=(4*cof4_1(1)*cof4_1(3)-cof4_1(2)*cof4_1(2))/(4*cof4_1(1));
ylocation5_1=(4*cof5_1(1)*cof5_1(3)-cof5_1(2)*cof5_1(2))/(4*cof5_1(1));
ylocation6_1=(4*cof6_1(1)*cof6_1(3)-cof6_1(2)*cof6_1(2))/(4*cof6_1(1));
ylocation7_1=(4*cof7_1(1)*cof7_1(3)-cof7_1(2)*cof7_1(2))/(4*cof7_1(1));
ylocation8_1=(4*cof8_1(1)*cof8_1(3)-cof8_1(2)*cof8_1(2))/(4*cof8_1(1));
ylocation9_1=(4*cof9_1(1)*cof9_1(3)-cof9_1(2)*cof9_1(2))/(4*cof9_1(1));
ylocation10_1=(4*cof10_1(1)*cof10_1(3)-cof10_1(2)*cof10_1(2))/(4*cof10_1(1));
```

```
cof1_2=polyfit(n,error102_1,2);
cof2_2=polyfit(n,error102_2,2);
cof3_2=polyfit(n,error102_3,2);
cof4_2=polyfit(n,error102_4,2);
cof5_2=polyfit(n,error102_5,2);
cof6_2=polyfit(n,error102_6,2);
cof7_2=polyfit(n,error102_7,2);
cof8_2=polyfit(n,error102_8,2);
cof9_2=polyfit(n,error102_9,2);
cof10_2=polyfit(n,error102_10,2);
ylocation1_2=(4*cof1_2(1)*cof1_2(3)-cof1_2(2)*cof1_2(2))/(4*cof1_2(1));
ylocation2_2=(4*cof2_2(1)*cof2_2(3)-cof2_2(2)*cof2_2(2))/(4*cof2_2(1));
ylocation3_2=(4*cof3_2(1)*cof3_2(3)-cof3_2(2)*cof3_2(2))/(4*cof3_2(1));
ylocation4_2=(4*cof4_2(1)*cof4_2(3)-cof4_2(2)*cof4_2(2))/(4*cof4_2(1));
ylocation5_2=(4*cof5_2(1)*cof5_2(3)-cof5_2(2)*cof5_2(2))/(4*cof5_2(1));
ylocation6_2=(4*cof6_2(1)*cof6_2(3)-cof6_2(2)*cof6_2(2))/(4*cof6_2(1));
ylocation7_2=(4*cof7_2(1)*cof7_2(3)-cof7_2(2)*cof7_2(2))/(4*cof7_2(1));
ylocation8_2=(4*cof8_2(1)*cof8_2(3)-cof8_2(2)*cof8_2(2))/(4*cof8_2(1));
ylocation9_2=(4*cof9_2(1)*cof9_2(3)-cof9_2(2)*cof9_2(2))/(4*cof9_2(1));
```

$ylocation10_2=(4*cof10_2(1)*cof10_2(3)-cof10_2(2)*cof10_2(2))/(4*cof10_2(1));$

cof1_3=polyfit(n,error103_1,2);
 cof2_3=polyfit(n,error103_2,2);
 cof3_3=polyfit(n,error103_3,2);
 cof4_3=polyfit(n,error103_4,2);
 cof5_3=polyfit(n,error103_5,2);
 cof6_3=polyfit(n,error103_6,2);
 cof7_3=polyfit(n,error103_7,2);
 cof8_3=polyfit(n,error103_8,2);
 cof9_3=polyfit(n,error103_9,2);
 cof10_3=polyfit(n,error103_10,2);

$ylocation1_3=(4*cof1_3(1)*cof1_3(3)-cof1_3(2)*cof1_3(2))/(4*cof1_3(1));$
 $ylocation2_3=(4*cof2_3(1)*cof2_3(3)-cof2_3(2)*cof2_3(2))/(4*cof2_3(1));$
 $ylocation3_3=(4*cof3_3(1)*cof3_3(3)-cof3_3(2)*cof3_3(2))/(4*cof3_3(1));$
 $ylocation4_3=(4*cof4_3(1)*cof4_3(3)-cof4_3(2)*cof4_3(2))/(4*cof4_3(1));$
 $ylocation5_3=(4*cof5_3(1)*cof5_3(3)-cof5_3(2)*cof5_3(2))/(4*cof5_3(1));$
 $ylocation6_3=(4*cof6_3(1)*cof6_3(3)-cof6_3(2)*cof6_3(2))/(4*cof6_3(1));$
 $ylocation7_3=(4*cof7_3(1)*cof7_3(3)-cof7_3(2)*cof7_3(2))/(4*cof7_3(1));$
 $ylocation8_3=(4*cof8_3(1)*cof8_3(3)-cof8_3(2)*cof8_3(2))/(4*cof8_3(1));$
 $ylocation9_3=(4*cof9_3(1)*cof9_3(3)-cof9_3(2)*cof9_3(2))/(4*cof9_3(1));$
 $ylocation10_3=(4*cof10_3(1)*cof10_3(3)-cof10_3(2)*cof10_3(2))/(4*cof10_3(1));$

cof1_4=polyfit(n,error104_1,2);
 cof2_4=polyfit(n,error104_2,2);
 cof3_4=polyfit(n,error104_3,2);
 cof4_4=polyfit(n,error104_4,2);
 cof5_4=polyfit(n,error104_5,2);
 cof6_4=polyfit(n,error104_6,2);
 cof7_4=polyfit(n,error104_7,2);
 cof8_4=polyfit(n,error104_8,2);
 cof9_4=polyfit(n,error104_9,2);
 cof10_4=polyfit(n,error104_10,2);

$ylocation1_4=(4*cof1_4(1)*cof1_4(3)-cof1_4(2)*cof1_4(2))/(4*cof1_4(1));$
 $ylocation2_4=(4*cof2_4(1)*cof2_4(3)-cof2_4(2)*cof2_4(2))/(4*cof2_4(1));$
 $ylocation3_4=(4*cof3_4(1)*cof3_4(3)-cof3_4(2)*cof3_4(2))/(4*cof3_4(1));$
 $ylocation4_4=(4*cof4_4(1)*cof4_4(3)-cof4_4(2)*cof4_4(2))/(4*cof4_4(1));$
 $ylocation5_4=(4*cof5_4(1)*cof5_4(3)-cof5_4(2)*cof5_4(2))/(4*cof5_4(1));$
 $ylocation6_4=(4*cof6_4(1)*cof6_4(3)-cof6_4(2)*cof6_4(2))/(4*cof6_4(1));$
 $ylocation7_4=(4*cof7_4(1)*cof7_4(3)-cof7_4(2)*cof7_4(2))/(4*cof7_4(1));$
 $ylocation8_4=(4*cof8_4(1)*cof8_4(3)-cof8_4(2)*cof8_4(2))/(4*cof8_4(1));$
 $ylocation9_4=(4*cof9_4(1)*cof9_4(3)-cof9_4(2)*cof9_4(2))/(4*cof9_4(1));$
 $ylocation10_4=(4*cof10_4(1)*cof10_4(3)-cof10_4(2)*cof10_4(2))/(4*cof10_4(1));$

cof1_5=polyfit(n,error105_1,2);
 cof2_5=polyfit(n,error105_2,2);

```

cof3_5=polyfit(n,error105_3,2);
cof4_5=polyfit(n,error105_4,2);
cof5_5=polyfit(n,error105_5,2);
cof6_5=polyfit(n,error105_6,2);
cof7_5=polyfit(n,error105_7,2);
cof8_5=polyfit(n,error105_8,2);
cof9_5=polyfit(n,error105_9,2);
cof10_5=polyfit(n,error105_10,2);
ylocation1_5=(4*cof1_5(1)*cof1_5(3)-cof1_5(2)*cof1_5(2))/(4*cof1_5(1));
ylocation2_5=(4*cof2_5(1)*cof2_5(3)-cof2_5(2)*cof2_5(2))/(4*cof2_5(1));
ylocation3_5=(4*cof3_5(1)*cof3_5(3)-cof3_5(2)*cof3_5(2))/(4*cof3_5(1));
ylocation4_5=(4*cof4_5(1)*cof4_5(3)-cof4_5(2)*cof4_5(2))/(4*cof4_5(1));
ylocation5_5=(4*cof5_5(1)*cof5_5(3)-cof5_5(2)*cof5_5(2))/(4*cof5_5(1));
ylocation6_5=(4*cof6_5(1)*cof6_5(3)-cof6_5(2)*cof6_5(2))/(4*cof6_5(1));
ylocation7_5=(4*cof7_5(1)*cof7_5(3)-cof7_5(2)*cof7_5(2))/(4*cof7_5(1));
ylocation8_5=(4*cof8_5(1)*cof8_5(3)-cof8_5(2)*cof8_5(2))/(4*cof8_5(1));
ylocation9_5=(4*cof9_5(1)*cof9_5(3)-cof9_5(2)*cof9_5(2))/(4*cof9_5(1));
ylocation10_5=(4*cof10_5(1)*cof10_5(3)-cof10_5(2)*cof10_5(2))/(4*cof10_5(1));

```

```

cof1_6=polyfit(n,error106_1,2);
cof2_6=polyfit(n,error106_2,2);
cof3_6=polyfit(n,error106_3,2);
cof4_6=polyfit(n,error106_4,2);
cof5_6=polyfit(n,error106_5,2);
cof6_6=polyfit(n,error106_6,2);
cof7_6=polyfit(n,error106_7,2);
cof8_6=polyfit(n,error106_8,2);
cof9_6=polyfit(n,error106_9,2);
cof10_6=polyfit(n,error106_10,2);
ylocation1_6=(4*cof1_6(1)*cof1_6(3)-cof1_6(2)*cof1_6(2))/(4*cof1_6(1));
ylocation2_6=(4*cof2_6(1)*cof2_6(3)-cof2_6(2)*cof2_6(2))/(4*cof2_6(1));
ylocation3_6=(4*cof3_6(1)*cof3_6(3)-cof3_6(2)*cof3_6(2))/(4*cof3_6(1));
ylocation4_6=(4*cof4_6(1)*cof4_6(3)-cof4_6(2)*cof4_6(2))/(4*cof4_6(1));
ylocation5_6=(4*cof5_6(1)*cof5_6(3)-cof5_6(2)*cof5_6(2))/(4*cof5_6(1));
ylocation6_6=(4*cof6_6(1)*cof6_6(3)-cof6_6(2)*cof6_6(2))/(4*cof6_6(1));
ylocation7_6=(4*cof7_6(1)*cof7_6(3)-cof7_6(2)*cof7_6(2))/(4*cof7_6(1));
ylocation8_6=(4*cof8_6(1)*cof8_6(3)-cof8_6(2)*cof8_6(2))/(4*cof8_6(1));
ylocation9_6=(4*cof9_6(1)*cof9_6(3)-cof9_6(2)*cof9_6(2))/(4*cof9_6(1));
ylocation10_6=(4*cof10_6(1)*cof10_6(3)-cof10_6(2)*cof10_6(2))/(4*cof10_6(1));

```

```

cof1_7=polyfit(n,error107_1,2);
cof2_7=polyfit(n,error107_2,2);
cof3_7=polyfit(n,error107_3,2);
cof4_7=polyfit(n,error107_4,2);
cof5_7=polyfit(n,error107_5,2);
cof6_7=polyfit(n,error107_6,2);

```

```

cof7_7=polyfit(n,error107_7,2);
cof8_7=polyfit(n,error107_8,2);
cof9_7=polyfit(n,error107_9,2);
cof10_7=polyfit(n,error107_10,2);
ylocation1_7=(4*cof1_7(1)*cof1_7(3)-cof1_7(2)*cof1_7(2))/(4*cof1_7(1));
ylocation2_7=(4*cof2_7(1)*cof2_7(3)-cof2_7(2)*cof2_7(2))/(4*cof2_7(1));
ylocation3_7=(4*cof3_7(1)*cof3_7(3)-cof3_7(2)*cof3_7(2))/(4*cof3_7(1));
ylocation4_7=(4*cof4_7(1)*cof4_7(3)-cof4_7(2)*cof4_7(2))/(4*cof4_7(1));
ylocation5_7=(4*cof5_7(1)*cof5_7(3)-cof5_7(2)*cof5_7(2))/(4*cof5_7(1));
ylocation6_7=(4*cof6_7(1)*cof6_7(3)-cof6_7(2)*cof6_7(2))/(4*cof6_7(1));
ylocation7_7=(4*cof7_7(1)*cof7_7(3)-cof7_7(2)*cof7_7(2))/(4*cof7_7(1));
ylocation8_7=(4*cof8_7(1)*cof8_7(3)-cof8_7(2)*cof8_7(2))/(4*cof8_7(1));
ylocation9_7=(4*cof9_7(1)*cof9_7(3)-cof9_7(2)*cof9_7(2))/(4*cof9_7(1));
ylocation10_7=(4*cof10_7(1)*cof10_7(3)-cof10_7(2)*cof10_7(2))/(4*cof10_7(1));

```

```

cof1_8=polyfit(n,error108_1,2);
cof2_8=polyfit(n,error108_2,2);
cof3_8=polyfit(n,error108_3,2);
cof4_8=polyfit(n,error108_4,2);
cof5_8=polyfit(n,error108_5,2);
cof6_8=polyfit(n,error108_6,2);
cof7_8=polyfit(n,error108_7,2);
cof8_8=polyfit(n,error108_8,2);
cof9_8=polyfit(n,error108_9,2);
cof10_8=polyfit(n,error108_10,2);
ylocation1_8=(4*cof1_8(1)*cof1_8(3)-cof1_8(2)*cof1_8(2))/(4*cof1_8(1));
ylocation2_8=(4*cof2_8(1)*cof2_8(3)-cof2_8(2)*cof2_8(2))/(4*cof2_8(1));
ylocation3_8=(4*cof3_8(1)*cof3_8(3)-cof3_8(2)*cof3_8(2))/(4*cof3_8(1));
ylocation4_8=(4*cof4_8(1)*cof4_8(3)-cof4_8(2)*cof4_8(2))/(4*cof4_8(1));
ylocation5_8=(4*cof5_8(1)*cof5_8(3)-cof5_8(2)*cof5_8(2))/(4*cof5_8(1));
ylocation6_8=(4*cof6_8(1)*cof6_8(3)-cof6_8(2)*cof6_8(2))/(4*cof6_8(1));
ylocation7_8=(4*cof7_8(1)*cof7_8(3)-cof7_8(2)*cof7_8(2))/(4*cof7_8(1));
ylocation8_8=(4*cof8_8(1)*cof8_8(3)-cof8_8(2)*cof8_8(2))/(4*cof8_8(1));
ylocation9_8=(4*cof9_8(1)*cof9_8(3)-cof9_8(2)*cof9_8(2))/(4*cof9_8(1));
ylocation10_8=(4*cof10_8(1)*cof10_8(3)-cof10_8(2)*cof10_8(2))/(4*cof10_8(1));

```

```

hold off
plot(1/cof1_1(1),ylocation1_1,'k');
xlabel('1/a of the function y=ax^2+bx+c'); ylabel('Ymin');
%title('Echo from Aluminum Plate')
axis([-0.05 0.2 0 100000]);
hold on
plot(1/cof2_1(1),ylocation2_1,'ok');
hold on
plot(1/cof3_1(1),ylocation3_1,'xk');
hold on

```



```

plot(1/cof4_1(1),ylocation4_1,'+k');
hold on
plot(1/cof5_1(1),ylocation5_1,'*k');
hold on
plot(1/cof6_1(1),ylocation6_1,'sk');
hold on
plot(1/cof7_1(1),ylocation7_1,'dk');
hold on
plot(1/cof8_1(1),ylocation8_1,'vk');
hold on
plot(1/cof9_1(1),ylocation9_1,'^k');
hold on
plot(1/cof10_1(1),ylocation10_1,'<k');
%legend('Al','Cardboard','Foamblock','Plant','Brickwall','Ivy on
brickwall','Block','Chair','Chairwithbag','Thin cardboard')

```

```

hold on
plot(1/cof1_2(1),ylocation1_2,'.k');
hold on
plot(1/cof2_2(1),ylocation2_2,'ok');
hold on
plot(1/cof3_2(1),ylocation3_2,'xk');
hold on
plot(1/cof4_2(1),ylocation4_2,'+k');
hold on
plot(1/cof5_2(1),ylocation5_2,'*k');
hold on
plot(1/cof6_2(1),ylocation6_2,'sk');
hold on
plot(1/cof7_2(1),ylocation7_2,'dk');
hold on
plot(1/cof8_2(1),ylocation8_2,'vk');
hold on
plot(1/cof9_2(1),ylocation9_2,'^k');
hold on
plot(1/cof10_2(1),ylocation10_2,'<k');

```

```

hold on
plot(1/cof1_3(1),ylocation1_3,'.k');
hold on
plot(1/cof2_3(1),ylocation2_3,'ok');
hold on
plot(1/cof3_3(1),ylocation3_3,'xk');
hold on
plot(1/cof4_3(1),ylocation4_3,'+k');
hold on

```

```

plot(1/cof5_3(1),ylocation5_3,'*k');
hold on
plot(1/cof6_3(1),ylocation6_3,'sk');
hold on
plot(1/cof7_3(1),ylocation7_3,'dk');
hold on
plot(1/cof8_3(1),ylocation8_3,'vk');
hold on
plot(1/cof9_3(1),ylocation9_3,'^k');
hold on
plot(1/cof10_3(1),ylocation10_3,'<k');

```

```

hold on
plot(1/cof1_4(1),ylocation1_4,'.k');
hold on
plot(1/cof2_4(1),ylocation2_4,'ok');
hold on
plot(1/cof3_4(1),ylocation3_4,'xk');
hold on
plot(1/cof4_4(1),ylocation4_4,'+k');
hold on
plot(1/cof5_4(1),ylocation5_4,'*k');
hold on
plot(1/cof6_4(1),ylocation6_4,'sk');
hold on
plot(1/cof7_4(1),ylocation7_4,'dk');
hold on
plot(1/cof8_4(1),ylocation8_4,'vk');
hold on
plot(1/cof9_4(1),ylocation9_4,'^k');
hold on
plot(1/cof10_4(1),ylocation10_4,'<k');

```

```

hold on
plot(1/cof1_5(1),ylocation1_5,'.k');
hold on
plot(1/cof2_5(1),ylocation2_5,'ok');
hold on
plot(1/cof3_5(1),ylocation3_5,'xk');
hold on
plot(1/cof4_5(1),ylocation4_5,'+k');
hold on
plot(1/cof5_5(1),ylocation5_5,'*k');
hold on
plot(1/cof6_5(1),ylocation6_5,'sk');
hold on

```

```

plot(1/cof7_5(1),ylocation7_5,'dk');
hold on
plot(1/cof8_5(1),ylocation8_5,'vk');
hold on
plot(1/cof9_5(1),ylocation9_5,'^k');
hold on
plot(1/cof10_5(1),ylocation10_5,'<k');

```

```

hold on
plot(1/cof1_6(1),ylocation1_6,'.k');
hold on
plot(1/cof2_6(1),ylocation2_6,'ok');
hold on
plot(1/cof3_6(1),ylocation3_6,'xk');
hold on
plot(1/cof4_6(1),ylocation4_6,'+k');
hold on
plot(1/cof5_6(1),ylocation5_6,'*k');
hold on
plot(1/cof6_6(1),ylocation6_6,'sk');
hold on
plot(1/cof7_6(1),ylocation7_6,'dk');
hold on
plot(1/cof8_6(1),ylocation8_6,'vk');
hold on
plot(1/cof9_6(1),ylocation9_6,'^k');
hold on
plot(1/cof10_6(1),ylocation10_6,'<k');

```

```

hold on
plot(1/cof1_7(1),ylocation1_7,'.k');
hold on
plot(1/cof2_7(1),ylocation2_7,'ok');
hold on
plot(1/cof3_7(1),ylocation3_7,'xk');
hold on
plot(1/cof4_7(1),ylocation4_7,'+k');
hold on
plot(1/cof5_7(1),ylocation5_7,'*k');
hold on
plot(1/cof6_7(1),ylocation6_7,'sk');
hold on
plot(1/cof7_7(1),ylocation7_7,'dk');
hold on
plot(1/cof8_7(1),ylocation8_7,'vk');
hold on

```

```

plot(1/cof9_7(1),ylocation9_7,'^k');
hold on
plot(1/cof10_7(1),ylocation10_7,'<k');

hold on
plot(1/cof1_8(1),ylocation1_8,'k');
hold on
plot(1/cof2_8(1),ylocation2_8,'ok');
hold on
plot(1/cof3_8(1),ylocation3_8,'xk');
hold on
plot(1/cof4_8(1),ylocation4_8,'+k');
hold on
plot(1/cof5_8(1),ylocation5_8,'*k');
hold on
plot(1/cof6_8(1),ylocation6_8,'sk');
hold on
plot(1/cof7_8(1),ylocation7_8,'dk');
hold on
plot(1/cof8_8(1),ylocation8_8,'vk');
hold on
plot(1/cof9_8(1),ylocation9_8,'^k');
hold on
plot(1/cof10_8(1),ylocation10_8,'<k');
hold off

x1_average=mean([1/cof1_1(1), 1/cof1_2(1), 1/cof1_3(1), 1/cof1_4(1), 1/cof1_5(1),
1/cof1_6(1), 1/cof1_7(1), 1/cof1_8(1)]);
y1_average=mean([ylocation1_1/10000, ylocation1_2/10000, ylocation1_3/10000,
ylocation1_4/10000, ylocation1_5/10000, ylocation1_6/10000, ylocation1_7/10000,
ylocation1_8/10000]);
R1=sqrt(x1_average^2+y1_average^2);

x2_average=mean([1/cof2_1(1), 1/cof2_2(1), 1/cof2_3(1), 1/cof2_4(1), 1/cof2_5(1),
1/cof2_6(1), 1/cof2_7(1), 1/cof2_8(1)]);
y2_average=mean([ylocation2_1/10000, ylocation2_2/10000, ylocation2_3/10000,
ylocation2_4/10000, ylocation2_5/10000, ylocation2_6/10000, ylocation2_7/10000,
ylocation2_8/10000]);
R2=sqrt(x2_average^2+y2_average^2);

x3_average=mean([1/cof3_1(1), 1/cof3_2(1), 1/cof3_3(1), 1/cof3_4(1), 1/cof3_5(1),
1/cof3_6(1), 1/cof3_7(1), 1/cof3_8(1)]);
y3_average=mean([ylocation3_1/10000, ylocation3_2/10000, ylocation3_3/10000,
ylocation3_4/10000, ylocation3_5/10000, ylocation3_6/10000, ylocation3_7/10000,
ylocation3_8/10000]);

```

R3=sqrt(x3_average^2+y3_average^2);

x4_average=mean([1/cof4_1(1), 1/cof4_2(1), 1/cof4_3(1), 1/cof4_4(1), 1/cof4_5(1),
1/cof4_6(1), 1/cof4_7(1), 1/cof4_8(1)]);

y4_average=mean([ylocation4_1/10000, ylocation4_2/10000, ylocation4_3/10000,
ylocation4_4/10000, ylocation4_5/10000, ylocation4_6/10000, ylocation4_7/10000,
ylocation4_8/10000]);

R4=sqrt(x4_average^2+y4_average^2);

x5_average=mean([1/cof5_1(1), 1/cof5_2(1), 1/cof5_3(1), 1/cof5_4(1), 1/cof5_5(1),
1/cof5_6(1), 1/cof5_7(1), 1/cof5_8(1)]);

y5_average=mean([ylocation5_1/10000, ylocation5_2/10000, ylocation5_3/10000,
ylocation5_4/10000, ylocation5_5/10000, ylocation5_6/10000, ylocation5_7/10000,
ylocation5_8/10000]);

R5=sqrt(x5_average^2+y5_average^2);

x6_average=mean([1/cof6_1(1), 1/cof6_2(1), 1/cof6_3(1), 1/cof6_4(1), 1/cof6_5(1),
1/cof6_6(1), 1/cof6_7(1), 1/cof6_8(1)]);

y6_average=mean([ylocation6_1/10000, ylocation6_2/10000, ylocation6_3/10000,
ylocation6_4/10000, ylocation6_5/10000, ylocation6_6/10000, ylocation6_7/10000,
ylocation6_8/10000]);

R6=sqrt(x6_average^2+y6_average^2);

x7_average=mean([1/cof7_1(1), 1/cof7_2(1), 1/cof7_3(1), 1/cof7_4(1), 1/cof7_5(1),
1/cof7_6(1), 1/cof7_7(1), 1/cof7_8(1)]);

y7_average=mean([ylocation7_1/10000, ylocation7_2/10000, ylocation7_3/10000,
ylocation7_4/10000, ylocation7_5/10000, ylocation7_6/10000, ylocation7_7/10000,
ylocation7_8/10000]);

R7=sqrt(x7_average^2+y7_average^2);

x8_average=mean([1/cof8_1(1), 1/cof8_2(1), 1/cof8_3(1), 1/cof8_4(1), 1/cof8_5(1),
1/cof8_6(1), 1/cof8_7(1), 1/cof8_8(1)]);

y8_average=mean([ylocation8_1/10000, ylocation8_2/10000, ylocation8_3/10000,
ylocation8_4/10000, ylocation8_5/10000, ylocation8_6/10000, ylocation8_7/10000,
ylocation8_8/10000]);

R8=sqrt(x8_average^2+y8_average^2);

x9_average=mean([1/cof9_1(1), 1/cof9_2(1), 1/cof9_3(1), 1/cof9_4(1), 1/cof9_5(1),
1/cof9_6(1), 1/cof9_7(1), 1/cof9_8(1)]);

```

y9_average=mean([ylocation9_1/10000, ylocation9_2/10000, ylocation9_3/10000,
ylocation9_4/10000, ylocation9_5/10000, ylocation9_6/10000, ylocation9_7/10000,
ylocation9_8/10000]);
R9=sqrt(x9_average^2+y9_average^2);

```

```

x10_average=mean([1/cof10_1(1), 1/cof10_2(1), 1/cof10_3(1), 1/cof10_4(1),
1/cof10_5(1), 1/cof10_6(1), 1/cof10_7(1), 1/cof10_8(1)]);
y10_average=mean([ylocation10_1/10000, ylocation10_2/10000, ylocation10_3/10000,
ylocation10_4/10000, ylocation10_5/10000, ylocation10_6/10000, ylocation10_7/10000,
ylocation10_8/10000]);
R10=sqrt(x10_average^2+y10_average^2);

```

```

R=[R1 R2 R3 R4 R5 R6 R7 R8 R9 R10];
ssR=sort(R);

```

```

switch ssR(1)
case R1
    fprintf('The first choice is Aluminum plate.\n');
case R2
    fprintf('The first choice is cardboard.\n');
case R3
    fprintf('The first choice is foamblock.\n');
case R4
    fprintf('The first choice is plant.\n');
case R5
    fprintf('The first choice is brick wall.\n');
case R6
    fprintf('The first choice is ivy on brick wall.\n');
case R7
    fprintf('The first choice is block.\n');
case R8
    fprintf('The first choice is chair.\n');
case R9
    fprintf('The first choice is chair with bag.\n');
case R10
    fprintf('The first choice is thin card board.\n');
otherwise
end

```

```

switch ssR(2)
case R1
    fprintf('The second choice is Aluminum plate.\n');
case R2

```

```

    fprintf('The second choice is cardboard.\n');
case R3
    fprintf('The second choice is foamblock.\n');
case R4
    fprintf('The second choice is plant.\n');
case R5
    fprintf('The second choice is brick wall.\n');
case R6
    fprintf('The second choice is ivy on brick wall.\n');
case R7
    fprintf('The second choice is block.\n');
case R8
    fprintf('The second choice is chair.\n');
case R9
    fprintf('The second choice is chair with bag.\n');
case R10
    fprintf('The second choice is thin card board.\n');
otherwise
end

```

```

switch ssR(3)
case R1
    fprintf('The third choice is Aluminum plate.\n');
case R2
    fprintf('The third choice is cardboard.\n');
case R3
    fprintf('The third choice is foamblock.\n');
case R4
    fprintf('The third choice is plant.\n');
case R5
    fprintf('The third choice is brick wall.\n');
case R6
    fprintf('The third choice is ivy on brick wall.\n');
case R7
    fprintf('The third choice is block.\n');
case R8
    fprintf('The third choice is chair.\n');
case R9
    fprintf('The third choice is chair with bag.\n');
case R10
    fprintf('The third choice is thin card board.\n');
otherwise
end

```

```

switch ssR(4)

```

```

case R1
    fprintf('The fourth choice is Aluminum plate.\n');
case R2
    fprintf('The fourth choice is cardboard.\n');
case R3
    fprintf('The fourth choice is foamblock.\n');
case R4
    fprintf('The fourth choice is plant.\n');
case R5
    fprintf('The fourth choice is brick wall.\n');
case R6
    fprintf('The fourth choice is ivy on brick wall.\n');
case R7
    fprintf('The fourth choice is block.\n');
case R8
    fprintf('The fourth choice is chair.\n');
case R9
    fprintf('The fourth choice is chair with bag.\n');
case R10
    fprintf('The fourth choice is thin card board.\n');
otherwise
end

```

```

switch ssR(5)
case R1
    fprintf('The fifth choice is Aluminum plate.\n');
case R2
    fprintf('The fifth choice is cardboard.\n');
case R3
    fprintf('The fifth choice is foamblock.\n');
case R4
    fprintf('The fifth choice is plant.\n');
case R5
    fprintf('The fifth choice is brick wall.\n');
case R6
    fprintf('The fifth choice is ivy on brick wall.\n');
case R7
    fprintf('The fifth choice is block.\n');
case R8
    fprintf('The fifth choice is chair.\n');
case R9
    fprintf('The fifth choice is chair with bag.\n');
case R10
    fprintf('The fifth choice is thin card board.\n');
otherwise
end

```


Appendix C

Matlab Code for Analysis of Outside Cylindrical Objects

Spread2.m

```

clear;
LP3_P9S1=FWHM_LP3_100cm;
LP3_P9S2=FWHM_LP3_125cm;
LP3_P9S3=FWHM_LP3_150cm;
LP3_P9S4=FWHM_LP3_175cm;
LP3_P9S5=FWHM_LP3_200cm;
LP3_P9S6=FWHM_LP3_225cm;
LP3_P9S7=FWHM_LP3_250cm;
LP3_P9S8=FWHM_LP3_275cm;
LP3_P9S9=FWHM_LP3_300cm;

x_LamppostNo3=[LP3_P9S1(1), LP3_P9S2(1), LP3_P9S3(1), LP3_P9S4(1),
LP3_P9S5(1), LP3_P9S6(1), LP3_P9S7(1), LP3_P9S8(1), LP3_P9S9(1)];
y_LamppostNo3=[LP3_P9S1(2), LP3_P9S2(2), LP3_P9S3(2), LP3_P9S4(2),
LP3_P9S5(2), LP3_P9S6(2), LP3_P9S7(2), LP3_P9S8(2), LP3_P9S9(2)];
avgx_LamppostNo3=(sum(x_LamppostNo3))/length(x_LamppostNo3);
avgy_LamppostNo3=(sum(y_LamppostNo3))/length(y_LamppostNo3);
spd_LamppostNo3=(x_LamppostNo3(1)-avgx_LamppostNo3).^2+(y_LamppostNo3(1)-
avgy_LamppostNo3).^2+(x_LamppostNo3(2)-
avgx_LamppostNo3).^2+(y_LamppostNo3(2)-
avgy_LamppostNo3).^2+(x_LamppostNo3(3)-
avgx_LamppostNo3).^2+(y_LamppostNo3(3)-
avgy_LamppostNo3).^2+(x_LamppostNo3(4)-
avgx_LamppostNo3).^2+(y_LamppostNo3(4)-
avgy_LamppostNo3).^2+(x_LamppostNo3(5)-
avgx_LamppostNo3).^2+(y_LamppostNo3(5)-
avgy_LamppostNo3).^2+(x_LamppostNo3(6)-
avgx_LamppostNo3).^2+(y_LamppostNo3(6)-
avgy_LamppostNo3).^2+(x_LamppostNo3(7)-
avgx_LamppostNo3).^2+(y_LamppostNo3(7)-
avgy_LamppostNo3).^2+(x_LamppostNo3(8)-
avgx_LamppostNo3).^2+(y_LamppostNo3(8)-
avgy_LamppostNo3).^2+(x_LamppostNo3(9)-
avgx_LamppostNo3).^2+(y_LamppostNo3(9)-avgy_LamppostNo3).^2;
spd_LamppostNo3=spd_LamppostNo3/length(x_LamppostNo3);

LP4_P10S1=FWHM_LP4_100cm;
LP4_P10S2=FWHM_LP4_110cm;
LP4_P10S3=FWHM_LP4_120cm;

```

LP4_P10S4=FWHM_LP4_130cm;
 LP4_P10S5=FWHM_LP4_140cm;
 LP4_P10S6=FWHM_LP4_150cm;
 LP4_P10S7=FWHM_LP4_160cm;
 LP4_P10S8=FWHM_LP4_170cm;
 LP4_P10S9=FWHM_LP4_180cm;
 LP4_P10S10=FWHM_LP4_190cm;

x_LamppostNo4=[LP4_P10S1(1), LP4_P10S2(1), LP4_P10S3(1), LP4_P10S4(1),
 LP4_P10S5(1), LP4_P10S6(1), LP4_P10S7(1), LP4_P10S8(1), LP4_P10S9(1),
 LP4_P10S10(1)];
 y_LamppostNo4=[LP4_P10S1(2), LP4_P10S2(2), LP4_P10S3(2), LP4_P10S4(2),
 LP4_P10S5(2), LP4_P10S6(2), LP4_P10S7(2), LP4_P10S8(2), LP4_P10S9(2),
 LP4_P10S10(2)];

avgx_LamppostNo4=(sum(x_LamppostNo4))/length(x_LamppostNo4);
 avgy_LamppostNo4=(sum(y_LamppostNo4))/length(y_LamppostNo4);
 spd_LamppostNo4=(x_LamppostNo4(1)-avgx_LamppostNo4).^2+(y_LamppostNo4(1)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(2)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(2)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(3)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(3)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(4)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(4)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(5)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(5)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(6)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(6)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(7)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(7)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(8)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(8)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(9)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(9)-
 avgy_LamppostNo4).^2+(x_LamppostNo4(10)-
 avgx_LamppostNo4).^2+(y_LamppostNo4(10)-avgy_LamppostNo4).^2;
 spd_LamppostNo4=spd_LamppostNo4/length(x_LamppostNo4);

T18_T18S1=FWHM_T18_100cm;
 T18_T18S2=FWHM_T18_120cm;
 T18_T18S3=FWHM_T18_140cm;
 T18_T18S4=FWHM_T18_150cm;
 T18_T18S5=FWHM_T18_160cm;
 T18_T18S6=FWHM_T18_170cm;
 T18_T18S7=FWHM_T18_180cm;
 T18_T18S8=FWHM_T18_190cm;

T18_T18S9=FWHM_T18_200cm;
 T18_T18S10=FWHM_T18_220cm;

x_TreeNo18=[T18_T18S1(1),
 T18_T18S2(1),T18_T18S3(1),T18_T18S4(1),T18_T18S5(1),T18_T18S6(1),T18_T18S7
 (1),T18_T18S8(1),T18_T18S9(1),T18_T18S10(1)];
 y_TreeNo18=[T18_T18S1(2),
 T18_T18S2(2),T18_T18S3(2),T18_T18S4(2),T18_T18S5(2),T18_T18S6(2),T18_T18S7
 (2),T18_T18S8(2),T18_T18S9(2),T18_T18S10(2)];
 avgx_TreeNo18=(sum(x_TreeNo18))/length(x_TreeNo18);
 avgy_TreeNo18=(sum(y_TreeNo18))/length(y_TreeNo18);
 spd_TreeNo18=(x_TreeNo18(1)-avgx_TreeNo18).^2+(y_TreeNo18(1)-
 avgy_TreeNo18).^2+(x_TreeNo18(2)-avgx_TreeNo18).^2+(y_TreeNo18(2)-
 avgy_TreeNo18).^2+(x_TreeNo18(3)-avgx_TreeNo18).^2+(y_TreeNo18(3)-
 avgy_TreeNo18).^2+(x_TreeNo18(4)-avgx_TreeNo18).^2+(y_TreeNo18(4)-
 avgy_TreeNo18).^2+(x_TreeNo18(5)-avgx_TreeNo18).^2+(y_TreeNo18(5)-
 avgy_TreeNo18).^2+(x_TreeNo18(6)-avgx_TreeNo18).^2+(y_TreeNo18(6)-
 avgy_TreeNo18).^2+(x_TreeNo18(7)-avgx_TreeNo18).^2+(y_TreeNo18(7)-
 avgy_TreeNo18).^2+(x_TreeNo18(8)-avgx_TreeNo18).^2+(y_TreeNo18(8)-
 avgy_TreeNo18).^2+(x_TreeNo18(9)-avgx_TreeNo18).^2+(y_TreeNo18(9)-
 avgy_TreeNo18).^2+(x_TreeNo18(10)-avgx_TreeNo18).^2+(y_TreeNo18(10)-
 avgy_TreeNo18).^2;
 spd_TreeNo18=spd_TreeNo18/length(x_TreeNo18);

T19_T19S1=FWHM_T19_90cm;
 T19_T19S2=FWHM_T19_100cm;
 T19_T19S3=FWHM_T19_110cm;
 T19_T19S4=FWHM_T19_120cm;
 T19_T19S5=FWHM_T19_130cm;
 T19_T19S6=FWHM_T19_140cm;
 T19_T19S7=FWHM_T19_150cm;
 T19_T19S8=FWHM_T19_160cm;
 T19_T19S9=FWHM_T19_170cm;
 T19_T19S10=FWHM_T19_180cm;

x_TreeNo19=[T19_T19S1(1),
 T19_T19S2(1),T19_T19S3(1),T19_T19S4(1),T19_T19S5(1),T19_T19S6(1),T19_T19S7
 (1),T19_T19S8(1),T19_T19S9(1),T19_T19S10(1)];
 y_TreeNo19=[T19_T19S1(2),
 T19_T19S2(2),T19_T19S3(2),T19_T19S4(2),T19_T19S5(2),T19_T19S6(2),T19_T19S7
 (2),T19_T19S8(2),T19_T19S9(2),T19_T19S10(2)];
 avgx_TreeNo19=(sum(x_TreeNo19))/length(x_TreeNo19);
 avgy_TreeNo19=(sum(y_TreeNo19))/length(y_TreeNo19);
 spd_TreeNo19=(x_TreeNo19(1)-avgx_TreeNo19).^2+(y_TreeNo19(1)-
 avgy_TreeNo19).^2+(x_TreeNo19(2)-avgx_TreeNo19).^2+(y_TreeNo19(2)-
 avgy_TreeNo19).^2+(x_TreeNo19(3)-avgx_TreeNo19).^2+(y_TreeNo19(3)-

$\text{avg}_y_TreeNo19.^2+(x_TreeNo19(4)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(4)-\text{avg}_y_TreeNo19).^2+(x_TreeNo19(5)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(5)-\text{avg}_y_TreeNo19).^2+(x_TreeNo19(6)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(6)-\text{avg}_y_TreeNo19).^2+(x_TreeNo19(7)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(7)-\text{avg}_y_TreeNo19).^2+(x_TreeNo19(8)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(8)-\text{avg}_y_TreeNo19).^2+(x_TreeNo19(9)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(9)-\text{avg}_y_TreeNo19).^2+(x_TreeNo19(10)-\text{avg}_x_TreeNo19).^2+(y_TreeNo19(10)-\text{avg}_y_TreeNo19).^2;$
 $\text{spd_TreeNo19}=\text{spd_TreeNo19}/\text{length}(x_TreeNo19);$

$T20_T20S1=\text{FWHM_T20_100cm};$
 $T20_T20S2=\text{FWHM_T20_110cm};$
 $T20_T20S3=\text{FWHM_T20_120cm};$
 $T20_T20S4=\text{FWHM_T20_140cm};$
 $T20_T20S5=\text{FWHM_T20_150cm};$
 $T20_T20S6=\text{FWHM_T20_160cm};$
 $T20_T20S7=\text{FWHM_T20_165cm};$
 $T20_T20S8=\text{FWHM_T20_170cm};$
 $T20_T20S9=\text{FWHM_T20_180cm};$
 $T20_T20S10=\text{FWHM_T20_210cm};$

$x_TreeNo20=[T20_T20S1(1),$
 $T20_T20S2(1),T20_T20S3(1),T20_T20S4(1),T20_T20S5(1),T20_T20S6(1),T20_T20S7$
 $(1),T20_T20S8(1),T20_T20S9(1),T20_T20S10(1)];$
 $y_TreeNo20=[T20_T20S1(2),$
 $T20_T20S2(2),T20_T20S3(2),T20_T20S4(2),T20_T20S5(2),T20_T20S6(2),T20_T20S7$
 $(2),T20_T20S8(2),T20_T20S9(2),T20_T20S10(2)];$
 $\text{avg}_x_TreeNo20=(\text{sum}(x_TreeNo20))/\text{length}(x_TreeNo20);$
 $\text{avg}_y_TreeNo20=(\text{sum}(y_TreeNo20))/\text{length}(y_TreeNo20);$
 $\text{spd_TreeNo20}=(x_TreeNo20(1)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(1)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(2)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(2)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(3)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(3)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(4)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(4)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(5)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(5)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(6)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(6)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(7)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(7)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(8)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(8)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(9)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(9)-\text{avg}_y_TreeNo20).^2+(x_TreeNo20(10)-\text{avg}_x_TreeNo20).^2+(y_TreeNo20(10)-\text{avg}_y_TreeNo20).^2;$
 $\text{spd_TreeNo20}=\text{spd_TreeNo20}/\text{length}(x_TreeNo20);$

```

x_EmergencyPillar=[0.0278,0.0137,0.0313,0.0571,0.0313,0.0556,0.0423,0.0141,0.0857];
y_EmergencyPillar=[1.1777,0.7479,0.6408,1.1328,0.3078,1.0239,1.2087,0.8710,1.4099]/
16;
avgx_EmergencyPillar=(sum(x_EmergencyPillar))/length(x_EmergencyPillar);
avgy_EmergencyPillar=(sum(y_EmergencyPillar))/length(y_EmergencyPillar);
spd_EmergencyPillar=(x_EmergencyPillar(1)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(1)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(2)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(2)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(3)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(3)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(4)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(4)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(5)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(5)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(6)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(6)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(7)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(7)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(8)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(8)-
avgy_EmergencyPillar).^2+(x_EmergencyPillar(9)-
avgx_EmergencyPillar).^2+(y_EmergencyPillar(9)-avgy_EmergencyPillar).^2;
spd_EmergencyPillar=spd_EmergencyPillar/length(x_EmergencyPillar);

plot(avgx_LamppostNo3,spd_LamppostNo3,'o');
xlabel('Average Asymmetry'); ylabel('Average Squared Euclidean Distance');
%axis([0 0.1 0 0.02]);
hold on
plot(avgx_TreeNo18,spd_TreeNo18,'*');
hold on
plot(avgx_TreeNo19,spd_TreeNo19,'*');
hold on
plot(avgx_LamppostNo4,spd_LamppostNo4,'o');
hold on
plot(avgx_TreeNo20,spd_TreeNo20,'*');
hold on

x_TreeNo21=[0.2,0.3051,0.1868,0.2165,0.0714,0.0588,0.0805,0.0270,0.0606,0.0556];
y_TreeNo21=[0.8349,0.7233,0.9171,0.6393,1.1961,1.3137,1.4429,1.9381,4.3017,1.4790
]/16;
avgx_TreeNo21=(sum(x_TreeNo21))/length(x_TreeNo21);

```

```

avg_y_TreeNo21=(sum(y_TreeNo21))/length(y_TreeNo21);
spd_TreeNo21=(x_TreeNo21(1)-avg_x_TreeNo21).^2+(y_TreeNo21(1)-
avg_y_TreeNo21).^2+(x_TreeNo21(2)-avg_x_TreeNo21).^2+(y_TreeNo21(2)-
avg_y_TreeNo21).^2+(x_TreeNo21(3)-avg_x_TreeNo21).^2+(y_TreeNo21(3)-
avg_y_TreeNo21).^2+(x_TreeNo21(4)-avg_x_TreeNo21).^2+(y_TreeNo21(4)-
avg_y_TreeNo21).^2+(x_TreeNo21(5)-avg_x_TreeNo21).^2+(y_TreeNo21(5)-
avg_y_TreeNo21).^2+(x_TreeNo21(6)-avg_x_TreeNo21).^2+(y_TreeNo21(6)-
avg_y_TreeNo21).^2+(x_TreeNo21(7)-avg_x_TreeNo21).^2+(y_TreeNo21(7)-
avg_y_TreeNo21).^2+(x_TreeNo21(8)-avg_x_TreeNo21).^2+(y_TreeNo21(8)-
avg_y_TreeNo21).^2+(x_TreeNo21(9)-avg_x_TreeNo21).^2+(y_TreeNo21(9)-
avg_y_TreeNo21).^2+(x_TreeNo21(10)-avg_x_TreeNo21).^2+(y_TreeNo21(10)-
avg_y_TreeNo21).^2;
spd_TreeNo21=spd_TreeNo21/length(x_TreeNo21);

```

```

x_LamppostNo1=[0.0137,0.0476,0.0435,0.0286,0.0333,0.0313,0.0645,0.0606,0.1515];
y_LamppostNo1=[0.7831,0.7414,0.9882,0.6835,0.7466,1.0504,0.8389,0.7389,0.7997]/16
;
avg_x_LamppostNo1=(sum(x_LamppostNo1))/length(x_LamppostNo1);
avg_y_LamppostNo1=(sum(y_LamppostNo1))/length(y_LamppostNo1);
spd_LamppostNo1=(x_LamppostNo1(1)-avg_x_LamppostNo1).^2+(y_LamppostNo1(1)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(2)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(2)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(3)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(3)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(4)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(4)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(5)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(5)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(6)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(6)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(7)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(7)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(8)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(8)-
avg_y_LamppostNo1).^2+(x_LamppostNo1(9)-
avg_x_LamppostNo1).^2+(y_LamppostNo1(9)-avg_y_LamppostNo1).^2;
spd_LamppostNo1=spd_LamppostNo1/length(x_LamppostNo1);

```

```

x_TreeNo22=[0.0704,0.0145,0.0606,0.0571,0.0556,0.3704,0.2439,0.0303,0.3200,0.6316
];
y_TreeNo22=[0.8384,1.3736,1.1638,1.2983,0.6182,1.9251,1.7044,1.586,2.3558,3.5657]/
16;
avg_x_TreeNo22=(sum(x_TreeNo22))/length(x_TreeNo22);
avg_y_TreeNo22=(sum(y_TreeNo22))/length(y_TreeNo22);
spd_TreeNo22=(x_TreeNo22(1)-avg_x_TreeNo22).^2+(y_TreeNo22(1)-
avg_y_TreeNo22).^2+(x_TreeNo22(2)-avg_x_TreeNo22).^2+(y_TreeNo22(2)-

```

```

avgx_TreeNo22=(sum(x_TreeNo22))/length(x_TreeNo22);
avgy_TreeNo22=(sum(y_TreeNo22))/length(y_TreeNo22);
spd_TreeNo22=(x_TreeNo22(1)-avgx_TreeNo22).^2+(y_TreeNo22(1)-
avgy_TreeNo22).^2+(x_TreeNo22(2)-avgx_TreeNo22).^2+(y_TreeNo22(2)-
avgy_TreeNo22).^2+(x_TreeNo22(3)-avgx_TreeNo22).^2+(y_TreeNo22(3)-
avgy_TreeNo22).^2+(x_TreeNo22(4)-avgx_TreeNo22).^2+(y_TreeNo22(4)-
avgy_TreeNo22).^2+(x_TreeNo22(5)-avgx_TreeNo22).^2+(y_TreeNo22(5)-
avgy_TreeNo22).^2+(x_TreeNo22(6)-avgx_TreeNo22).^2+(y_TreeNo22(6)-
avgy_TreeNo22).^2+(x_TreeNo22(7)-avgx_TreeNo22).^2+(y_TreeNo22(7)-
avgy_TreeNo22).^2+(x_TreeNo22(8)-avgx_TreeNo22).^2+(y_TreeNo22(8)-
avgy_TreeNo22).^2+(x_TreeNo22(9)-avgx_TreeNo22).^2+(y_TreeNo22(9)-
avgy_TreeNo22).^2+(x_TreeNo22(10)-avgx_TreeNo22).^2+(y_TreeNo22(10)-
avgy_TreeNo22).^2;
spd_TreeNo22=spd_TreeNo22/length(x_TreeNo22);

```

```

x_LamppostNo2=[0,0.0411,0.0625,0.0137,0.0725,0.0667,0.0746,0.0769,0.0435,0.0789];
y_LamppostNo2=[0.3088,0.5373,0.7433,0.9251,0.8115,1.3573,1.0979,1.4527,1.7951,1.7
195]/16;

```

```

avgx_LamppostNo2=(sum(x_LamppostNo2))/length(x_LamppostNo2);
avgy_LamppostNo2=(sum(y_LamppostNo2))/length(y_LamppostNo2);
spd_LamppostNo2=(x_LamppostNo2(1)-avgx_LamppostNo2).^2+(y_LamppostNo2(1)-
avgy_LamppostNo2).^2+(x_LamppostNo2(2)-avgx_LamppostNo2).^2+(y_LamppostNo2(2)-
avgy_LamppostNo2).^2+(x_LamppostNo2(3)-avgx_LamppostNo2).^2+(y_LamppostNo2(3)-
avgy_LamppostNo2).^2+(x_LamppostNo2(4)-avgx_LamppostNo2).^2+(y_LamppostNo2(4)-
avgy_LamppostNo2).^2+(x_LamppostNo2(5)-avgx_LamppostNo2).^2+(y_LamppostNo2(5)-
avgy_LamppostNo2).^2+(x_LamppostNo2(6)-avgx_LamppostNo2).^2+(y_LamppostNo2(6)-
avgy_LamppostNo2).^2+(x_LamppostNo2(7)-avgx_LamppostNo2).^2+(y_LamppostNo2(7)-
avgy_LamppostNo2).^2+(x_LamppostNo2(8)-avgx_LamppostNo2).^2+(y_LamppostNo2(8)-
avgy_LamppostNo2).^2+(x_LamppostNo2(9)-avgx_LamppostNo2).^2+(y_LamppostNo2(9)-
avgy_LamppostNo2).^2;
spd_LamppostNo2=spd_LamppostNo2/length(x_LamppostNo2);

```

```

x_TreeNo8=[1,0,0.0633,0.0278,0.0769,0,1,0.1333,0.0794,1];
y_TreeNo8=[1.0421,1.7912,2.0239,1.6235,1.6365,1.9877,3.3259,1.3782,1.7675,1.3834]/
16;

```

```

avgx_TreeNo8=(sum(x_TreeNo8))/length(x_TreeNo8);
avgy_TreeNo8=(sum(y_TreeNo8))/length(y_TreeNo8);
spd_TreeNo8=(x_TreeNo8(1)-avgx_TreeNo8).^2+(y_TreeNo8(1)-
avgy_TreeNo8).^2+(x_TreeNo8(2)-avgx_TreeNo8).^2+(y_TreeNo8(2)-
avgy_TreeNo8).^2+(x_TreeNo8(3)-avgx_TreeNo8).^2+(y_TreeNo8(3)-
avgy_TreeNo8).^2+(x_TreeNo8(4)-avgx_TreeNo8).^2+(y_TreeNo8(4)-
avgy_TreeNo8).^2+(x_TreeNo8(5)-avgx_TreeNo8).^2+(y_TreeNo8(5)-
avgy_TreeNo8).^2+(x_TreeNo8(6)-avgx_TreeNo8).^2+(y_TreeNo8(6)-
avgy_TreeNo8).^2+(x_TreeNo8(7)-avgx_TreeNo8).^2+(y_TreeNo8(7)-
avgy_TreeNo8).^2+(x_TreeNo8(8)-avgx_TreeNo8).^2+(y_TreeNo8(8)-
avgy_TreeNo8).^2+(x_TreeNo8(9)-avgx_TreeNo8).^2+(y_TreeNo8(9)-
avgy_TreeNo8).^2+(x_TreeNo8(10)-avgx_TreeNo8).^2+(y_TreeNo8(10)-
avgy_TreeNo8).^2;
spd_TreeNo8=spd_TreeNo8/length(x_TreeNo8);

```

```

avg_y_TreeNo8).^2+(x_TreeNo8(5)-avg_x_TreeNo8).^2+(y_TreeNo8(5)-
avg_y_TreeNo8).^2+(x_TreeNo8(6)-avg_x_TreeNo8).^2+(y_TreeNo8(6)-
avg_y_TreeNo8).^2+(x_TreeNo8(7)-avg_x_TreeNo8).^2+(y_TreeNo8(7)-
avg_y_TreeNo8).^2+(x_TreeNo8(8)-avg_x_TreeNo8).^2+(y_TreeNo8(8)-
avg_y_TreeNo8).^2+(x_TreeNo8(9)-avg_x_TreeNo8).^2+(y_TreeNo8(9)-
avg_y_TreeNo8).^2+(x_TreeNo8(10)-avg_x_TreeNo8).^2+(y_TreeNo8(10)-
avg_y_TreeNo8).^2;
spd_TreeNo8=spd_TreeNo8/length(x_TreeNo8);

```

```

% x_Alpipe=[0.4,0.5,0.2,0.2,0.4,0.3,0.3,0.4,0.2,0.5];
% y_Alpipe=[0.7283,0.5971,0.5723,0.7826,0.8638,0.5562,0.6720,0.4871,0.6603,0.4484]/
16;

```

```

x_EmergencyPillar=[0.0278,0.0137,0.0313,0.0571,0.0313,0.0556,0.0423,0.0141,0.0857];
y_EmergencyPillar=[1.1777,0.7479,0.6408,1.1328,0.3078,1.0239,1.2087,0.8710,1.4099]/
16;

```

```

avg_x_EmergencyPillar=(sum(x_EmergencyPillar))/length(x_EmergencyPillar);
avg_y_EmergencyPillar=(sum(y_EmergencyPillar))/length(y_EmergencyPillar);
spd_EmergencyPillar=(x_EmergencyPillar(1)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(1)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(2)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(2)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(3)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(3)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(4)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(4)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(5)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(5)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(6)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(6)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(7)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(7)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(8)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(8)-
avg_y_EmergencyPillar).^2+(x_EmergencyPillar(9)-
avg_x_EmergencyPillar).^2+(y_EmergencyPillar(9)-avg_y_EmergencyPillar).^2;
spd_EmergencyPillar=spd_EmergencyPillar/length(x_EmergencyPillar);

```

```

% x_TreeGroup=[0.0233,0.7838,0.4574,0.7121,0.8058,0.0877,0.1971,0.2727,0.0118,0.06
06,0.0141,0,0.0323,0.0986,0.3884,0.0933,0.1081,0.0843,0.0789,0.0805,1,0.1264,0.0286,
0.0110,0.0588,0.0805];
% y_TreeGroup=[2.1338,1.6757,2.1167,0.8449,0.7758,1.1171,1.7134,0.9673,0.7695,1.00
07,1.1142,1.2144,1.6210,1.6978,1.4545,1.5238,0.9827,1.5169,1.4792,0.6664,1.0515,1.27
82,0.6965,0.8529,1.1237,0.6664]/16;

```



```

x_TreeNo14=[0.0824,0.6324,0.5446,0.6991];
y_TreeNo14=[1.2763,1.8893,0.4383,1.9748]/16;
avgx_TreeNo14=(sum(x_TreeNo14))/length(x_TreeNo14);
avgy_TreeNo14=(sum(y_TreeNo14))/length(y_TreeNo14);
spd_TreeNo14=(x_TreeNo14(1)-avgx_TreeNo14).^2+(y_TreeNo14(1)-
avgy_TreeNo14).^2+(x_TreeNo14(2)-avgx_TreeNo14).^2+(y_TreeNo14(2)-
avgy_TreeNo14).^2+(x_TreeNo14(3)-avgx_TreeNo14).^2+(y_TreeNo14(3)-
avgy_TreeNo14).^2+(x_TreeNo14(4)-avgx_TreeNo14).^2+(y_TreeNo14(4)-
avgy_TreeNo14).^2;
spd_TreeNo14=spd_TreeNo14/length(x_TreeNo14);

plot(avgx_LamppostNo1,spd_LamppostNo1,'o');
%xlabel('Average Asymmetry'); ylabel('Average Squared Euclidean Distance');
%axis([0 0.1 0 0.02]);
hold on
plot(avgx_TreeNo21,spd_TreeNo21,'*');
hold on
plot(avgx_TreeNo22,spd_TreeNo22,'*');
hold on
plot(avgx_LamppostNo2,spd_LamppostNo2,'o');
hold on
plot(avgx_TreeNo8,spd_TreeNo8,'*');
hold on
plot(avgx_EmergencyPillar,spd_EmergencyPillar,'o');
hold on
plot(avgx_TreeNo14,spd_TreeNo14,'*');
hold off

```

FWHM_LP3_100cm.m

```

function asymm=FWHM_LP3_100cm
clear;
x=load('April2004LP3X32Y8_100cm');
%save AprilLP3X32Y8_100cm x;
column=3;
row=4;
total_row=8;
for i=1:2000
    data_Yrow_Xcolumn(i)=x(2000*(total_row*(column-1)+row-1)+i);
end
for i=1:2000
    data_Y4X1(i)=x(2000*(8*0+4-1)+i);
    data_Y4X2(i)=x(2000*(8*1+4-1)+i);
    data_Y4X3(i)=x(2000*(8*2+4-1)+i);

```

```

.....
.....

data_Y4X30(i)=x(2000*(8*29+4-1)+i);
data_Y4X31(i)=x(2000*(8*30+4-1)+i);
data_Y4X32(i)=x(2000*(8*31+4-1)+i);
end

```

```

for i=1:200
  x0_echo(i)=data_Y4X1(550+i);
  x1_echo(i)=data_Y4X2(550+i);
  x2_echo(i)=data_Y4X3(550+i);

```

```

.....
.....

  x29_echo(i)=data_Y4X30(550+i);
  x30_echo(i)=data_Y4X31(550+i);
  x31_echo(i)=data_Y4X32(550+i);
end

```

```

f0=fft(x0_echo);
f1=fft(x1_echo);
f2=fft(x2_echo);
.....
.....

```

```

f29=fft(x29_echo);
f30=fft(x30_echo);
f31=fft(x31_echo);

```

```

for i=1:5
  f0(i)=0;
  f1(i)=0;
  f2(i)=0;
.....
.....

```

```

  f29(i)=0;
  f30(i)=0;
  f31(i)=0;
end

```

```

for i=196:200
  f0(i)=0;
  f1(i)=0;
  f2(i)=0;
.....
.....

```

```

    f29(i)=0;
    f30(i)=0;
    f31(i)=0;
end

x0_echo_improve=ifft(f0);
x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
    .....
    .....

x29_echo_improve=ifft(f29);
x30_echo_improve=ifft(f30);
x31_echo_improve=ifft(f31);

%n=1:200;
%plot(n,x10_echo_improve)

energy(1)=sum(abs(x0_echo_improve));
energy(2)=sum(abs(x1_echo_improve));
energy(3)=sum(abs(x2_echo_improve));
    .....
    .....

energy(30)=sum(abs(x29_echo_improve));
energy(31)=sum(abs(x30_echo_improve));
energy(32)=sum(abs(x31_echo_improve));

%2th fitting and calculating FWHM

%n=1:16;
%cof1=polyfit(n,energy(1:16),2);
%xx=1.1:0.1:16;
%yy=cof1(1)*xx.^2+cof1(2)*xx+cof1(3);
%plot(xx,yy,'-b');
%xlabel('x steps'); ylabel('energy');title('energy distribution for Lamp Poster No.1 at
100cm');
%hold on
%m=1:16;
%plot(m, energy, 'ob')
%hold off
%FWHM=sqrt((cof1(2)^2/cof1(1)^2)-(4*cof1(3)-2*energy(8)-energy(1)-
energy(16))/cof1(1))

%5th order fitting

for i=1:32

```

```

    if energy(i)==max(energy)
        center=i;
    else
    end
end

n=(center-8):(center+8);
cof1=polyfit(n,energy((center-8):(center+8)),5);
xx=(center-8+0.1):0.1:(center+8);
yy=cof1(1)*xx.^5+cof1(2)*xx.^4+cof1(3)*xx.^3+cof1(4)*xx.^2+cof1(5)*xx+cof1(6);
n_new=(-7):8;
xx_new=(-7.9):0.1:8;
plot(xx_new,yy,'-k',n_new,energy((center-7):(center+8)),'ok','linewidth',1.5);
xlabel('scan angle (degrees)'); ylabel('square root of the signal energy');

y_FWHM=0.5*(max(yy)+min(yy));
for i=1:160
    if yy(i)==max(yy)
        n_max=i;
    else
    end
end

for i=1:n_max
    error1(i)=abs(yy(i)-y_FWHM);
    if error1(i)==min(error1)
        n_left=i;
    else
    end
end

for i=n_max:160
    error2(i-n_max+1)=abs(yy(i)-y_FWHM);
    if error2(i-n_max+1)==min(error2)
        n_right=i;
    else
    end
end

FWHM=(n_right-n_left)/10;

asymmetry=abs((n_right-n_max)-(n_max-n_left))/10/FWHM;

dev=0;
for i=1:16
    dev=abs(energy(center-8+i)-yy(10*i))+dev;
end

```

```

end

dev=dev/(max(yy)-min(yy))/16;

asymm=[asymmetry,dev];

```

FWHM_LampPosterNo1_100cm.m

```

function asymm=FWHM_LampPosterNo1_100cm
clear;
x=load('March2003LampPosterNo1_100cmText');
save MarchLampPosterNo1_100cmText x;
column=3;
row=4;
for i=1:2000
    data_Yrow_Xcolumn(i)=x(2000*(8*(column-1)+row-1)+i);
end
for i=1:2000
    data_Y4X1(i)=x(2000*(8*0+4-1)+i);
    data_Y4X2(i)=x(2000*(8*1+4-1)+i);
    data_Y4X3(i)=x(2000*(8*2+4-1)+i);
    data_Y4X4(i)=x(2000*(8*3+4-1)+i);
    data_Y4X5(i)=x(2000*(8*4+4-1)+i);
    data_Y4X6(i)=x(2000*(8*5+4-1)+i);
    data_Y4X7(i)=x(2000*(8*6+4-1)+i);
    data_Y4X8(i)=x(2000*(8*7+4-1)+i);
    data_Y4X9(i)=x(2000*(8*8+4-1)+i);
    data_Y4X10(i)=x(2000*(8*9+4-1)+i);
    data_Y4X11(i)=x(2000*(8*10+4-1)+i);
    data_Y4X12(i)=x(2000*(8*11+4-1)+i);
    data_Y4X13(i)=x(2000*(8*12+4-1)+i);
    data_Y4X14(i)=x(2000*(8*13+4-1)+i);
    data_Y4X15(i)=x(2000*(8*14+4-1)+i);
    data_Y4X16(i)=x(2000*(8*15+4-1)+i);
end

for i=1:200
    x0_echo(i)=data_Y4X1(550+i);
    x1_echo(i)=data_Y4X2(550+i);
    x2_echo(i)=data_Y4X3(550+i);
    x3_echo(i)=data_Y4X4(550+i);
    x4_echo(i)=data_Y4X5(550+i);

```

```
x5_echo(i)=data_Y4X6(550+i);
x6_echo(i)=data_Y4X7(550+i);
x7_echo(i)=data_Y4X8(550+i);
x8_echo(i)=data_Y4X9(550+i);
x9_echo(i)=data_Y4X10(550+i);
x10_echo(i)=data_Y4X11(550+i);
x11_echo(i)=data_Y4X12(550+i);
x12_echo(i)=data_Y4X13(550+i);
x13_echo(i)=data_Y4X14(550+i);
x14_echo(i)=data_Y4X15(550+i);
x15_echo(i)=data_Y4X16(550+i);
end
```

```
f0=fft(x0_echo);
f1=fft(x1_echo);
f2=fft(x2_echo);
f3=fft(x3_echo);
f4=fft(x4_echo);
f5=fft(x5_echo);
f6=fft(x6_echo);
f7=fft(x7_echo);
f8=fft(x8_echo);
f9=fft(x9_echo);
f10=fft(x10_echo);
f11=fft(x11_echo);
f12=fft(x12_echo);
f13=fft(x13_echo);
f14=fft(x14_echo);
f15=fft(x15_echo);
```

```
for i=1:5
    f0(i)=0;
    f1(i)=0;
    f2(i)=0;
    f3(i)=0;
    f4(i)=0;
    f5(i)=0;
    f6(i)=0;
    f7(i)=0;
    f8(i)=0;
    f9(i)=0;
    f10(i)=0;
    f11(i)=0;
```

```
f12(i)=0;
f13(i)=0;
f14(i)=0;
f15(i)=0;
end

for i=196:200
    f0(i)=0;
    f1(i)=0;
    f2(i)=0;
    f3(i)=0;
    f4(i)=0;
    f5(i)=0;
    f6(i)=0;
    f7(i)=0;
    f8(i)=0;
    f9(i)=0;
    f10(i)=0;
    f11(i)=0;
    f12(i)=0;
    f13(i)=0;
    f14(i)=0;
    f15(i)=0;
end

x0_echo_improve=ifft(f0);
x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
x3_echo_improve=ifft(f3);
x4_echo_improve=ifft(f4);
x5_echo_improve=ifft(f5);
x6_echo_improve=ifft(f6);
x7_echo_improve=ifft(f7);
x8_echo_improve=ifft(f8);
x9_echo_improve=ifft(f9);
x10_echo_improve=ifft(f10);
x11_echo_improve=ifft(f11);
x12_echo_improve=ifft(f12);
x13_echo_improve=ifft(f13);
x14_echo_improve=ifft(f14);
x15_echo_improve=ifft(f15);

%n=1:200;
%plot(n,x10_echo_improve)

energy(1)=sum(abs(x0_echo_improve));
```

```

energy(2)=sum(abs(x1_echo_improve));
energy(3)=sum(abs(x2_echo_improve));
energy(4)=sum(abs(x3_echo_improve));
energy(5)=sum(abs(x4_echo_improve));
energy(6)=sum(abs(x5_echo_improve));
energy(7)=sum(abs(x6_echo_improve));
energy(8)=sum(abs(x7_echo_improve));
energy(9)=sum(abs(x8_echo_improve));
energy(10)=sum(abs(x9_echo_improve));
energy(11)=sum(abs(x10_echo_improve));
energy(12)=sum(abs(x11_echo_improve));
energy(13)=sum(abs(x12_echo_improve));
energy(14)=sum(abs(x13_echo_improve));
energy(15)=sum(abs(x14_echo_improve));
energy(16)=sum(abs(x15_echo_improve));

%2th fitting and calculating FWHM

%n=1:16;
%cof1=polyfit(n,energy(1:16),2);
%xx=1.1:0.1:16;
%yy=cof1(1)*xx.^2+cof1(2)*xx+cof1(3);
%plot(xx,yy,'-b');
%xlabel('x steps'); ylabel('energy');title('energy distribution for Lamp Poster No.1 at
100cm');
%hold on
%m=1:16;
%plot(m, energy, 'ob')
%hold off
%FWHM=sqrt((cof1(2)^2/cof1(1)^2)-(4*cof1(3)-2*energy(8)-energy(1)-
energy(16))/cof1(1))

%5th order fitting
n=1:16;
cof1=polyfit(n,energy,5);
xx=0.1:0.1:16;
yy=cof1(1)*xx.^5+cof1(2)*xx.^4+cof1(3)*xx.^3+cof1(4)*xx.^2+cof1(5)*xx+cof1(6);
n_new=(-7):8;
xx_new=(-7.9):0.1:8;
plot(xx_new,yy,'-k',n_new,energy,'ok','linewidth',1.5);
xlabel('scan angle (degrees)'); ylabel('square root of the signal energy');

y_FWHM=0.5*(max(yy)+min(yy));
for i=1:160
    if yy(i)==max(yy)

```



```

        n_max=i;
    else
    end
end

for i=1:n_max
    error1(i)=abs(yy(i)-y_FWHM);
    if error1(i)==min(error1)
        n_left=i;
    else
    end
end

for i=n_max:160
    error2(i-n_max+1)=abs(yy(i)-y_FWHM);
    if error2(i-n_max+1)==min(error2)
        n_right=i;
    else
    end
end

FWHM=(n_right-n_left)/10;

asymmetry=abs((n_right-n_max)-(n_max-n_left))/10/FWHM;

dev=0;
for i=1:16
    dev=abs(energy(i)-yy(10*i))+dev;
end

dev=dev/(max(yy)-min(yy));

asymm=[asymmetry,dev];

```

LPTree.m

```

clear;
x_TreeNo21=[0.2,0.3051,0.1868,0.2165,0.0714,0.0588,0.0805,0.0270,0.0606,0.0556];
y_TreeNo21=[0.8349,0.7233,0.9171,0.6393,1.1961,1.3137,1.4429,1.9381,4.3017,1.4790
]/16;
x_LamppostNo1=[0.0137,0.0476,0.0435,0.0286,0.0333,0.0313,0.0645,0.0606,0.1515];
y_LamppostNo1=[0.7831,0.7414,0.9882,0.6835,0.7466,1.0504,0.8389,0.7389,0.7997]/16
;
x_TreeNo22=[0.0704,0.0145,0.0606,0.0571,0.0556,0.3704,0.2439,0.0303,0.3200,0.6316
];

```

```

y_TreeNo22=[0.8384,1.3736,1.1638,1.2983,0.6182,1.9251,1.7044,1.586,2.3558,3.5657]/
16;
x_TreeNo8=[1,0,0.0633,0.0278,0.0769,0,1,0.1333,0.0794,1];
y_TreeNo8=[1.0421,1.7912,2.0239,1.6235,1.6365,1.9877,3.3259,1.3782,1.7675,1.3834]/
16;
%x_Alpipe=[0.4,0.5,0.2,0.2,0.4,0.3,0.3,0.4,0.2,0.5];
y_Alpipe=[0.7283,0.5971,0.5723,0.7826,0.8638,0.5562,0.6720,0.4871,0.6603,0.4484]/16
;
x_EmergencyPillar=[0.0278,0.0137,0.0313,0.0571,0.0313,0.0556,0.0423,0.0141,0.0857];
y_EmergencyPillar=[1.1777,0.7479,0.6408,1.1328,0.3078,1.0239,1.2087,0.8710,1.4099]/
16;
x_LamppostNo2=[0,0.0411,0.0625,0.0137,0.0725,0.0667,0.0746,0.0769,0.0435,0.0789];
y_LamppostNo2=[0.3088,0.5373,0.7433,0.9251,0.8115,1.3573,1.0979,1.4527,1.7951,1.7
195]/16;
x_TreeGroup=[0.0233,0.7838,0.4574,0.7121,0.8058,0.0877,0.1971,0.2727,0.0118,0.060
6,0.0141,0,0.0323,0.0986,0.3884,0.0933,0.1081,0.0843,0.0789,0.0805,1,0.1264,0.0286,0.
0110,0.0588,0.0805];
y_TreeGroup=[2.1338,1.6757,2.1167,0.8449,0.7758,1.1171,1.7134,0.9673,0.7695,1.000
7,1.1142,1.2144,1.6210,1.6978,1.4545,1.5238,0.9827,1.5169,1.4792,0.6664,1.0515,1.278
2,0.6965,0.8529,1.1237,0.6664]/16;
x_TreeNo14=[0.0824,0.6324,0.5446,0.6991];
y_TreeNo14=[1.2763,1.8893,0.4383,1.9748]/16;

avgx_TreeNo21=(sum(x_TreeNo21))/length(x_TreeNo21);
avgy_TreeNo21=(sum(y_TreeNo21))/length(y_TreeNo21);

plot(x_TreeNo21,y_TreeNo21,'*');
xlabel('Asymmetry'); ylabel('Deviation');
hold on
plot(avgx_TreeNo21,avgy_TreeNo21,'^');
hold off

%plot(x_TreeNo8,y_TreeNo8,'*');
%xlabel('Asymmetry'); ylabel('Deviation');
%hold on
%plot(x_EmergencyPillar,y_EmergencyPillar,'o');
%hold off
%hold on
%plot(x_LamppostNo1,y_LamppostNo1,'o');
%hold off
%plot(x_TreeGroup,y_TreeGroup,'*');
%hold off
%axis([0 12 0 12]);

%plot(x_TreeNo14,y_TreeNo14,'*');
%xlabel('Asymmetry');ylabel('Deviation');

```

```

%if max(x_TreeNo14)<1.5
% fprintf('It is a lamppost. ')
%else
% fprintf('It is a tree. ')
%end

%x_TreeGroup_rough=[0.0233,0.7838,0.4574,0.7121,0.8058,0.0877,0.1971,0.2727,0.38
84,0.0933,1,0.1264,0.0286,0.0110,0.0588,0.0805];
%y_TreeGroup_rough=[2.1338,1.6757,2.1167,0.8449,0.7758,1.1171,1.7134,0.9673,1.45
45,1.5238,1.0515,1.2782,0.6965,0.8529,1.1237,0.6664]/16;

%x_TreeGroup_smooth=[0.0118,0.0606,0.0141,0,0.0323,0.0986,0.1081,0.0843,0.0789,0
.0805];
%y_TreeGroup_smooth=[0.7695,1.0007,1.1142,1.2144,1.6210,1.6978,0.9827,1.5169,1.4
792,0.6664]/16;

%x_PoleGroup=[0.0704,0.0303,0.0476,0.0606,0.0270,0, 0, 0,
0.0476,0.0462,0.0746,0.0606,0.0286,0.0820,0.0278,0.0141,0, 0.0704,0.0435,0.0278];
%y_PoleGroup=[0.0373,0.0489,0.0348,0.0304,0.0498,0.0412,0.0566,0.0705,0.0680,0.07
49,0.0700,0.0542,0.0850,0.0755,0.0565,0.0442,0.0581,0.0818,0.0782,0.0439];
%plot(x_TreeGroup_rough,y_TreeGroup_rough,'*');
%xlabel('Asymmetry'); ylabel('Deviation');
%hold on
%plot(x_TreeGroup_smooth,y_TreeGroup_smooth,'.');
%hold on
%plot(x_PoleGroup, y_PoleGroup,'o');
%hold off
%plot(x_LamppostNo1,y_LamppostNo1,'o');
%hold on
%plot(x_LamppostNo2,y_LamppostNo2,'o');
%hold off

```

Appendix D

Matlab Code for Analysis of Outside Extended Objects

AI.m

```

a1=DE_Fence4InleftEdge_100cmDev;    a1=[1 a1];
a2=DE_Fence4Inleft_100cmDev;        a2=[2 a2];
a3=DE_Fence4Incenter_100cmDev;      a3=[3 a3];
a4=DE_Fence4Inright_100cmDev;       a4=[4 a4];
a5=DE_Fence4InrightEdge_100cmDev;   a5=[5 a5];
a=[a1;a2;a3;a4;a5];

a1_2nd=DE_Fence4InleftEdge_120cmDev;  a1_2nd=[1 a1_2nd];
a2_2nd=DE_Fence4Inleft_120cmDev;      a2_2nd=[2 a2_2nd];
a3_2nd=DE_Fence4Incenter_120cmDev;    a3_2nd=[3 a3_2nd];
a4_2nd=DE_Fence4Inright_120cmDev;     a4_2nd=[4 a4_2nd];
a5_2nd=DE_Fence4InrightEdge_120cmDev; a5_2nd=[5 a5_2nd];
a_2nd=[a1_2nd;a2_2nd;a3_2nd;a4_2nd;a5_2nd];

a1_3rd=DE_Fence4InleftEdge_140cmDev;  a1_3rd=[1 a1_3rd];
a2_3rd=DE_Fence4Inleft_140cmDev;      a2_3rd=[2 a2_3rd];
a3_3rd=DE_Fence4Incenter_140cmDev;    a3_3rd=[3 a3_3rd];
a4_3rd=DE_Fence4Inright_140cmDev;     a4_3rd=[4 a4_3rd];
a5_3rd=DE_Fence4InrightEdge_140cmDev; a5_3rd=[5 a5_3rd];
a_3rd=[a1_3rd;a2_3rd;a3_3rd;a4_3rd;a5_3rd];

b1=DE_BrickwallNo1_nMatchDev;        b1=[1 b1];
b2=DE_BrickwallNo2_nMatchDev;        b2=[2 b2];
b3=DE_BrickwallNo3_nMatchDev;        b3=[3 b3];
b4=DE_BrickwallNo4_nMatchDev;        b4=[4 b4];
b5=DE_BrickwallNo5_nMatchDev;        b5=[5 b5];
b=[b1;b2;b3;b4;b5];

b1_2nd=DE_BrickwallSample2_1Dev;      b1_2nd=[1 b1_2nd];
b2_2nd=DE_BrickwallSample2_2Dev;      b2_2nd=[2 b2_2nd];
b3_2nd=DE_BrickwallSample2_3Dev;      b3_2nd=[3 b3_2nd];
b4_2nd=DE_BrickwallSample2_4Dev;      b4_2nd=[4 b4_2nd];
b5_2nd=DE_BrickwallSample2_5Dev;      b5_2nd=[5 b5_2nd];
b_2nd=[b1_2nd;b2_2nd;b3_2nd;b4_2nd;b5_2nd];

b1_3rd=DE_Brickwall3_1Nov;            b1_3rd=[1 b1_3rd];
b2_3rd=DE_Brickwall3_2Nov;            b2_3rd=[2 b2_3rd];

```

b3_3rd=DE_Brickwall3_3Nov; b3_3rd=[3 b3_3rd];
 b4_3rd=DE_Brickwall3_4Nov; b4_3rd=[4 b4_3rd];
 b5_3rd=DE_Brickwall3_5Nov; b5_3rd=[5 b5_3rd];
 b_3rd=[b1_3rd;b2_3rd;b3_3rd;b4_3rd;b5_3rd];

c1=DE_BushNo1_Dev; c1=[1 c1];
 c2=DE_BushNo2_Dev; c2=[2 c2];
 c3=DE_BushNo3_Dev; c3=[3 c3];
 c4=DE_BushNo4_Dev; c4=[4 c4];
 c5=DE_BushNo5_Dev; c5=[5 c5];
 c=[c1;c2;c3;c4;c5];

c1_2nd=DE_BushSample2_1Dev; c1_2nd=[1 c1_2nd];
 c2_2nd=DE_BushSample2_2Dev; c2_2nd=[2 c2_2nd];
 c3_2nd=DE_BushSample2_3Dev; c3_2nd=[3 c3_2nd];
 c4_2nd=DE_BushSample2_4Dev; c4_2nd=[4 c4_2nd];
 c5_2nd=DE_BushSample2_5Dev; c5_2nd=[5 c5_2nd];
 c_2nd=[c1_2nd;c2_2nd;c3_2nd;c4_2nd;c5_2nd];

c1_3rd=DE_Bush3_1Nov; c1_3rd=[1 c1_3rd];
 c2_3rd=DE_Bush3_2Nov; c2_3rd=[1 c2_3rd];
 c3_3rd=DE_Bush3_3Nov; c3_3rd=[1 c3_3rd];
 c4_3rd=DE_Bush3_4Nov; c4_3rd=[1 c4_3rd];
 c5_3rd=DE_Bush3_5Nov; c5_3rd=[1 c5_3rd];
 c_3rd=[c1_3rd;c2_3rd; c3_3rd; c4_3rd; c5_3rd];

c1_4th=DE_Bush4_1Nov; c1_4th=[1 c1_4th];
 c2_4th=DE_Bush4_2Nov; c2_4th=[1 c2_4th];
 c3_4th=DE_Bush4_3Nov; c3_4th=[1 c3_4th];
 c4_4th=DE_Bush4_4Nov; c4_4th=[1 c4_4th];
 c5_4th=DE_Bush4_5Nov; c5_4th=[1 c5_4th];
 c_4th=[c1_4th;c2_4th; c3_4th; c4_4th; c5_4th];

d1=DE_BwallNo2leftEdge_100cmDev; d1=[1 d1];
 d2=DE_BwallNo2left_100cmDev; d2=[2 d2];
 d3=DE_BwallNo2center_100cmDev; d3=[3 d3];
 d4=DE_BwallNo2right_100cmDev; d4=[4 d4];
 d5=DE_BwallNo2rightEdge_100cmDev; d5=[5 d5];
 d=[d1;d2;d3;d4;d5];

d1_2nd=DE_BwallNo2leftEdge_120cmDev; d1_2nd=[1 d1_2nd];
 d2_2nd=DE_BwallNo2left_120cmDev; d2_2nd=[2 d2_2nd];

d3_2nd=DE_BwallNo2center_120cmDev; d3_2nd=[3 d3_2nd];
d4_2nd=DE_BwallNo2right_120cmDev; d4_2nd=[4 d4_2nd];
d5_2nd=DE_BwallNo2rightEdge_120cmDev; d5_2nd=[5 d5_2nd];
d_2nd=[d1_2nd;d2_2nd;d3_2nd;d4_2nd;d5_2nd];

d1_3rd=DE_BwallNo2leftEdge_140cmDev; d1_3rd=[1 d1_3rd];
d2_3rd=DE_BwallNo2left_140cmDev; d2_3rd=[2 d2_3rd];
d3_3rd=DE_BwallNo2center_140cmDev; d3_3rd=[3 d3_3rd];
d4_3rd=DE_BwallNo2right_140cmDev; d4_3rd=[4 d4_3rd];
d5_3rd=DE_BwallNo2rightEdge_140cmDev; d5_3rd=[5 d5_3rd];
d_3rd=[d1_3rd;d2_3rd;d3_3rd;d4_3rd;d5_3rd];

e1=DE_BwallNo1leftEdge_100cmNov; e1=[1 e1];
e2=DE_BwallNo1left_100cmNov; e2=[2 e2];
e3=DE_BwallNo1center_100cmNov; e3=[3 e3];
e4=DE_BwallNo1right_100cmNov; e4=[4 e4];
e5=DE_BwallNo1rightEdge_100cmNov; e5=[5 e5];
e=[e1;e2;e3;e4;e5];

e1_2nd=DE_BwallNo1leftEdge_120cmNov; e1_2nd=[1 e1_2nd];
e2_2nd=DE_BwallNo1left_120cmNov; e2_2nd=[2 e2_2nd];
e3_2nd=DE_BwallNo1center_120cmNov; e3_2nd=[3 e3_2nd];
e4_2nd=DE_BwallNo1right_120cmNov; e4_2nd=[4 e4_2nd];
e5_2nd=DE_BwallNo1rightEdge_120cmNov; e5_2nd=[5 e5_2nd];
e_2nd=[e1_2nd;e2_2nd;e3_2nd;e4_2nd;e5_2nd];

e1_3rd=DE_BwallNo1leftEdge_140cmNov; e1_3rd=[1 e1_3rd];
e2_3rd=DE_BwallNo1left_140cmNov; e2_3rd=[2 e2_3rd];
e3_3rd=DE_BwallNo1center_140cmNov; e3_3rd=[3 e3_3rd];
e4_3rd=DE_BwallNo1right_140cmNov; e4_3rd=[4 e4_3rd];
e5_3rd=DE_BwallNo1rightEdge_140cmNov; e5_3rd=[5 e5_3rd];
e_3rd=[e1_3rd;e2_3rd;e3_3rd;e4_3rd;e5_3rd];

f1=DE_MFenceleftEdge_100cmDev; f1=[1 f1];
f2=DE_MFenceleft_100cmDev; f2=[2 f2];
f3=DE_MFencecenter_100cmDev; f3=[3 f3];
f4=DE_MFenceright_100cmDev; f4=[4 f4];
f5=DE_MFencerightEdge_100cmDev; f5=[5 f5];
f=[f1;f2;f3;f4;f5];

f1_2nd=DE_MFenceleftEdge_120cmDev; f1_2nd=[1 f1_2nd];
f2_2nd=DE_MFenceleft_120cmDev; f2_2nd=[2 f2_2nd];
f3_2nd=DE_MFencecenter_120cmDev; f3_2nd=[3 f3_2nd];
f4_2nd=DE_MFenceright_120cmDev; f4_2nd=[4 f4_2nd];
f5_2nd=DE_MFencerightEdge_120cmDev; f5_2nd=[5 f5_2nd];
f_2nd=[f1_2nd;f2_2nd;f3_2nd;f4_2nd;f5_2nd];

```

f1_3rd=DE_MFenceleftEdge_140cmDev;      f1_3rd=[1 f1_3rd];
f2_3rd=DE_MFenceleft_140cmDev;          f2_3rd=[2 f2_3rd];
f3_3rd=DE_MFencecenter_140cmDev;        f3_3rd=[3 f3_3rd];
f4_3rd=DE_MFenceright_140cmDev;         f4_3rd=[4 f4_3rd];
f5_3rd=DE_MFencerightEdge_140cmDev;     f5_3rd=[5 f5_3rd];
f_3rd=[f1_3rd;f2_3rd;f3_3rd;f4_3rd;f5_3rd];

```

```

g1=DE_Fence4leftEdge_100cmDev;          g1=[1 g1];
g2=DE_Fence4left_100cmDev;              g2=[2 g2];
g3=DE_Fence4center_100cmDev;            g3=[3 g3];
g4=DE_Fence4right_100cmDev;              g4=[4 g4];
g5=DE_Fence4rightEdge_100cmDev;         g5=[5 g5];
g=[g1;g2;g3;g4;g5];

```

```

g1_2nd=DE_Fence4leftEdge_140cmDev;      g1_2nd=[1 g1_2nd];
g2_2nd=DE_Fence4left_140cmDev;          g2_2nd=[2 g2_2nd];
g3_2nd=DE_Fence4center_140cmDev;        g3_2nd=[3 g3_2nd];
g4_2nd=DE_Fence4right_140cmDev;         g4_2nd=[4 g4_2nd];
g5_2nd=DE_Fence4rightEdge_140cmDev;     g5_2nd=[5 g5_2nd];
g_2nd=[g1_2nd;g2_2nd;g3_2nd;g4_2nd;g5_2nd];

```

```

g1_3rd=DE_Fence4leftEdge100cmNov;       g1_3rd=[1 g1_3rd];
g2_3rd=DE_Fence4left100cmNov;           g2_3rd=[2 g2_3rd];
g3_3rd=DE_Fence4center100cmNov;         g3_3rd=[3 g3_3rd];
g4_3rd=DE_Fence4right100cmNov;          g4_3rd=[4 g4_3rd];
g5_3rd=DE_Fence4rightEdge100cmNov;      g5_3rd=[5 g5_3rd];
g_3rd=[g1_3rd;g2_3rd;g3_3rd;g4_3rd;g5_3rd];

```

```

AA_2nd=[norm(a_2nd(:,2)-a(:,2)) norm(a_2nd(:,3)-a(:,3)));
norm(a_2nd(:,2)-b(:,2)) norm(a_2nd(:,3)-b(:,3)));
norm(a_2nd(:,2)-c(:,2)) norm(a_2nd(:,3)-c(:,3)));
norm(a_2nd(:,2)-d(:,2)) norm(a_2nd(:,3)-d(:,3)));
norm(a_2nd(:,2)-e(:,2)) norm(a_2nd(:,3)-e(:,3)));
norm(a_2nd(:,2)-f(:,2)) norm(a_2nd(:,3)-f(:,3)));
norm(a_2nd(:,2)-g(:,2)) norm(a_2nd(:,3)-g(:,3))]
AA_2nd=[AA_2nd(:,1) 10*AA_2nd(:,2)];
bar(AA_2nd); title('A 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

AA_3rd=[norm(a_3rd(:,2)-a(:,2)) norm(a_3rd(:,3)-a(:,3)));
norm(a_3rd(:,2)-b(:,2)) norm(a_3rd(:,3)-b(:,3)));
norm(a_3rd(:,2)-c(:,2)) norm(a_3rd(:,3)-c(:,3)));
norm(a_3rd(:,2)-d(:,2)) norm(a_3rd(:,3)-d(:,3)));
norm(a_3rd(:,2)-e(:,2)) norm(a_3rd(:,3)-e(:,3)));
norm(a_3rd(:,2)-f(:,2)) norm(a_3rd(:,3)-f(:,3)));
norm(a_3rd(:,2)-g(:,2)) norm(a_3rd(:,3)-g(:,3))]

```

```
AA_3rd=[AA_3rd(:,1) 10*AA_3rd(:,2)];
bar(AA_3rd); title('A 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');
```

```
BB_2nd=[norm(b_2nd(:,2)-a(:,2)) norm(b_2nd(:,3)-a(:,3));
norm(b_2nd(:,2)-b(:,2)) norm(b_2nd(:,3)-b(:,3));
norm(b_2nd(:,2)-c(:,2)) norm(b_2nd(:,3)-c(:,3));
norm(b_2nd(:,2)-d(:,2)) norm(b_2nd(:,3)-d(:,3));
norm(b_2nd(:,2)-e(:,2)) norm(b_2nd(:,3)-e(:,3));
norm(b_2nd(:,2)-f(:,2)) norm(b_2nd(:,3)-f(:,3));
norm(b_2nd(:,2)-g(:,2)) norm(b_2nd(:,3)-g(:,3))]
BB_2nd=[BB_2nd(:,1) 10*BB_2nd(:,2)];
bar(BB_2nd); title('B 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');
```

```
BB_3rd=[norm(b_3rd(:,2)-a(:,2)) norm(b_3rd(:,3)-a(:,3));
norm(b_3rd(:,2)-b(:,2)) norm(b_3rd(:,3)-b(:,3));
norm(b_3rd(:,2)-c(:,2)) norm(b_3rd(:,3)-c(:,3));
norm(b_3rd(:,2)-d(:,2)) norm(b_3rd(:,3)-d(:,3));
norm(b_3rd(:,2)-e(:,2)) norm(b_3rd(:,3)-e(:,3));
norm(b_3rd(:,2)-f(:,2)) norm(b_3rd(:,3)-f(:,3));
norm(b_3rd(:,2)-g(:,2)) norm(b_3rd(:,3)-g(:,3))]
BB_3rd=[BB_3rd(:,1) 10*BB_3rd(:,2)];
bar(BB_3rd); title('B 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');
```

```
CC_2nd=[norm(c_2nd(:,2)-a(:,2)) norm(c_2nd(:,3)-a(:,3));
norm(c_2nd(:,2)-b(:,2)) norm(c_2nd(:,3)-b(:,3));
norm(c_2nd(:,2)-c(:,2)) norm(c_2nd(:,3)-c(:,3));
norm(c_2nd(:,2)-d(:,2)) norm(c_2nd(:,3)-d(:,3));
norm(c_2nd(:,2)-e(:,2)) norm(c_2nd(:,3)-e(:,3));
norm(c_2nd(:,2)-f(:,2)) norm(c_2nd(:,3)-f(:,3));
norm(c_2nd(:,2)-g(:,2)) norm(c_2nd(:,3)-g(:,3))]
CC_2nd=[CC_2nd(:,1) 10*CC_2nd(:,2)];
bar(CC_2nd); title('C 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');
```

```
CC_3rd=[norm(c_3rd(:,2)-a(:,2)) norm(c_3rd(:,3)-a(:,3));
norm(c_3rd(:,2)-b(:,2)) norm(c_3rd(:,3)-b(:,3));
norm(c_3rd(:,2)-c(:,2)) norm(c_3rd(:,3)-c(:,3));
norm(c_3rd(:,2)-d(:,2)) norm(c_3rd(:,3)-d(:,3));
norm(c_3rd(:,2)-e(:,2)) norm(c_3rd(:,3)-e(:,3));
norm(c_3rd(:,2)-f(:,2)) norm(c_3rd(:,3)-f(:,3));
norm(c_3rd(:,2)-g(:,2)) norm(c_3rd(:,3)-g(:,3))]
CC_3rd=[CC_3rd(:,1) 10*CC_3rd(:,2)];
bar(CC_3rd); title('C 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');
```

```
CC_4th=[norm(c_4th(:,2)-a(:,2)) norm(c_4th(:,3)-a(:,3));
norm(c_4th(:,2)-b(:,2)) norm(c_4th(:,3)-b(:,3));
norm(c_4th(:,2)-c(:,2)) norm(c_4th(:,3)-c(:,3));
```



```

norm(c_4th(:,2)-d(:,2)) norm(c_4th(:,3)-d(:,3));
norm(c_4th(:,2)-e(:,2)) norm(c_4th(:,3)-e(:,3));
norm(c_4th(:,2)-f(:,2)) norm(c_4th(:,3)-f(:,3));
norm(c_4th(:,2)-g(:,2)) norm(c_4th(:,3)-g(:,3));
CC_4th=[CC_4th(:,1) 10*CC_4th(:,2)];
bar(CC_4th); title('C 4th'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

DD_2nd=[norm(d_2nd(:,2)-a(:,2)) norm(d_2nd(:,3)-a(:,3));
norm(d_2nd(:,2)-b(:,2)) norm(d_2nd(:,3)-b(:,3));
norm(d_2nd(:,2)-c(:,2)) norm(d_2nd(:,3)-c(:,3));
norm(d_2nd(:,2)-d(:,2)) norm(d_2nd(:,3)-d(:,3));
norm(d_2nd(:,2)-e(:,2)) norm(d_2nd(:,3)-e(:,3));
norm(d_2nd(:,2)-f(:,2)) norm(d_2nd(:,3)-f(:,3));
norm(d_2nd(:,2)-g(:,2)) norm(d_2nd(:,3)-g(:,3));]
DD_2nd=[DD_2nd(:,1) 10*DD_2nd(:,2)];
bar(DD_2nd); title('D 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

DD_3rd=[norm(d_3rd(:,2)-a(:,2)) norm(d_3rd(:,3)-a(:,3));
norm(d_3rd(:,2)-b(:,2)) norm(d_3rd(:,3)-b(:,3));
norm(d_3rd(:,2)-c(:,2)) norm(d_3rd(:,3)-c(:,3));
norm(d_3rd(:,2)-d(:,2)) norm(d_3rd(:,3)-d(:,3));
norm(d_3rd(:,2)-e(:,2)) norm(d_3rd(:,3)-e(:,3));
norm(d_3rd(:,2)-f(:,2)) norm(d_3rd(:,3)-f(:,3));
norm(d_3rd(:,2)-g(:,2)) norm(d_3rd(:,3)-g(:,3));]
DD_3rd=[DD_3rd(:,1) 10*DD_3rd(:,2)];
bar(DD_3rd); title('D 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

EE_2nd=[norm(e_2nd(:,2)-a(:,2)) norm(e_2nd(:,3)-a(:,3));
norm(e_2nd(:,2)-b(:,2)) norm(e_2nd(:,3)-b(:,3));
norm(e_2nd(:,2)-c(:,2)) norm(e_2nd(:,3)-c(:,3));
norm(e_2nd(:,2)-d(:,2)) norm(e_2nd(:,3)-d(:,3));
norm(e_2nd(:,2)-e(:,2)) norm(e_2nd(:,3)-e(:,3));
norm(e_2nd(:,2)-f(:,2)) norm(e_2nd(:,3)-f(:,3));
norm(e_2nd(:,2)-g(:,2)) norm(e_2nd(:,3)-g(:,3));]
EE_2nd=[EE_2nd(:,1) 10*EE_2nd(:,2)];
bar(EE_2nd); title('E 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

EE_3rd=[norm(e_3rd(:,2)-a(:,2)) norm(e_3rd(:,3)-a(:,3));
norm(e_3rd(:,2)-b(:,2)) norm(e_3rd(:,3)-b(:,3));
norm(e_3rd(:,2)-c(:,2)) norm(e_3rd(:,3)-c(:,3));
norm(e_3rd(:,2)-d(:,2)) norm(e_3rd(:,3)-d(:,3));
norm(e_3rd(:,2)-e(:,2)) norm(e_3rd(:,3)-e(:,3));
norm(e_3rd(:,2)-f(:,2)) norm(e_3rd(:,3)-f(:,3));
norm(e_3rd(:,2)-g(:,2)) norm(e_3rd(:,3)-g(:,3));]
EE_3rd=[EE_3rd(:,1) 10*EE_3rd(:,2)];
bar(EE_3rd); title('E 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

FF_2nd=[norm(f_2nd(:,2)-a(:,2)) norm(f_2nd(:,3)-a(:,3));
norm(f_2nd(:,2)-b(:,2)) norm(f_2nd(:,3)-b(:,3));
norm(f_2nd(:,2)-c(:,2)) norm(f_2nd(:,3)-c(:,3));
norm(f_2nd(:,2)-d(:,2)) norm(f_2nd(:,3)-d(:,3));
norm(f_2nd(:,2)-e(:,2)) norm(f_2nd(:,3)-e(:,3));
norm(f_2nd(:,2)-f(:,2)) norm(f_2nd(:,3)-f(:,3));
norm(f_2nd(:,2)-g(:,2)) norm(f_2nd(:,3)-g(:,3))]
FF_2nd=[FF_2nd(:,1) 10*FF_2nd(:,2)];
bar(FF_2nd); title('F 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

FF_3rd=[norm(f_3rd(:,2)-a(:,2)) norm(f_3rd(:,3)-a(:,3));
norm(f_3rd(:,2)-b(:,2)) norm(f_3rd(:,3)-b(:,3));
norm(f_3rd(:,2)-c(:,2)) norm(f_3rd(:,3)-c(:,3));
norm(f_3rd(:,2)-d(:,2)) norm(f_3rd(:,3)-d(:,3));
norm(f_3rd(:,2)-e(:,2)) norm(f_3rd(:,3)-e(:,3));
norm(f_3rd(:,2)-f(:,2)) norm(f_3rd(:,3)-f(:,3));
norm(f_3rd(:,2)-g(:,2)) norm(f_3rd(:,3)-g(:,3))]
FF_3rd=[FF_3rd(:,1) 10*FF_3rd(:,2)];
bar(FF_3rd); title('F 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

GG_2nd=[norm(g_2nd(:,2)-a(:,2)) norm(g_2nd(:,3)-a(:,3));
norm(g_2nd(:,2)-b(:,2)) norm(g_2nd(:,3)-b(:,3));
norm(g_2nd(:,2)-c(:,2)) norm(g_2nd(:,3)-c(:,3));
norm(g_2nd(:,2)-d(:,2)) norm(g_2nd(:,3)-d(:,3));
norm(g_2nd(:,2)-e(:,2)) norm(g_2nd(:,3)-e(:,3));
norm(g_2nd(:,2)-f(:,2)) norm(g_2nd(:,3)-f(:,3));
norm(g_2nd(:,2)-g(:,2)) norm(g_2nd(:,3)-g(:,3))]
GG_2nd=[GG_2nd(:,1) 10*GG_2nd(:,2)];
bar(GG_2nd); title('G 2nd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

```

GG_3rd=[norm(g_3rd(:,2)-a(:,2)) norm(g_3rd(:,3)-a(:,3));
norm(g_3rd(:,2)-b(:,2)) norm(g_3rd(:,3)-b(:,3));
norm(g_3rd(:,2)-c(:,2)) norm(g_3rd(:,3)-c(:,3));
norm(g_3rd(:,2)-d(:,2)) norm(g_3rd(:,3)-d(:,3));
norm(g_3rd(:,2)-e(:,2)) norm(g_3rd(:,3)-e(:,3));
norm(g_3rd(:,2)-f(:,2)) norm(g_3rd(:,3)-f(:,3));
norm(g_3rd(:,2)-g(:,2)) norm(g_3rd(:,3)-g(:,3))]
GG_3rd=[GG_3rd(:,1) 10*GG_3rd(:,2)];
bar(GG_3rd); title('G 3rd'); xlabel('different objects'); ylabel('Euclidean Distance');

```

DE_Fence4Inleft_100cmDev.m

```

function feature=DE_Fence4Inleft_100cmDev
clear;
x=load('JuneFence4Inleft100cm');
%save MayTotalBrickWallX128Y8_2 x;
% column=3 and row=4 means that data_Yrow_Xcolumn is the A_scan data of the 3th
column and 4th row.
column=3;
row=4;
for i=1:2000
    data_Yrow_Xcolumn(i)=x(2000*(4*(column-1)+row-1)+i);
end
for i=1:2000
    data_Y2X1(i)=x(2000*(4*0+2-1)+i);
    data_Y2X2(i)=x(2000*(4*1+2-1)+i);
    data_Y2X3(i)=x(2000*(4*2+2-1)+i);
    .....
    data_Y2X94(i)=x(2000*(4*93+2-1)+i);
    data_Y2X95(i)=x(2000*(4*94+2-1)+i);
    data_Y2X96(i)=x(2000*(4*95+2-1)+i);
end

for i=1:500
    x0_echo(i)=data_Y2X1(500+i);
    x1_echo(i)=data_Y2X2(500+i);
    x2_echo(i)=data_Y2X3(500+i);
    .....
    x93_echo(i)=data_Y2X94(500+i);
    x94_echo(i)=data_Y2X95(500+i);
    x95_echo(i)=data_Y2X96(500+i);

end

f0=fft(x0_echo);
f1=fft(x1_echo);
f2=fft(x2_echo);
    .....
    .....

f93=fft(x93_echo);
f94=fft(x94_echo);
f95=fft(x95_echo);

for i=1:5
    f0(i)=0;
    f1(i)=0;

```

```

f2(i)=0;
.....

f93(i)=0;
f94(i)=0;
f95(i)=0;
end

for i=496:500
f0(i)=0;
f1(i)=0;
f2(i)=0;
.....

f93(i)=0;
f94(i)=0;
f95(i)=0;
end

x0_echo_improve=ifft(f0);
x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
.....

x93_echo_improve=ifft(f93);
x94_echo_improve=ifft(f94);
x95_echo_improve=ifft(f95);

energy(1)=sum(abs(x0_echo_improve));
energy(2)=sum(abs(x1_echo_improve));
energy(3)=sum(abs(x2_echo_improve));
.....

energy(94)=sum(abs(x93_echo_improve));
energy(95)=sum(abs(x94_echo_improve));
energy(96)=sum(abs(x95_echo_improve));

%The following is designed to find the amplitude of strongest A-scan among the A-scans
facing to the central part of the fence.

x_echo_improve(1,:)=abs(x0_echo_improve);
x_echo_improve(2,:)=abs(x1_echo_improve);
x_echo_improve(3,:)=abs(x2_echo_improve);
.....

```

```

x_echo_improve(94,:)=abs(x93_echo_improve);
x_echo_improve(95,:)=abs(x94_echo_improve);
x_echo_improve(96,:)=abs(x95_echo_improve);

% The following is designed to find n main peaks of the square root of signal energy
%plots and make the n-peak template to compare with the actual data.

for i=1:7
    deviation(i)=getdev2(i,energy);
end

for i=1:6
    difference(i)=deviation(i)-deviation(i+1);
end

%4th order fitting of the deviation vs.n curve above
cof=polyfit(1:7,deviation,4);
x1=1:0.01:7;
y1=cof(1).*x1.^4+cof(2).*x1.^3+cof(3).*x1.^2+cof(4).*x1+cof(5);
Dy1=4*cof(1).*x1.^3+3*cof(2).*x1.^2+2*cof(3).*x1+cof(4);
plot(x1,Dy1)

if difference(1)<0.03
    num_peak=1;
else
    if difference(2)<0.03
        num_peak=2;
    else
        if difference(3)<0.03
            num_peak=3;
        else
            if difference(4)<0.03
                num_peak=4;
            else
                if difference(5)<0.03
                    num_peak=5;
                else
                    if difference(6)<0.03
                        num_peak=6;
                    else
                        end
                    end
                end
            end
        end
    end
end
end

```

```

        end
    end
end

num_peak;

derivative(1)=4*cof(1).*1.5^3+3*cof(2).*1.5^2+2*cof(3).*1.5+cof(4);
derivative(2)=4*cof(1).*2.5^3+3*cof(2).*2.5^2+2*cof(3).*2.5+cof(4);
derivative(3)=4*cof(1).*3.5^3+3*cof(2).*3.5^2+2*cof(3).*3.5+cof(4);
derivative(4)=4*cof(1).*4.5^3+3*cof(2).*4.5^2+2*cof(3).*4.5+cof(4);
derivative(5)=4*cof(1).*5.5^3+3*cof(2).*5.5^2+2*cof(3).*5.5+cof(4);
derivative(6)=4*cof(1).*6.5^3+3*cof(2).*6.5^2+2*cof(3).*6.5+cof(4);

derivative

if derivative(1)>-0.01
    num_peak_der=1;
else
    if derivative(2)>-0.01
        num_peak_der=2;
    else
        if derivative(3)>-0.01
            num_peak_der=3;
        else
            if derivative(4)>-0.01
                num_peak_der=4;
            else
                if derivative(5)>-0.01
                    num_peak_der=5;
                else
                    if deriative(6)>-0.01
                        num_peak_der=6;
                    else
                        end
                    end
                end
            end
        end
    end
end

location_npeak=npeak2(energy, num_peak_der)

if num_peak_der==1
    else if num_peak_der==2
        a1=(min(energy(location_npeak(1)),energy(location_npeak(2))))-
min(energy(location_npeak(1):location_npeak(2)));

```

```

a2=(max(energy(location_npeak(1)),energy(location_npeak(2))))-
min(energy(location_npeak(1):location_npeak(2)));
ratio=a1/a2;
if (ratio<0.2)|(abs(location_npeak(1)-location_npeak(2))<15)
    num_peak_der=1;
else
    num_peak_der=2;
end
else if num_peak_der==3
    b1=(energy(location_npeak(1))-
min(energy(location_npeak(1):location_npeak(2))));

b2=(max([energy(location_npeak(1)),energy(location_npeak(2)),energy(location_npeak(
3))]))-min(energy(location_npeak(1):location_npeak(2)));
    ratio1=b1/b2;

    c1=(energy(location_npeak(3))-
min(energy(location_npeak(2):location_npeak(3))));

c2=(max([energy(location_npeak(1)),energy(location_npeak(2)),energy(location_npeak(
3))]))-min(energy(location_npeak(1):location_npeak(2)));
    ratio2=c1/c2;

    if (ratio1<0.2)|(abs(location_npeak(1)-location_npeak(2))<15)
        num_peak_der=num_peak_der-1;
    else
        end
    if (ratio2<0.2)|(abs(location_npeak(2)-location_npeak(3))<15)
        num_peak_der=num_peak_der-1;
    else
        end
    else if num_peak_der==4
        d1=(energy(location_npeak(1))-
min(energy(location_npeak(1):location_npeak(2))));

d2=(max([energy(location_npeak(1)),energy(location_npeak(2)),energy(location_npeak(
3)),energy(location_npeak(4))]))-min(energy(location_npeak(1):location_npeak(2)));
        ratio1=d1/d2;

        e1=(energy(location_npeak(3))-
min(energy(location_npeak(2):location_npeak(3))));

e2=(max([energy(location_npeak(1)),energy(location_npeak(2)),energy(location_npeak(
3)),energy(location_npeak(4))]))-min(energy(location_npeak(2):location_npeak(3)));
        ratio2=e1/e2;

```

```

        f1=(energy(location_npeak(4))-
min(energy(location_npeak(2):location_npeak(4))));

f2=(max([energy(location_npeak(1)),energy(location_npeak(2)),energy(location_npeak(3
)),energy(location_npeak(4))]))-min(energy(location_npeak(2):location_npeak(4)));
        ratio3=f1/f2;
        if (ratio1<0.2)|(abs(location_npeak(1)-location_npeak(2))<15)
            num_peak_der=num_peak_der-1;
        else
            end
        if (ratio2<0.2)|(abs(location_npeak(2)-location_npeak(3))<15)
            num_peak_der=num_peak_der-1;
        else
            end
        if (ratio3<0.2)|(abs(location_npeak(2)-location_npeak(4))<15)
            num_peak_der=num_peak_der-1;
        else
            end
    else
        end
    end
end
end
end
end

```

```
feature=[num_peak_der, deviation(num_peak_der)];
```

```
plot(energy,'o')
```

getdev2.m

```
function deviation=getdev2(n,energy)
```

```
featurenpeak=npeak2(energy,n);
```

```

for j=1:(n-1)
    for i=1:96
        if energy(i)==min(energy(featurenpeak(j):featurenpeak(j+1)))
            n_min(j)=i;
        else
            end
        end
    end
end
end

```

```
x1=1:96;
```



```

y1(1:featurenpeak(1))=((energy(featurenpeak(1))-
energy(1))/(featurenpeak(1))^5*(x1(1:featurenpeak(1))).^5+energy(1);
y1(featurenpeak(n):96)=((energy(featurenpeak(n))-energy(96))/(featurenpeak(n)-
96)^5)*(x1(featurenpeak(n):96)-96).^5+energy(96);

for i=1:(n-1)
    y1(featurenpeak(i):featurenpeak(i+1))=fitpeak(featurenpeak(i), n_min(i),
featurenpeak(i+1), energy);
end

deviation=0;
for i=1:96
    deviation=deviation+abs(y1(i)-energy(i));
end

deviation=deviation/(max(energy)-min(energy))/96;

```

npeak2.m

```

% This sub-program is designed to detect the n main peaks of the square root of signal
% energy plots of data file E_FenceInleftX96Y4_100cm, E_FenceInrightX96Y4_100cm,
% E_FenceInleftX96Y4_120cm,E_FenceInrightX96Y4_120cm,
E_FenceInleftX96Y4_140cm,
% E_FenceInrightX96Y4_140cm, and E_MFenceleftX96Y4_100cm,
E_MFencerightX96Y4_100cm,
% E_MFenceleftX96Y4_120cm, E_MFencerightX96Y4_120cm,
E_MFenceleftX96Y4_140cm,
% E_MFencerightX96Y4_140cm.

%The purpose to detect the n peaks is to create an N-peak template to match the above
data.
% For E_Fences above, we will have good matches with the template.
% For E_MFences above, we will have bad matches with the template.

function featurenpeak=npeak2(energy,n)

j=2;
for i=2:95
    if ((energy(i)-energy(i-1))>0)& ((energy(i+1)-energy(i))<0)
        energy_peaks(j)=energy(i);
        energy_peaks_location(j)=i;
        j=j+1;
    else
        end
    end
end

```

```

energy_peaks(1)=energy(1);
energy_peaks_location(1)=1;
energy_peaks(j)=energy(96);
energy_peaks_location(j)=96;

m=1;
for k=2:(length(energy_peaks)-1)
    if ((energy_peaks(k)-energy_peaks(k-1))>0)&((energy_peaks(k+1)-
energy_peaks(k))<0)
        energy_peaks_2nd(m)=energy_peaks(k);
        energy_peaks_2nd_location(m)=energy_peaks_location(k);
        m=m+1;
    else
        end
end

%energy_peaks_2nd
%energy_peaks_2nd_location

if n>length(energy_peaks_2nd)
    fprintf('n is larger than the total peaks')
else
    end

f=sort(energy_peaks_2nd);

for j=1:n
    for i=1:length(energy_peaks_2nd)
        if f(length(energy_peaks_2nd)-(j-1))==energy_peaks_2nd(i)
            energy_peak_location(j)=energy_peaks_2nd_location(i);
        else
            end
        end
    end
end

featurenpeak=sort(energy_peak_location);

featureVsN.m

feature_Fence4In(1,:)=E_Fence4InLeftedge_100cmDev;
hold on;
feature_Fence4In(2,:)=E_Fence4InLeft_100cmDev;

```

```

hold on;
feature_Fence4In(3,:)=E_Fence4Incenter_100cmDev;
hold on;
feature_Fence4In(4,:)=E_Fence4Inright_100cmDev;
hold on;
feature_Fence4In(5,:)=E_Fence4Inrightedge_100cmDev;
title('Picket Fence Scanned from Inside
(Fence4In)');xlabel('N(Template)');ylabel('Deviation');

feature_FBrickWall(1,:)=E_BrickwallNo1_nMatchDev;
hold on;
feature_FBrickWall(2,:)=E_BrickwallNo2_nMatchDev;
hold on;
feature_FBrickWall(3,:)=E_BrickwallNo3_nMatchDev;
hold on;
feature_FBrickWall(4,:)=E_BrickwallNo4_nMatchDev;
hold on;
feature_FBrickWall(5,:)=E_BrickwallNo5_nMatchDev;
title('Flat Brick Wall (BrickWallNo.1-5)');xlabel('N(Template)');ylabel('Deviation');

feature_Fence4(1,:)=E_Fence4Leftedge_100cmDev;
hold on;
feature_Fence4(2,:)=E_Fence4Left_100cmDev;
hold on;
feature_Fence4(3,:)=E_Fence4center_100cmDev;
hold on;
feature_Fence4(4,:)=E_Fence4right_100cmDev;
hold on;
feature_Fence4(5,:)=E_Fence4rightedge_100cmDev;
title('Picket Fence Scanned from outside
(Fence4)');xlabel('N(Template)');ylabel('Deviation');

feature_MFence(1,:)=E_MFenceLeftedge_100cmDev;
hold on;
feature_MFence(2,:)=E_MFenceLeft_100cmDev;
hold on;
feature_MFence(3,:)=E_MFencecenter_100cmDev;
hold on;
feature_MFence(4,:)=E_MFenceright_100cmDev;
hold on;
feature_MFence(5,:)=E_MFencerightedge_100cmDev;
title('Metal Fence with Brick Pillar (MFence)'); xlabel('N(Template)');ylabel('Deviation');

feature_Hedge(1,:)=E_BushNo1_Dev;
hold on;

```

```

feature_Hedge(2,:)=E_BushNo2_Dev;
hold on;
feature_Hedge(3,:)=E_BushNo3_Dev;
hold on;
feature_Hedge(4,:)=E_BushNo4_Dev;
hold on;
feature_Hedge(5,:)=E_BushNo5_Dev;
title('Hedge (BushNo.1-5)');xlabel('N(Template)');ylabel('Deviation');

```

```

feature_BwallNo2(1,:)=E_BwallNo2Leftedge_100cmDev;
hold on;
feature_BwallNo2(2,:)=E_BwallNo2Left_100cmDev;
hold on;
feature_BwallNo2(3,:)=E_BwallNo2center_100cmDev;
hold on;
feature_BwallNo2(4,:)=E_BwallNo2right_100cmDev;
hold on;
feature_BwallNo2(5,:)=E_BwallNo2rightedge_100cmDev;
title('Brick Wall with Thin Buttress
(BWallNo.2)');xlabel('N(Template)');ylabel('Deviation');

```

```

feature_BwallNo1(1,:)=E_BwallNo1Leftedge_100cmNov;
hold on;
feature_BwallNo1(2,:)=E_BwallNo1Left_100cmNov;
hold on;
feature_BwallNo1(3,:)=E_BwallNo1center_100cmNov;
hold on;
feature_BwallNo1(4,:)=E_BwallNo1right_100cmNov;
hold on;
feature_BwallNo1(5,:)=E_BwallNo1rightedge_100cmNov;
title('Brick Wall with Thick Buttress
(BWallNo.1)');xlabel('N(Template)');ylabel('Deviation');

```

E_Fence4Inleft_100cmDev.m

```

function feature=E_Fence4Inleft_100cmDev
clear;
x=load('JuneFence4Inleft100cm');
%save MayTotalBrickWallX128Y8_2 x;
% column=3 and row=4 means that data_Yrow_Xcolumn is the A_scan data of the 3th
column and 4th row.
column=3;
row=4;

```

```

for i=1:2000
    data_Yrow_Xcolumn(i)=x(2000*(4*(column-1)+row-1)+i);
end
for i=1:2000
    data_Y2X1(i)=x(2000*(4*0+2-1)+i);
    data_Y2X2(i)=x(2000*(4*1+2-1)+i);
    data_Y2X3(i)=x(2000*(4*2+2-1)+i);
    .....
    data_Y2X94(i)=x(2000*(4*93+2-1)+i);
    data_Y2X95(i)=x(2000*(4*94+2-1)+i);
    data_Y2X96(i)=x(2000*(4*95+2-1)+i);
end

for i=1:500
    x0_echo(i)=data_Y2X1(500+i);
    x1_echo(i)=data_Y2X2(500+i);
    x2_echo(i)=data_Y2X3(500+i);
    .....
    x93_echo(i)=data_Y2X94(500+i);
    x94_echo(i)=data_Y2X95(500+i);
    x95_echo(i)=data_Y2X96(500+i);

end

f0=fft(x0_echo);
f1=fft(x1_echo);
f2=fft(x2_echo);
    .....
    f93=fft(x93_echo);
    f94=fft(x94_echo);
    f95=fft(x95_echo);

for i=1:5
    f0(i)=0;
    f1(i)=0;
    f2(i)=0;
    .....
    f93(i)=0;
    f94(i)=0;
    f95(i)=0;
end

```

```

for i=496:500
    f0(i)=0;
    f1(i)=0;
    f2(i)=0;
        .....

    f93(i)=0;
    f94(i)=0;
    f95(i)=0;
end

x0_echo_improve=ifft(f0);
x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
        .....

x93_echo_improve=ifft(f93);
x94_echo_improve=ifft(f94);
x95_echo_improve=ifft(f95);

energy(1)=sum(abs(x0_echo_improve));
energy(2)=sum(abs(x1_echo_improve));
energy(3)=sum(abs(x2_echo_improve));
        .....

energy(94)=sum(abs(x93_echo_improve));
energy(95)=sum(abs(x94_echo_improve));
energy(96)=sum(abs(x95_echo_improve));

%The following is designed to find the amplitude of strongest A-scan among the A-scans
facing to the central part of the fence.

x_echo_improve(1,:)=abs(x0_echo_improve);
x_echo_improve(2,:)=abs(x1_echo_improve);
x_echo_improve(3,:)=abs(x2_echo_improve);
        .....

x_echo_improve(94,:)=abs(x93_echo_improve);
x_echo_improve(95,:)=abs(x94_echo_improve);
x_echo_improve(96,:)=abs(x95_echo_improve);

% The following is designed to find n main peaks of the square root of signal energy

```

```

% plots and make the n-peak template to compare with the actual data.

for i=1:7
    deviation(i)=getdev2(i,energy);
end
plot(deviation,'x');
hold on;
%axis([-2 10 -0.1 0.5])

%4th order fitting of the deviation vs.n curve above
cof=polyfit(1:7,deviation,4);
x1=-2:0.01:7;
y1=cof(1).*x1.^4+cof(2).*x1.^3+cof(3).*x1.^2+cof(4).*x1+cof(5);
l_4th=cof(1);
m_2nd=cof(3)-3*(cof(2)).^2/(8*cof(1));
n_1st=(cof(2)).^3/(8*cof(1).^2)-cof(2)*cof(3)/(2*cof(1))+cof(4);
origin_X=-cof(2)/(4*cof(1));
origin_Y=(-3)*cof(2).^4/(256*cof(1).^3)+cof(2).^2*cof(3)/(16*cof(1).^2)-
cof(4)*cof(2)/(4*cof(1))+cof(5);
%plot(x1,y1);
x2=1:0.01:7;
y2=cof(1)*(x2+cof(2)/(4*cof(1))).^4+m_2nd*(x2+cof(2)/(4*cof(1))).^2+n_1st*(x2+cof(
2)/(4*cof(1)))+origin_Y;
plot(x2,y2,'-')
hold off

feature=[l_4th m_2nd n_1st origin_X origin_Y];

```

Appendix E

Matlab Code for Analysis of Outside Rectangular and Cylindrical Objects

Am_RoundTrashCanFT3X104Y2.m

```

function feature=Am_RoundTrashCanFT3X104Y2
clear;
x=load('FebRoundTrashCanFT3X104Y2');
% column=3 and row=4 means that data_Yrow_Xcolumn is the A_scan data of the 3th
column and 4th row.
column=3;
row=2;
for i=1:2000
    data_Yrow_Xcolumn(i)=x(2000*(4*(column-1)+row-1)+i);
end
for i=1:2000
    data_Y2X1(i)=x(2000*(2*0+2-1)+i);
    data_Y2X2(i)=x(2000*(2*1+2-1)+i);
    data_Y2X3(i)=x(2000*(2*2+2-1)+i);
    .....
    data_Y2X102(i)=x(2000*(2*101+2-1)+i);
    data_Y2X103(i)=x(2000*(2*102+2-1)+i);
    data_Y2X104(i)=x(2000*(2*103+2-1)+i);
end

for i=1:500
    x0_echo(i)=data_Y2X1(500+i);
    x1_echo(i)=data_Y2X2(500+i);
    x2_echo(i)=data_Y2X3(500+i);
    .....

    x101_echo(i)=data_Y2X102(500+i);
    x102_echo(i)=data_Y2X103(500+i);
    x103_echo(i)=data_Y2X104(500+i);
end

f0=fft(x0_echo);
f1=fft(x1_echo);
f2=fft(x2_echo);
    .....

```



```

f101=fft(x101_echo);
f102=fft(x102_echo);
f103=fft(x103_echo);

%filter the very low frequency component of xn_echo and yn_echo
for i=1:5
    f0(i)=0;
    f1(i)=0;
    f2(i)=0;
        .....

    f101(i)=0;
    f102(i)=0;
    f103(i)=0;
end

for i=496:500
    f0(i)=0;
    f1(i)=0;
    f2(i)=0;
        .....

    f101(i)=0;
    f102(i)=0;
    f103(i)=0;
end

x0_echo_improve=ifft(f0);
x1_echo_improve=ifft(f1);
x2_echo_improve=ifft(f2);
    .....

.....
x101_echo_improve=ifft(f101);
x102_echo_improve=ifft(f102);
x103_echo_improve=ifft(f103);

energy(1)=max(abs(x2_echo_improve));
energy(2)=max(abs(x3_echo_improve));
energy(3)=max(abs(x4_echo_improve));
    .....

energy(98)=max(abs(x99_echo_improve));
energy(99)=max(abs(x100_echo_improve));
energy(100)=max(abs(x101_echo_improve));

```

```

for i=1:100
    if energy(i)==max(energy)
        center_Num=i;
    else
    end
end

if max(energy)>=950
    for i=1:100
        if abs(energy(i)-1000)<50
            right_Num=i;
        else
        end
    end

    for i=1:100
        if abs(energy(101-i)-1000)<50
            left_Num=101-i;
        else
        end
    end

    center_Num
    left_Num
    right_Num
    n=1:100;
    plot(n,energy,'ok')
    hold on;

    cof1=polyfit(1:left_Num,energy(1:left_Num),4);
    xx1=1:0.1:left_Num;
    yy1=cof1(1)*xx1.^4+cof1(2)*xx1.^3+cof1(3)*xx1.^2+cof1(4)*xx1+cof1(5);
    plot(xx1,yy1,'-k','linewidth',1.5);

    cof2=polyfit(right_Num:96,energy(right_Num:96),4);
    xx2=right_Num:0.1:100;
    yy2=cof2(1)*xx2.^4+cof2(2)*xx2.^3+cof2(3)*xx2.^2+cof2(4)*xx2+cof2(5);
    plot(xx2,yy2,'-k','linewidth',1.5);
    hold off;

    y_500=500;

    for i=1:length(yy1)

```

```

        error1(i)=abs(yy1(i)-y_500);
        if error1(i)==min(error1)
            n_left=i;
        else
            end
        end

    for i=1:length(yy2)
        error2(i)=abs(yy2(i)-y_500);
        if error2(i)==min(error2)
            n_right=i;
        else
            end
        end

    n_right=n_right+right_Num*10;

    Width=(n_right-n_left)/10;

    length_Top=abs(right_Num-left_Num+1);

else
    n=1:100;
    plot(n,energy,'ok')
    hold on;

    cof1=polyfit(1:center_Num,energy(1:center_Num),4);
    xx1=1:0.1:center_Num;
    yy1=cof1(1)*xx1.^4+cof1(2)*xx1.^3+cof1(3)*xx1.^2+cof1(4)*xx1+cof1(5);
    plot(xx1,yy1,'-k','linewidth',1.5);

    cof2=polyfit(center_Num:100,energy(center_Num:100),4);
    xx2=center_Num:0.1:100;
    yy2=cof2(1)*xx2.^4+cof2(2)*xx2.^3+cof2(3)*xx2.^2+cof2(4)*xx2+cof2(5);
    plot(xx2,yy2,'-k','linewidth',1.5);
    hold off;

    y_500=500;

    for i=1:length(yy1)
        error1(i)=abs(yy1(i)-y_500);
        if error1(i)==min(error1)
            n_left=i;
        else
            end
        end
    end
end

```

```

for i=1:length(yy2)
    error2(i)=abs(yy2(i)-y_500);
    if error2(i)==min(error2)
        n_right=i;
    else
        end
    end
end

n_right=n_right+center_Num*10;

Width=(n_right-n_left)/10;

length_Top=1;

end

for i=1:2000
    data_Y2XCenter_NumPlus2(i)=x(2000*(2*(center_Num+2-1)+2-1)+i);
end

for i=1:1700
    if data_Y2XCenter_NumPlus2(2001-i)>=500
        counter=2001-i;
    else
        end
    end
end

feature=[length_Top Width counter];

```

Fit_Ch6.m

```

a1=Am_BrickwallFT3X104Y2;
a2=Am_BrickwallFT4X104Y2;
a3=Am_BrickwallFT5X104Y2;
a4=Am_BrickwallFT6X104Y2;
a5=Am_BrickwallFT7X104Y2;
a6=Am_BrickwallFT8X104Y2;
a7=Am_BrickwallFT9X104Y2;
a8=Am_BrickwallFT10X104Y2;

b1=Am_ConvexBWallFT3X104Y2No2;
b2=Am_ConvexBWallFT4X104Y2No2;
b3=Am_ConvexBWallFT5X104Y2No2;

```

b4=Am_ConvexBWallFT6X104Y2No2;
 b5=Am_ConvexBWallFT7X104Y2No2;
 b6=Am_ConvexBWallFT8X104Y2No2;
 b7=Am_ConvexBWallFT9X104Y2No2;
 b8=Am_ConvexBWallFT10X104Y2No2;

c1=Am_SquareTrashCanNo2FT3X104Y2;
 c2=Am_SquareTrashCanNo2FT4X104Y2;
 c3=Am_SquareTrashCanNo2FT5X104Y2;
 c4=Am_SquareTrashCanNo2FT6X104Y2;
 c5=Am_SquareTrashCanNo2FT7X104Y2;
 c6=Am_SquareTrashCanNo2FT8X104Y2;
 c7=Am_SquareTrashCanNo2FT9X104Y2;
 c8=Am_SquareTrashCanNo2FT10X104Y2;

d1=Am_RoundTrashCanFT3X104Y2;
 d2=Am_RoundTrashCanFT4X104Y2;
 d3=Am_RoundTrashCanFT5X104Y2;
 d4=Am_RoundTrashCanFT6X104Y2;
 d5=Am_RoundTrashCanFT7X104Y2;
 d6=Am_RoundTrashCanFT8X104Y2;
 d7=Am_RoundTrashCanFT9X104Y2;
 d8=Am_RoundTrashCanFT10X104Y2;

e1=Am_SquareLPFT3X104Y2;
 e2=Am_SquareLPFT4X104Y2;
 e3=Am_SquareLPFT5X104Y2;
 e4=Am_SquareLPFT6X104Y2;
 e5=Am_SquareLPFT7X104Y2;
 e6=Am_SquareLPFT8X104Y2;
 e7=Am_SquareLPFT9X104Y2;
 e8=Am_SquareLPFT10X104Y2;

f1=Am_RoundLPFT3X104Y2;
 f2=Am_RoundLPFT4X104Y2;
 f3=Am_RoundLPFT5X104Y2;
 f4=Am_RoundLPFT6X104Y2;
 f5=Am_RoundLPFT7X104Y2;
 f6=Am_RoundLPFT8X104Y2;
 f7=Am_RoundLPFT9X104Y2;
 f8=Am_RoundLPFT10X104Y2;

Length_Top_Brickwall=[a1(1) a2(1) a3(1) a4(1) a5(1) a6(1) a7(1) a8(1)];

Width_Brickwall=[a1(2) a2(2) a3(2) a4(2) a5(2) a6(2) a7(2) a8(2)];

```

Counter_Brickwall=[a1(3) a2(3) a3(3) a4(3) a5(3) a6(3) a7(3) a8(3)];
%Length_Top_Brickwall=[22 9 9 7 6 5 4 1];
%Width_Brickwall=[29.4 17.7 14.9 15.4 18.2 13.3 10.1 3.3];
%Counter_Brickwall=[579 745 922 1073 1303 1463 1647 1808];
cof1_Brickwall=polyfit(Counter_Brickwall,Length_Top_Brickwall,1)
cof2_Brickwall=polyfit(Counter_Brickwall,Width_Brickwall,1)
plot(Counter_Brickwall, Length_Top_Brickwall,'or');
hold on;
plot(Counter_Brickwall, Width_Brickwall,'sr');

```

```

Length_Top_ConvexBWall=[b1(1) b2(1) b3(1) b4(1) b5(1) b6(1) b7(1) b8(1)];
Width_ConvexBWall=[b1(2) b2(2) b3(2) b4(2) b5(2) b6(2) b7(2) b8(2)];
Counter_ConvexBWall=[b1(3) b2(3) b3(3) b4(3) b5(3) b6(3) b7(3) b8(3)];
%Length_Top_ConvexBWall=[21 11 3 1 1 1 1 1];
%Width_ConvexBWall=[33 23.8 17 2.8 9.1 8.2 4 1];
%Counter_ConvexBWall=[559 742 927 1165 1279 1466 1656 1834];
cof1_ConvexBWall=polyfit(Counter_ConvexBWall(1:4),Length_Top_ConvexBWall(1:4),1)
cof2_ConvexBWall=polyfit(Counter_ConvexBWall,Width_ConvexBWall,1)
plot(Counter_ConvexBWall, Length_Top_ConvexBWall,'om');
hold on;
plot(Counter_ConvexBWall, Width_ConvexBWall,'sm');

```

```

Length_Top_SquareTrashCan=[c1(1) c2(1) c3(1) c4(1) c5(1) c6(1) c7(1) c8(1)];
Width_SquareTrashCan=[c1(2) c2(2) c3(2) c4(2) c5(2) c6(2) c7(2) c8(2)];
Counter_SquareTrashCan=[c1(3) c2(3) c3(3) c4(3) c5(3) c6(3) c7(3) c8(3)];
%Length_Top_SquareTrashCan=[44 14 4 1 1 1 1 1];
%Width_SquareTrashCan=[50.1 36.2 21.7 15.5 10.2 3.1 1.1 1];
%Counter_SquareTrashCan=[551 727 899 1098 1258 1430 1610 1744];
cof1_SquareTrashCan=polyfit(Counter_SquareTrashCan(1:4),Length_Top_SquareTrashCan(1:4),1)

```

```

cof2_SquareTrashCan=polyfit(Counter_SquareTrashCan,Width_SquareTrashCan,1)
plot(Counter_SquareTrashCan, Length_Top_SquareTrashCan,'ok');
plot(Counter_SquareTrashCan, Width_SquareTrashCan,'sk');

Length_Top_RoundTrashCan=[d1(1) d2(1) d3(1) d4(1) d5(1) d6(1) d7(1) d8(1)];
Width_RoundTrashCan=[d1(2) d2(2) d3(2) d4(2) d5(2) d6(2) d7(2) d8(2)];
Counter_RoundTrashCan=[d1(3) d2(3) d3(3) d4(3) d5(3) d6(3) d7(3) d8(3)];
%Length_Top_RoundTrashCan=[15 8 1 1 1 1 1 1];
%Width_RoundTrashCan=[25.6 18.1 9.2 2.8 3.3 1.1 1 1];
%Counter_RoundTrashCan=[577 756 922 1114 1109 1473 1675 1860];
cof1_RoundTrashCan=polyfit(Counter_RoundTrashCan(1:3),Length_Top_RoundTrashC
an(1:3),1)
cof2_RoundTrashCan=polyfit(Counter_RoundTrashCan(1:7),Width_RoundTrashCan(1:
7),1)
plot(Counter_RoundTrashCan, Length_Top_RoundTrashCan,'og');
plot(Counter_RoundTrashCan, Width_RoundTrashCan,'sg');
Length_Top_SquareLP=[e1(1) e2(1) e3(1) e4(1) e5(1) e6(1) e7(1) e8(1)];
Width_SquareLP=[e1(2) e2(2) e3(2) e4(2) e5(2) e6(2) e7(2) e8(2)];
Counter_SquareLP=[e1(3) e2(3) e3(3) e4(3) e5(3) e6(3) e7(3) e8(3)];
%Length_Top_SquareLP=[14 10 6 1 1 1 1 1];
%Width_SquareLP=[28.3 19.1 14.4 7.7 11.4 7.5 1.6 1];
%Counter_SquareLP=[591 728 896 1081 1295 1461 1608 1832];
cof1_SquareLP=polyfit(Counter_SquareLP(1:4),Length_Top_SquareLP(1:4),1)
cof2_SquareLP=polyfit(Counter_SquareLP,Width_SquareLP,1)
plot(Counter_SquareLP, Length_Top_SquareLP,'ob');
plot(Counter_SquareLP, Width_SquareLP,'sb');

Length_Top_RoundLP=[f1(1) f2(1) f3(1) f4(1) f5(1) f6(1) f7(1) f8(1)];
Width_RoundLP=[f1(2) f2(2) f3(2) f4(2) f5(2) f6(2) f7(2) f8(2)];
Counter_RoundLP=[f1(3) f2(3) f3(3) f4(3) f5(3) f6(3) f7(3) f8(3)];
%Length_Top_RoundLP=[3 1 1 1 1 1 1 1];

```

```

%Width_RoundLP=[13 1 1 1 1 1 1 1];
%Counter_RoundLP=[599 792 955 1174 1318 1504 1671 1824];
cof1_RoundLP=polyfit(Counter_RoundLP(1:2),Length_Top_RoundLP(1:2),1)
cof2_RoundLP=polyfit(Counter_RoundLP(1:2),Width_RoundLP(1:2),1)
plot(Counter_RoundLP, Length_Top_RoundLP,'oc');
plot(Counter_RoundLP, Width_RoundLP,'sc');
hold off;

```

```

cof1_Brickwall(1)=0.0122;
cof2_Brickwall(1)=0.0147;
cof1_ConvexBWall(1)=0.0332;
cof2_ConvexBWall(1)=0.0230;
cof1_SquareTrashCan(1)=0.0758;
cof2_SquareTrashCan(1)=0.0395;
cof1_RoundTrashCan(1)=0.0406;
cof2_RoundTrashCan(1)=0.0219;
cof1_SquareLP(1)=0.0262;
cof2_SquareLP(1)=0.0194;
cof1_RoundLP(1)=0.0104;
cof2_RoundLP(1)=0.0622;

```

```

plot(cof1_RoundLP(1), cof2_RoundLP(1),'.k');
axis([0 0.08 0 0.07]);
hold on;
plot(cof1_SquareLP(1), cof2_SquareLP(1),'sk');
plot(cof1_RoundTrashCan(1), cof2_RoundTrashCan(1),'ob');
plot(cof1_SquareTrashCan(1), cof2_SquareTrashCan(1),'sb');
plot(cof1_ConvexBWall(1), cof2_ConvexBWall(1),'or');
plot(cof1_Brickwall(1), cof2_Brickwall(1),'sr');
hold off;

```


Bibliography

- [1] Whitlow W.L. Au, *The Sonar of Dolphins*, Springer-Verlag.
- [2] H. Maxim, Preventing Collisions at Sea: A Mechanical Application of the Bat's Sixth Sense. By Sir Hiram Maxim. *Sci. Amer.*, July 27, 1912, pp. 80-81.
- [3] Donald R. Griffin, *Listening in the Dark*, Yale University Press, New Haven, CT, 1958.
- [4] James A. Simmons, The Acoustic Basis for Target Discrimination by FM Echolocating Bats, *J. Acoust. Soc. Am.*, Vol. 86, Issue 4, October 1989, pp. 1333-1350.
- [5] James A. Simmons, The Performance of Echolocation: Acoustic Images Perceived by Echolocating Bats, *Animal Sonar*, Naval Ocean Systems Center, Kailua, Hawaii.
- [6] James A. Simmons, M. Brock Fenton and Michael J. O'Farrell, Echolocation and Pursuit of Prey by Bats, *Science*, Vol. 203, Issue 5, January 1979, pp. 16-21.
- [7] Rudolf Kober and Hans-Ulrich Schnitzler, Information in Sonar Echoes of Fluttering Insects Available for Echolocating Bats, *J. Acoust. Soc. Am.*, Vol. 87, Issue 2, February, 1990.
- [8] Billur Barshan and Roman Kuc, A Bat-like Sonar System for Obstacle Localization, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, Issue 4, July/August 1992, pp. 636-646.
- [9] Roman Kuc, Three-dimensional Tracking Using Qualitative Bionic Sonar, *Robotics and Autonomous Systems*, Elsevier, Vol. 11, 1993, pp. 213-219.
- [10] Roman Kuc, Sensorimotor Model of Bat Echolocation and Prey Capture, *J. Acoust. Soc. Am.*, Vol. 96, Issue 4, October 1994, pp. 1966-1977.
- [11] Lindsay Kleeman and Roman Kuc, Mobile Robot Sonar for Target Localization and Classification, *The International Journal of Robotics Research*, Vol. 14, Issue 4, August 1995, pp. 295-318.
- [12] Roman Kuc. Fusing Binaural Sonar Information for Object Recognition, *Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 727-735.
- [13] Roman Kuc, Biologically Motivated Adaptive Sonar System, *J. Acoust. Soc. Am.*, Vol. 100, Issue 3, September 1996, pp. 1849-1854.

- [14] Roman Kuc, Biomimetic Sonar Recognizes Objects Using Binaural Information, *J. Acoust. Soc. Am.*, Vol. 102, Issue 2, 1997, pp. 689-696.
- [15] Roman Kuc, Biomimetic Sonar Recognizes Objects from Echoes, Chapter in *Biosonar* edited by Jeannette Thomas, University of Chicago Press, 2001.
- [16] Roman Kuc, Recognizing Retro-Reflectors with and Obliquely-Oriented Multi-Point Sonar and Acoustic Flow, *The International Journal of Robotics Research*, Vol. 22, Issue 2, 2003, pp. 129-148.
- [17] L. G. Ullate, M. T. Sanchez, E. Villanueva, M. Parrilla and J. J. Anaya, A Three-transducer Ultrasonic System for Object Location in Air, *Sensors and Actuators A*, Vol. 37-38, 1993, pp. 391-396.
- [18] Ken Sasaki and Masaharu Takano, Classification of Objects' Surface by Acoustic Transfer Function, *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, NC, July 7-10, 1992, pp. 821-828.
- [19] J. R. Llata, E. G. Sarabia and J. P. Oria, Three-dimensional Robotic Vision Using Ultrasonic Sensors, *Journal of Intelligent and Robotics System*, Vol. 33, 2002, pp. 267-284.
- [20] Billur Barshan, Location and Curvature Estimation of Spherical Targets Using Multiple Sonar Time-of-flight Measurements, *IEEE Transactions on Instrumentation and Measurement*, Vol. 48, No. 6. December 1999, pp. 1212-1223.
- [21] Billur Barshan and Deniz Baskent, Morphological Surface Profile Extraction with Multiple Range Sensors, *Pattern Recognition*, Vol. 34, 2001, pp. 1459-1467.
- [22] Birsal Ayruolu and Billur Barshan, Reliability Measure Assignment to Sonar for Robust Target Differentiation, *Pattern Recognition*, Vol. 35, 2002, pp.1403-1419.
- [23] Billur Barshan and Birsal Ayruolu, Comparative Analysis of Different Approaches to Target Differentiation and Localization with Sonar, *Pattern Recognition*, Vol. 36, Issue 5, May 2003, pp.1213-1231.
- [24] Birsal Ayruolu and Billur Barshan, Identification of Target Primitives with Multiple Decision-making Sonars Using Evidential Reasoning, *The International Journal of Robotics Research*, Vol. 17, Issue 6, June 1998, pp. 598-623.
- [25] Billur Barshan and Birsal Ayruolu, Performance Comparison of Four Time-of-flight Estimation Methods for Sonar Signals, *Electronics Letters*, Vol. 34, Issue 16, August 1998, pp. 1616-1617.
- [26] S. W. Utete, B. Barshan and B. Ayruolu, Voting as Validation in Robot Programming, *The International Journal of Robotics Research*, Vol. 18, Issue 4, April 1999, pp. 401-413.

- [27] B. Barshan and A. S. Sekmen, Radius of Curvature Estimation and Localization of Targets Using Multiple Sonar Sensors, *The Journal of the Acoustical Society of America*, Vol. 105, Issue 4, April 1999, pp. 2318-2331.
- [28] D. Baskent and Billur Barshan, Surface Profile Determination from Multiple Sonar Data Using Morphological Processing, *The International Journal of Robotics Research*, Vol. 18, Issue 8, August 1999, pp. 788-808.
- [29] Billur Barshan, Ultrasonic Surface Profile Determination by Spatial Voting, *Electronics Letters*, Vol. 35, Issue 25, December 1999, pp. 2232-2234.
- [30] Billur Barshan, Fast Processing Techniques for Accurate Ultrasonic Range Measurements, *Measurement Science and Technology*, Vol. 11, Issue 1, January 2000.
- [31] B. Barshan and D. Baskent, Comparison of Two Methods of Surface Profile Extraction from Multiple Ultrasonic Range Measurements, *Measurement Science and Technology*, Vol. 11, Issue 6, June 2000, pp. 833-844.
- [32] B. Barshan, B. Ayruolu and S. W. Utete, Neural Network-based Target Differentiation Using Sonar for Robotics Applications, *IEEE Transactions on Robotics and Automation*, Vol. 16, Issue 4, August 2000, pp. 435-442.
- [33] Birsal Ayruolu and Billur Barshan, Neural Networks for Improved Target Differentiation and Localization with Sonar, *Neural Networks*, Vol. 14, Issue 3, April 2001, pp. 355-373.
- [34] A. S. Sekmen and B. Barshan, Estimation of Object Location and Radius of Curvature Using Ultrasonic Sonar, *Applied Acoustics*, Vol. 62, Issue 7, July 2001, pp. 841-865.
- [35] Billur Barshan and Birsal Ayruolu, Fractional Fourier Transform Preprocessing for Neural Networks and its Application to Object Recognition, *Neural Networks*, Vol. 15, Issue 1, January 2002, pp. 131-140.
- [36] Billur Barshan and Birsal Ayruolu, Fuzzy Clustering and Enumeration of Target Type Based on Sonar Returns, *Pattern Recognition*, Vol. 37, Issue 2, February 2004, pp. 189-199.
- [37] Birsal Ayruolu and Billur Barshan, Reliability Measure Assignment to Sonar for Robust Target Differentiation, *Pattern Recognition*, Vol. 35, Issue 6, June 2002, pp. 1403-1419.
- [38] T. Aytaç and B. Barshan, Differentiation and Localization of Targets Using Infrared Sensors, *Optics Communications*, Vol. 210, Issue 1-2, September 2002, pp. 25-35.

- [39] T. Aytaç and B. Barshan, Rule-based Target Differentiation and Position Estimation Based on Infrared Intensity Measurements, *Optical Engineering*, November 2002.
- [40] Andrew Heale and Lindsay Kleeman, Fast Target Classification Using Sonar, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hawaii, USA, October 2001, pp. 1446-1451.
- [41] Lindsay Kleeman, Fast and Accurate Sonar Trackers Using Double Pulse Coding, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, Korea, October 1999, pp. 1185-1190.
- [42] Kok S. Chong and Lindsay Kleeman, Mobile Robot Map Building from an Advanced Sonar Array and Accurate Odometry, *The International Journal of Robotics Research*, Vol. 18, Issue 1, January 1999, pp. 20-36.
- [43] Lindsay Kleeman, Advanced Sonar and Odometry Error Modeling for Simultaneous Localization and Map Building, *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003, pp. 699-704.
- [44] Lindsay Kleeman, Real Time Mobile Robot Sonar with Interference Rejection, *Sensor Review*, Vol. 19, Issue 3, 1999, pp. 214-221.
- [45] K. S. Chong and L. Kleeman, Feature-based Mapping in Real, Large Scale Environments Using an Ultrasonic Array, *The International Journal of Robotics Research*, Vol. 18, Issue 1, January 1999, pp. 3-19.
- [46] R. A. Russell, L. Kleeman and S. Kennedy, Using Volatile Chemicals to Help Locate Targets in Complex Environments, *Australian Conference on Robotics and Automation*, Melbourne, August 30 - September 1, 2000, pp 87-92.
- [47] Price, G. Taylor and Lindsay Kleeman, Fast, Robust Colour Vision for the Monash Humanoid, *Australian Conference on Robotics and Automation*, Melbourne, August 30 - September 1, 2000, pp. 141-146.
- [48] Heale and Lindsay Kleeman, A Real Time DSP Sonar Echo Processor, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, October 2000, pp. 1261-1266.
- [49] Heale and Lindsay Kleeman, Fast Target Classification Using Sonar, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hawaii, USA October 2001, pp. 1446-1451.
- [50] Lindsay Kleeman, Advanced Sonar Sensing, *Proceedings 10th International Symposium Robotics Research*, November 2001, Lorne Victoria Australia, pp. 286-295.

- [51] G. Taylor and L. Kleeman, Flexible Self-calibrated Visual Servoing for a Humanoid Robot, *Proceedings of the Australian Conference on Robotics and Automation 2001*, Sydney, November 2001, pp. 79-84.
- [52] Lindsay Kleeman, Advanced Sonar with Velocity Compensation, *The International Journal of Robotics Research*, Vol. 23, Issue 2, 2004, pp.111-126.
- [53] John J. Leonard and Hugh F. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers.
- [54] Itiel E. Dior, Mark Zagaeski and Cynthia F. Moss, Three-dimensional Target Recognition via Sonar: a Neural Network Model, *Neural Networks*, Vol. 8, Issue 1, 1995, pp.149-160.
- [55] K. Madhava Krishna and Prem K. Kalra, Detection, Tracking and Avoidance of Multiple Dynamic Objects, *Journal of Intelligent and Robotic Systems*, Vol. 33, 2002, pp. 371-408.
- [56] Phillip J. McKerrow and Neil L. Harper, Recognizing Leafy Plants with In-air Sonar, *Sensor Review*, Vol. 19, Issue 3, 1999, pp. 202-206.
- [57] Phillip McKerrow and Neil Harper, Plant Acoustic Density Profile Model of CTFM Ultrasonic Sensing, *IEEE Sensors Journal*, Vol. 1, Issue 4, December 2001, pp. 245-255.
- [58] Neil Harper and Phillip McKerrow, Recognizing Plants with Ultrasonic Sensing for Mobile Robot Navigation, *Robotics and Autonomous Systems*, Vol. 34, 2001, pp. 71-82.
- [59] Danny Ratner and Phillip McKerrow, Navigating an Outdoor Robot along Continuous Landmarks with Ultrasonic Sensing, *Robotics and Autonomous Systems*, Vol. 45, 2003, pp. 73-82.
- [60] James L. Crowley, Navigation for an Intelligent Mobile Robot, *IEEE Journal of Robotics and Automation*, Vol. RA-1, Issue 1, March 1985, pp. 31-41.
- [61] Ricardo Carelli and Eduardo O. Freire, Corridor Navigation and Wall-following Stable Control for Sonar-based Mobile Robots, *Robotics and Autonomous Systems*, Vol. 45, 2003, pp. 235-247.
- [62] Tsung Nan Chou and Catherine Wykes, An Integrated Ultrasonic System for Detection, Recognition and Measurement, *Measurement*, Vol. 26, Issue 3, October 1999, pp. 179-190.
- [63] A. M. Flynn, Combining Sonar and Infrared Sensors for Mobile Robot Navigation, *The International Journal of Robotics Research*, Vol. 7, Issue 6, December 1988, pp. 5-13.

- [64] Richard E. Crandall, *Projects in Scientific Computation*, Springer-Verlag, 2000.
- [65] A. Rosenfeld, *Picture Processing by Computer*, New York: Academic Press, 1969, sect. 6.1.
- [66] R. Nagel and A. Rosenfeld, Ordered Search Techniques in Template Matching, *Proceedings of the IEEE*, Vol. 60, February 1972, pp. 242-244.
- [67] D. I. Barnea and H. F. Silverman, A Class of Algorithms for Fast Digital Image Registration, *IEEE Transactions on Computers*, Vol. c-21, Issue 2, February 1972, pp.179-186.
- [68] G. J. Vanderbrug and A. Rosenfeld, Two Stage Template Matching, *IEEE Transactions on Computers*, Vol. c-26, Issue 4, April 1977, pp. 384-393.
- [69] A. Rosenfeld and G. J. Vanderbrug, Coarse-fine Template Matching, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-7, February 1977, pp. 104-107.
- [70] A. Rosefeld and A. C. Kak, *Digital Picture Processing*, 2nd Edition, Vol. 2, Chapter 9, Academic Press, Orlando, 1982.
- [71] J. K. Cheng and T. S. Huang, Image Registration by Matching Relational Structures, *Pattern Recognition*, Vol. 17, Issue 1, 1984, pp. 149-159.
- [72] S. L. Tanimoto, Template Matching in Pyramids, *Computer Graphics and Image Processing*, Vol. 16, 1981, pp. 356-369.
- [73] S. P. Smith and A.K. Jain, Chord Distributions for Shape Matching, *Computer Graphics and Image Processing*, Vol. 20, 1982, pp. 259-271.
- [74] T. G. Ryall and J. Sandor, Statistical Pattern Matching, *Pattern Recognition Letters*, Vol. 9, 1989, pp.163-168.
- [75] A. L. Yuille and P. W. Hallinan, and D. S. Cohen, Feature Extraction from Faces Using Deformable Templates, *International Journal of Computer Vision*, Vol. 8, Issue 2, 1992, pp. 99-111.
- [76] R. Brunelli and T. Poggio, Face Recognition: Features Versus Templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, Issue 10, October 1993, pp. 1042-1052.
- [77] J. Y. Deng and F. Lai, Region-based Template Deformation and Masking for Eye-feature Extraction and Description, *Pattern Recognition*, Vol. 30, Issue 3, 1997, pp. 403-419.

- [78] J. Miao, B. Yin, K. Wang, L. Shen, and X. Chen, A Hierarchical Multiscale and Multiangle System for Human Face Detection in a Complex Background Using Gravity-center Template, *Pattern Recognition*, Vol. 32, 1999, pp. 1237-1248.
- [79] T. Kondo and H. Yan, Automatic Human Face Detection and Recognition under Non-uniform Illumination, *Pattern Recognition*, Vol. 32, 1999, pp. 1707-1718.
- [80] C. L. Su, Robotic Intelligence for Industrial Automation: Object Flaw Auto Detection and Pattern Recognition by Object Location Searching, Object Alignment and Geometry Comparison, *Journal of Intelligent and Robotic Systems*, Vol. 33, 2002, pp. 437-451.
- [81] T. Kawaguchi and M. Rizon, Iris Detection Using Intensity and Edge Information, *Pattern Recognition*, Vol. 36, 2003, pp. 549-562.
- [82] A. K. Jain, Y. Zhong and S. Lakshmanan, Object Matching Using Deformable Templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, Issue 3, March 1996, pp. 267-278.
- [83] A. K. Jain and A. Vailaya, Shape-based Retrieval: a Case Study with Trademark Image Databases, *Pattern Recognition*, Vol. 31, Issue 9, 1998, pp. 1369-1390.
- [84] Y. Zhong and A. K. Jain, Object Localization Using Color, Texture and Shape, *Pattern Recognition*, Vol. 33, 2000, pp. 671-684.
- [85] C. H. Chou and Y. C. Chen, Moment-preserving Pattern Matching, *Pattern Recognition*, Vol. 23, Issue 5, 1990, pp. 461-474.
- [86] M. Uenohara and T. Kanade, Use of Fourier and Karhunen-Loeve Decomposition for Fast Pattern Matching with a Large Set of Templates, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 19, Issue 8, August 1997, pp. 891-898.
- [87] M. S. Choi and W.Y. Kim, A Novel Two Stage Template Matching Method for Rotation and Illumination Invariance, *Pattern Recognition*, Vol. 35, 2002, pp. 119-129.
- [88] D. M. Lane and John P. Stoner, Automatic Interpretation of Sonar Imagery Using Qualitative Feature Matching, *IEEE Journal of Oceanic Engineering*, Vol. 19, Issue 3, July 1994, pp. 391-405.
- [89] M. Mignotte, C. Collet, P. Perez and P. Bouthemy, Hybrid Genetic Optimization and Statistical Model-based Approach for the Classification of Shadow Shapes in Sonar Imagery, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, Issue 2, February 2000, pp. 129-141.
- [90] F. Rosenblatt, The Perceptron: a Theory of Statistical Separability in Cognitive Systems. *Cornell Aeron.Lab., Rep. VG-1196-G-1*, Jan., 1958.

- [91] L. N. Kanal and A. Rosenfeld, *Progress in Pattern Recognition 2*, Elsevier Science Publishers B.V.
- [92] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Inc.
- [93] J. T. Tou, Feature Extraction in Pattern Recognition, *Pattern Recognition*, Vol. 1, Issue 1, July 1968, pp. 3-11.
- [94] Theo Pavlidis, 36 years on the Pattern Recognition Front, *Pattern Recognition Letters*, Vol. 24, 2003, pp. 1-7.
- [95] J. P. Maroy and M. Berthod, Natural Language Understanding by a Robot: A Pattern Recognition Problem, *Pattern Recognition*, Vol. 10, Issue 2, 1978, pp. 63-71.
- [96] B. Moayer and K.S. Fu, A Syntactic Approach to Fingerprint Pattern Recognition, *Pattern Recognition*, Vol. 7 Issues 1-2, June 1975, pp. 1-23.
- [97] Robert.M. Haralick, Structural Pattern Recognition, Homomorphisms, and Arrangements, *Pattern Recognition*, Vol.10, Issue 3, 1978, pp. 223-236.
- [98] Maher Ahmed and Rabab Kreidieh Ward, An Expert System for General Symbol Recognition, *Pattern Recognition*, Vol. 33, Issue 12, December 2000. pp. 1975-1988.
- [99] P. N. Suganthan, Structural Pattern Recognition: Using Genetic Algorithms, *Pattern Recognition*, Vol. 35, Issue 9, September 2002, pp. 1883-1893.
- [100] Juliusz L. Kulikowski, From Pattern Recognition to Image Interpretation, *Biocybernetics and Biomedical Engineering*, 2002, Vol. 22, Issue 2-3, pp. 177-197.
- [101] Matteo Pardo and Giorgio Sberveglieri, Learning from Data: A Tutorial with Emphasis on Modern Pattern Recognition Methods, *IEEE Sensors Journal*, Vol. 2, Issue 3, June 2002, pp. 203-217.
- [102] M. P. C. McQueen, A Generalization of Template Matching for Recognition of Real Objects, *Pattern Recognition*, Vol. 13, Issue 2, 1981, pp. 139-145.
- [103] Thomas G. Ryall and John Sandor, Statistical Pattern Matching, *Pattern Recognition Letters*, Vol. 9, 1989, pp. 163-168.
- [104] Raj Bhatnagar, Richard Horvitz and Rob Williams, A Hybrid System for Target Classification, *Pattern Recognition Letters*, Vol. 8, 1997, pp. 1399-1403.

- [105] Mostafa G. Mostafa, Elsayed E. Hemayed and Aly A. Farag, Target Recognition via 3D Object Reconstruction from Image Sequence and Contour Matching, *Pattern Recognition Letters*, Vol. 20, 1999, pp.1381-1387.
- [106] Martin Kermit and Age J. Eide, Audio Signal Identification via Pattern Capture and Template Matching, *Pattern Recognition Letters*, Vol. 21, 2000, pp. 269-275.
- [107] Lesley M. Murphy, Linear Feature Detection and Enhancement in Noisy Images via the Radon Transform, *Pattern Recognition Letters*, Vol. 4, 1986, pp. 279-284.
- [108] M.E. Diaz, J. Domingo and G. Ayala, A Gray-level 2D Feature Detector Using Circular Statistics, *Pattern Recognition Letters*, Vol. 18, 1997, pp. 1083-1089.
- [109] Yingjie Wang, Chin-Seng Chua and Yeong-Khing Ho, Facial Feature Detection and Face Recognition from 2D and 3D Images, *Pattern Recognition Letters*, Vol. 23, 2002, pp. 1191-1202.
- [110] Mark Fiala and Anup Basu, Hough Transform for Feature Detection in Panoramic Images, *Pattern Recognition Letters*, Vol. 23, 2002, pp. 1863-1874.
- [111] Lin-lin Huang, Akinobu Shimizu, Yoshihoro Hagihara and Hidefumi Kobatake, Gradient Feature Extraction for Classification-based Face Detection, *Pattern Recognition*, Vol. 36, 2003, pp. 2501-2511
- [112] K. Madhava Krishna and Prem K. Kalra, Detection, Tracking and Avoidance of Multiple Dynamic Objects, *Journal of Intelligent and Robotic Systems*, Vol. 33, 2002, pp. 371-408.
- [113] Tsung Nan Chou and Catherine Wykes, An Integrated Ultrasonic System for Detection, Recognition and Measurement, *Measurement*, Vol. 26, Issue 3, October 1999, pp. 179-190.
- [114] M. Lach and H. Ermert, An Acoustic Sensor System for Object Recognition, *Sensors and Actuators, A*, 25-27, 1991, pp. 541-547.
- [115] J. R. Llata, E. G. Sarabia and J. P. Oria, Three-dimensional Robotic Vision Using Ultrasonic Sensors, *Journal of Intelligent and Robotic Systems*, Vol. 33, 2002, pp. 267-284.
- [116] David M. Lane and John P. Stoner, Automatic Interpretation of Sonar Imagery Using Qualitative Feature Matching, *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 3, July 1994, pp. 391-405.
- [117] John J. Leonard and Hugh F. Durrant-Whyte, Mobile Robot Localization by Tracking Geometric Beacons, *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, June 1991, pp. 376-382.

- [118] Yoshiaki Nagashima and Shin'ichi Yuta, Ultrasonic Sensing for a Mobile Robot to Recognize an Environment — Measuring the Normal Direction of Walls, *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, NC, July 7-10, 1992, pp. 805-812.
- [119] Kai-Tai Song and Charles C. Chang, Ultrasonic Sensor Data Fusion for Environment Recognition, *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan July 26-30, 1993, pp. 384-390.
- [120] Robert J. Dickenson and Z. Ghassemlooy, A Feature extraction and Pattern Recognition Receiver Employing Wavelet Analysis and Artificial Intelligence for Signal Detection in Diffuse Optical Wireless Communications, *IEEE Wireless Communications*, Vol. 10, Issue 2, 2003, pp. 64-72.
- [121] William W. Streilein, Paolo Gaudiano and Gail A. Carpenter, A Neural Network for Object Recognition Through Sonar on a Mobile Robot, *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, Gaithersburg, MD, September 14-17, 1998. pp. 271-276.
- [122] Michael Drumheller, Mobile Robot Localization Using Sonar, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 2, March 1987, pp. 325-332.
- [123] Brian Yamauchi, Alan Schultz and William Adams, Mobile Robot Exploration and Map-building with Continuous Localization, *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium, May 1998, pp. 3715-3720.
- [124] Huzefa Akbarally and Lindsay Kleeman, A Sonar Sensor for Accurate 3D Target Localization and Classification, *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, May 21-27, 1995, Nagoya, Aichi, Japan, pp. 3003-3008.
- [125] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 1998.
- [126] Ruo Feng, *Ultrasound Handbook*, 1999.

Vita

Wen Gao

Wen Gao was born in Changzhou, China on November 14, 1973. She received her Bachelor of Science degree in Applied Physics from Fudan University, Shanghai, China in 1997. During the same year, Wen entered the master's program at the Institute of Acoustics in Tongji University, Shanghai, China. In the year 2000, she graduated with honors and entered the College of William and Mary in Virginia as a doctoral student in the Department of Applied Science. She received her Master of Science degree in Applied Science in 2002. Wen successfully defended her dissertation on April 15, 2005. She currently lives in Austin, Texas with her husband, Yongming.