To the Graduate Council:

I am submitting herewith a thesis written by Mir Shariful Azam entitled "Robust and Efficient Surface Interpolation Algorithms for Sonar Based Obstacle Negotiation of a Mobile Robot." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Mohan M. Trivedi

Dr. Mohan Trivedi, Major Professor

We have read this thesis
and recommend its acceptance:

Marshall Pace

Igor Alexeff

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

Robust and Efficient Surface Interpolation Algorithms for Sonar Based Obstacle Negotiation of a Mobile Robot

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Mir S. Azam

December, 1994

To the ones who truly love me.

Acknowledgments

My parents have been the key individuals helping me with my efforts to find directions in life. True and sincerest thanks goes to them. I am deeply indebted to Mom for her continuous support and faith she has shown towards me at all times. I would also like to thank Dr. Trivedi for his friendship and guidance over the past two years. I appreciate his letting me live with my own judgments. I wish to thank Dr. Igor Alexeff and Dr. Marshall Pace for their suggestions, support and time. Special thanks goes to Dr. Joyce Blair for caring, for believing in me. Thanks to Dr. ChuXin Chen for his continued concern for me even after he left our lab a year ago. I sincerely thank Mr. Nils Lassiter for helping me with every project that I worked on at the CVRR lab. I truly appreciate all the events I shared with the members of the CVRR lab—the discussions we had on technical and philosophical issues, the valuable help I received from from some of the members to complete this thesis, and the jokes we pulled on one another.

Abstract

Surface interpolation using discrete and insufficient range data is the major emphasis of this thesis. The underlying idea for the interpolation is to have a mobile robot make decisions while navigating in obstacle ridden environments. The interpolation techniques chosen and developed are kept computationally simple. Two dimensional polynomials serve as the basis of this surface interpolation technique; developing a new surface interpolation algorithm involves using those polynomials on the sampled surface plane. Three different polynomials—the Lagrange polynomial, the Piecewise Linear polynomial, and the Cubic Spline polynomial—are used to interpolate surfaces. Ultrasonic range sensors are used to obtain the range data, and careful sensor placement is necessary to interpolate surfaces. The technique developed is thoroughly tested with both synthetic and real data. Experiments are performed on available hardware such as a mobile robot and also on developed hardware such as the sensory template built for this thesis. In order to be able to do the experiments, integration between the sensory subsystem and the robotic subsystem plays an important role. Graphical user interfaces are built in order to have both teleoperation and automatic navigation modes for the mobile robot, for the three dimensional approximation technique utilities, and for plotting surfaces in three dimensions. Finally, comparisons among the interpolation techniques are carried out based on the obtained results from the analyses.

Using the surface interpolation technique developed here for this thesis, the Lagrange polynomial and the Piecewise Linear polynomial perform close to each other,

each one's results staying within 2% of those of the other one. The Cubic Spline polynomial based approach, however, deviates considerably from Lagrange and Linear algorithms based approaches by giving 5 or 6 times worse mean errors in some cases. Nevertheless, the three dimensional shapes of the original surfaces are preserved by all the polynomials. The unique contributions of this thesis are the less computation demand for surface interpolation, an extensive set of experiments over at least five different synthetic surfaces and at least two different real surfaces, friendly graphical user interfaces, and the implementation of a polynomial based reactive obstacle negotiation algorithm on a mobile robot in completely unknown but obstacle cluttered environments.

TABLE OF CONTENTS

CHAPTER		PAGE
1.	**Introduction, Problem Statement, and Motivation**	**1**
1.1	Interpolation and Reaction in Real Environments	3
1.2	Robot Navigation in Unknown Environments	6
1.2.1	Decision to Avoid or Negotiate Obstacles	7
1.2.2	Make Turns while Negotiating Obstacles	9
1.2.3	Approach Perpendicular to the Obstacle	9
1.3	Obstacle Negotiation with R2M2	10
1.4	Summary	13
2.	**The Ranging System**	**15**
2.1	The Ultrasonic Range Sensors	15
2.2	Principle Behind Ultrasonic Sound Based Ranging: The "Chirping" Concept	16
2.3	Description of the Sonar Ranging Module	22
2.4	Integrated System	24
2.5	Summary	26
3.	**Issues in Sensor Positioning**	**28**
3.1	Square Array Organization(SAO)	29
3.2	Hexagonal Organization of Sensory Template(HOST)	31

CHAPTER PAGE

3.3 Choosing One Sensor Array Type: The SAO Structure 32

3.4 Summary 34

4. Curve Interpolation 35

4.1 3-D Surface fitting 36

4.2 2-D Polynomial Interpolation 40

5. Interpolation Algorithms Basics 43

5.1 The Lagrange Algorithm 43

5.1.1 Lagrange Algorithm Computation Technique 46

5.2 Piecewise Linear Algorithm 50

5.2.1 Piecewise Linear Algorithm Computation Technique 52

5.3 Cubic Spline Algorithm 52

5.3.1 Boundary Conditions for Cubic Spline Polynomial 56

5.3.2 Computation Technique of Cubic Spline: 58

5.4 Comparing the Algorithms 60

5.5 The Chebyshev Polynomial 62

5.6 Summary 65

6. Surface Generation 66

6.1 Technique involved in Surface Function Generation 68

6.2 Masking 75

6.3 Summary 78

7. Experimental Validation Using Hardware 79

CHAPTER PAGE

7.1 Surface Interpolation from Real Range Data 80
7.2 Turning of R2M2 on Obstacles 81
7.3 Implementation of the Obstacle Negotiation Algorithm on R2M2 . . . 82
7.4 Summary 85

8. Graphical User Interface 86
8.1 The Drawing Area 87
8.2 The Panel for Utilities 91
8.3 Real Time graphical Controller for R2M2 94
8.4 Summary 95

9. Results Using Surface Interpolation 97
9.1 Plotting Surfaces 97
9.2 Error Estimation 98
9.2.1 Testing on Synthetic Surfaces 101
9.2.2 Testing on real surfaces 101
9.3 Summary 109

10. Concluding Remarks and Future Research 113
10.1 Achievements 114
10.2 Future Research Plans 115
10.3 Summary 116

BIBLIOGRAPHY 117

CHAPTER PAGE

VITA 121

LIST OF TABLES

TABLE		PAGE
5.1	*Newton's Divided Difference Table.*	49
9.1	*Error estimation for an exponential surface described by* $z = e^{-4x^2-4y^2}$.	102
9.2	*Error estimation for an product surface described by* $z = x \times y$.	102
9.3	*Error estimation for a plane described by* $z = const$.	102
9.4	*Error estimation for a ditch surface type.*	103
9.5	*Error estimation of stair-step surface.*	103

LIST OF FIGURES

FIGURE **PAGE**

1.1 *R2M2 robotic system in a typical obstacle ridden environment. The range information drawn with arrows in the diagram need to be interpolated before making decisions for the robot.* 3

1.2 *The underlying effect of interpolation and reaction in the sensory subsystem and the robotic subsystem of an intelligent machine.* 4

1.3 *Different issues in robot navigation in unknown environment. In figure (a) a decision needs to be made to move on a uniform surface. Figure (b) portrays the need for taking a turn on a ramp. And figure (c) shows the importance of robot positioning with respect to the obstacle.* 8

1.4 *The robot R2M2.* 11

1.5 *The robot R2M2.* 13

2.1 *The "chirping" concept. The sound wave travels from point A of the sensor, reflects off of the object in the way, and finally returns to point B of the sensor. Note that the distance traveled by the sound wave is not exactly equal to twice the distance of the object from the sensor.* . 18

2.2 *The multiple reflection of the ultrasonic sound as it reflects off two surfaces before the sensor hears the echo.* 20

2.3 *Range sensing being mislead by a convex corner adjoining two walls.* . 21

2.4 *Component layout of the SN28827 module for the transducers.* 23

FIGURE **PAGE**

2.5 *Integrated hardware system for surface generation.* 26

3.1 *The Square Array Organization(SAO) of the sensors. (a) The basic organization in a 3 by 3 array, (b) the masking after upsampling the data, and (c) the repeatability of the SAO.* 30

3.2 *The Hexagonal Organization of Sensory Template(HOST). Figure (a) shows the basic structure, and figure (b) illustrates the repeatability of the basic structure.* 33

5.1 *The Lagrange interpolating polynomial. Note the oscillation at the end-points. The dark dots are measured data points.* 44

5.2 *The Piecewise linear function going through a set of data points. This linear polynomial simply draws straight lines between pairs of points.* . 51

5.3 *The General Cubic Spline Polynomial connecting a sub-interval from point A to point B. The "upswing" and "downswing" are labeled in the diagram. This figure illustrates the polynomial's capability to adjust itself to faster changes in the data points through multi-swing features.* 54

5.4 *Comparison among the three polynomials going through the same points.* 61

6.1 *The system block diagram from an input/output(I/O) perspective.* . . 67

6.2 *The Polynomial Vector Directions in a 3 by 3 square array. The direction angle for rows are* $0°$, *for columns the angle is* $90°$, *and for diagonals it is* $45°$. 71

6.3 *The elements of the Weight Vector(WV) generation. The dark dot in the images show the location at which depth is to be estimated. (b) is a zoomed version of a circular region in (a). Figure (c) is a zoomed portion of (b) where the WV component generating elements* d_{row}, d_{ol}, *and* d_{dia} *are shown.* 74

6.4 *The updating of the mask. The dark dot in each figure represents the goal point at which depth is to be extracted. Figure(a) is the first step of masking where the entire region is considered. (b), (c), and (d) are the smaller masks around the goal point to get better polynomial and weight vectors.* 76

6.5 *The algorithm: the major steps to generate depth.* 77

7.1 *Incorporating range and inclinometer information.* 82

7.2 *Two dimensional obstacle negotiation of R2M2. Figure(a) shows climbing over a log as seen by the measured or dotted points. Figure(b) is negotiating a ramp, while Figures (c) and (d) are climbing stairs. Note how a curve is fit through the data points in each figure.* 84

8.1 *The transformation of only one coordinate(in this case, the x coordinate) of the 3-D world into the 2-D world.* 88

8.2 *An (x,y,0) point being mapped on the drawing plane as (x,y).* 90

8.3 *The control panel for XMir.* 93

8.4 *The graphical controller "r2m2" for the robot R2M2.* 95

9.1 *An approximation result. The figures at left show original surfaces of the exponential type. And the figures at right show the interpolated or approximated surfaces after initially sampling only 9 points from the original surfaces as shown by the dark dots. The top-right diagram is the result of a Lagrange polynomial based interpolation, and the bottom-right surface is the Piecewise Linear based interpolation of the respective surfaces at left.* 99

9.2 *Surface interpolation of an exponential surface using Cubic Spline based approach(on top), and approximation based on Lagrange algorithm for a product surface.* 104

9.3 *Product surface estimation by Piecewise Linear(top) and Cubic Spline(bottom) algorithms.* 105

9.4 *Approximating a ditch with Lagrange based algorithm at top, and with Linear algorithm at bottom.* 106

9.5 *Cubic Spline algorithm on a ditch function, and Lagrange on interpolation on a stair-step.* 107

9.6 *a stair-step function approximated by Linear and Cubic Spline algorithms.* 108

9.7 *The approximation of an edge using real range data using the Lagrange and the linear interpolations.* 110

9.8 *The approximation of an edge using the Cubic Spline approach, and Lagrange based interpolation of a board partially in the view in front of a wall using real range data.* 111

9.9 *A board placed in front of a wall. Real data approximation using the Linear and the Cubic Spline approach.* 112

CHAPTER 1

Introduction, Problem Statement, and Motivation

The principal idea behind this thesis is to explore the methods of three dimensional surface interpolation. The surface interpolation techniques were studied by using distance information from the sensors to the various regions of the surface of interest. The ultimate purpose of this sensory system is to develop and integrate a mobile robot with autonomous navigation and obstacle negotiation capabilities. In order to achieve increased autonomous mobility of a mobile robot in all three dimensions, an approach is being taken by using distance measurements from a local plane on the robot to approximate three dimensional surfaces fast—within 100 milliseconds, when the next set of data is collected from the sensory module.

The need for data interpolation in general, especially surface interpolation, is for various reasons and in a variety of applications as well. The concept of interpolation is set up on insufficient data, with less than a dozen data points on a square foot of surface plane. Some of the more sophisticated surface interpolation algorithms need to start with at least about 150 data points [1]. When data in certain regions are incomplete or considered insufficient, interpolation provides those estimates to allow further processing. The output of the data processing obviously depends on how well the missing data were interpolated. And the performance of the processing thus becomes closely related to the interpolation techniques.

There is another aspect of interpolation that applies to practical situations. Practical applications often become time critical. That is, interpolation and processing often need to be done fast enough so that the required task is done in the real world. Some interpolation and processing technique might be very sophisticated, but could still be useless for application simply because the results were achieved too late to apply. Therefore, interpolation and processing sometimes need to be simple, even at the cost of accuracy, only to get useful results and reasonable performance.

Surface interpolation algorithms from range data can also be utilized for scene characterization [2], environment mapping [3], and environment recognition [4]. Anytime knowledge about an unknown environment or an unknown object is seeked, sensor integration with a surface fitting algorithm of the kind described in this thesis is helpful. Also, the problems associated with ultrasonic data being noisy due to the transmission wave cone can be treated with neural networks [4] for better surface interpolation. Therefore, the scope of a surface interpolation algorithm based on range data is open for applyning in different areas.

Consider the robotic system R2M2(Robotic Research Mobile Manipulator) available in the laboratory. R2M2 is a tracked mobile manipulator capable of moving on uneven surfaces by adjusting its tracks while moving forward in a teleoperated mode [5]. If the system is to be automated in order for it to move on rough terrains without human intervention, the sensory information obtained by the R2M2(the sensory information in this case being the range data) needs to be interpolated in a fashion that decisions can be made about the adjustments of the tracks. Figure 1.1 illustrates a typical unknown environment that R2M2 might run into. If navigation has to be

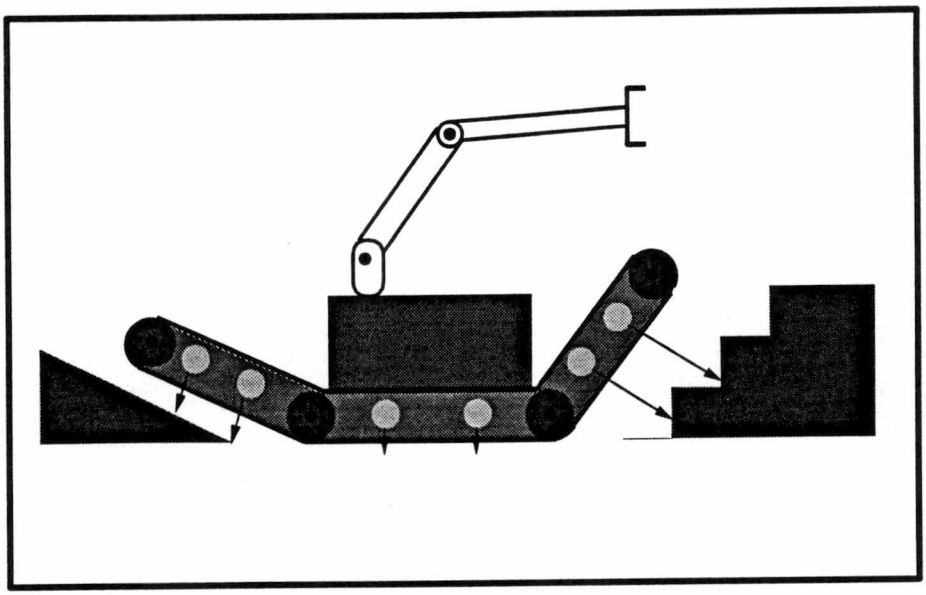


Figure 1.1: *R2M2 robotic system in a typical obstacle ridden environment. The range information drawn with arrows in the diagram need to be interpolated before making decisions for the robot.*

carried out in such an environment as described in Figure 1.1, the sensory range data must be interpolated by the system to understand the environment which allows the system to make decisions regarding its motion.

1.1 Interpolation and Reaction in Real Environments

Human beings are probably the best information interpolators without emphasizing on quantitative accuracy. Also, human beings are accustomed to respond reactively. Both of these hypotheses are going to be supported by examples in this section. And since people are the most intelligent forms known, it can be said that studies in the areas of interpolation and reactive-behavior are worth pursuing.

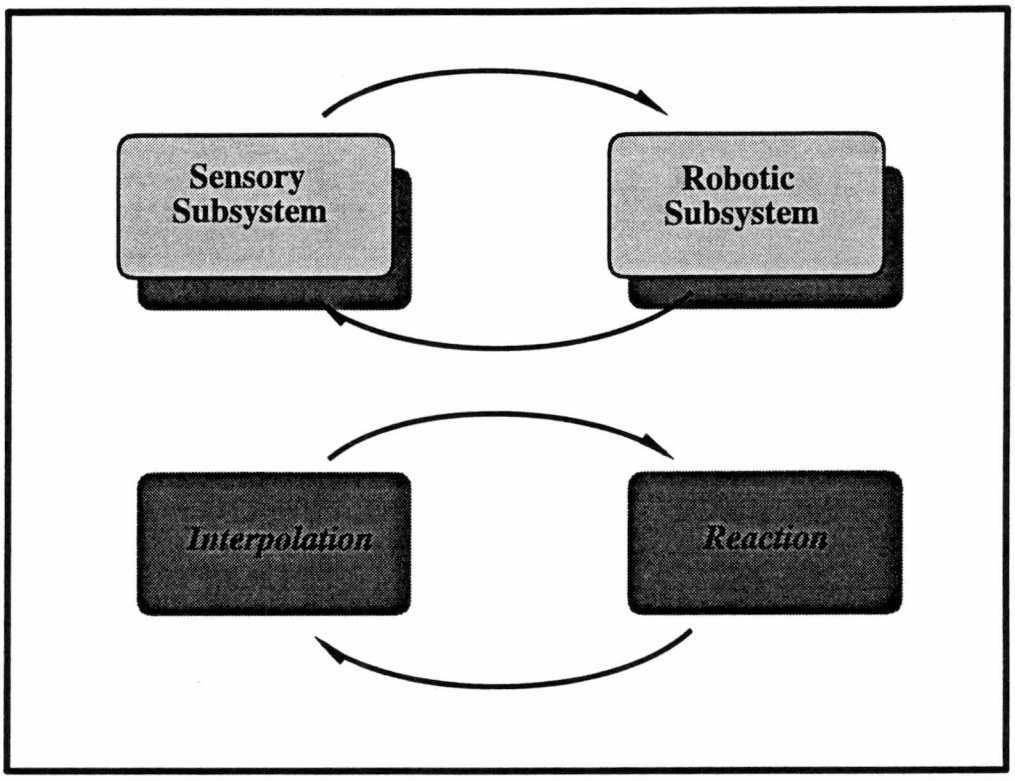


Figure 1.2: *The underlying effect of interpolation and reaction in the sensory subsystem and the robotic subsystem of an intelligent machine.*

Robotic systems can also be looked at from the perspective of interpolation and reaction. The sensory information available to the robot can be interpolated in order to establish the grounds for the decisions that the robot needs to make. Subsequently, the decisions taken by the robot based on localized sensory information can be interpreted as the reactive nature of the machine to the environment [6]. Figure 1.2 delineates this perspective to look at an intelligent robot. The following discussion describes the human nature similar to this interpolative and reactive nature in human beings.

Firstly, the interpolative nature of humans can be brought into attention. Consider

driving a car for example. When a person drives a car, he or she does not calculate beforehand exactly what acceleration to pursue when starting from rest, or exactly how far away is the car ahead of his or her, or precisely what radius of curvature to follow while taking a turn. People do not do those kinds of quantitative measurements. Rather, people's perceptions follow rules such as whether to accelerate fast or slow, whether to stay close to or far away from the car in front, or to take a sharp or wide turn. The quantitative measurements are only interpolated by the human brain, and those are transformed into motor skills. It is true that the speedometer provides exact speed information, but nobody looks at the speedometer at all times to maintain the desired speed. It is all a matter of interpolation in between real data—real data in this case being the speedometer reading. In the cases of maintaining a distance from the car in front or taking a turn, the information available is even more erroneous, provided only by instincts and experience. All the other information that are the inputs to the human brain controlling the act of driving are interpolated information.

Now comes the issue of the reactive nature of people. Take the same example as stated in describing data interpolation in everyday life—driving a vehicle, an everyday activity for almost everybody. A skilled and experienced driver has not been trained for all the situations that he or she faces on the road. The slowing down with the traffic, the changing lanes, driving on a road never driven before—all of these situations are taken care of reactively. Not a whole lot of emphasize on the accuracy of the quantitative data, or path planning, but these sorts of actions have plenty to do with adapting with the surrounding situations. When the car in front brakes, automatic reaction without pre-planning takes over the driver to hit the brakes; the empty lane

on the left or the right combined with a desire to change perhaps the velocity of traveling initiates lane changes without pre-arrangement; making the turns on an unknown road as the curves on the road approach—again, without significant or conscious path planning. It is reacting to the situation! No one tells us exactly what is coming in the future, but still we advance through the future by reacting to it. This does not suggest that the future should not have any plans whatsoever, this perception merely delineates that thorough and exhaustive planning is often redundant—in some cases, fruitless.

1.2 Robot Navigation in Unknown Environments

Robot navigation in an unknown environment requires sensor integration with the motor controls of the mobile robot [7]. Sensory information needs to be interpreted in three dimensions in order for the robot to make any decision about its direction of motion at every sensory data input. The common task of a mobile robot can be defined as *going from point A to point B.* Therefore, in the middle of the robot's navigation path, localized decisions need to be made to complete the navigation successfully.

By referring to localized decisions, it is generally meant that the robot runs in front of different obstacles, and it has to make decisions to how these obstacles can be *avoided* or *negotiated* to continue the desired motion. In other words, the robot has to "react" to the obstacles and update its path along the way. This reactive motion of the robot, without any doubt, needs the surface information on and around its path.

If the robot needs to avoid an obstacle, it needs to know how the outside surface contour facing the robot looks like at every time. Especially, if the robot decides to "negotiate" or *climb over* an obstacle, the path contour must be known. The robot must be aware of this path-surface because, in most cases, the robot would require to adjust its body structure to the path it is moving on. In other words, the robot must try to have maximum traction between its body and the path of motion. Thus surface approximation is crucial to mobile robot navigation in unstructured environments.

The need for determining surface structure is divided into three basic requirements in this thesis. They are,

- Deciding to avoid or negotiate obstacles,
- Making turns while negotiating obstacles, and
- Approaching perpendicular to negotiable obstacles for successful negotiation.

1.2.1 Decision to Avoid or Negotiate Obstacles

In an ideal environment with obstacles lined up with the mobile robot, one needs only two dimensional information in order to achieve successful obstacle negotiation. But, as it is a common knowledge, obstacles in an unstructured environment would not be lined up with the robot in a lot of cases. Thus, while approaching in such a manner that the robot sees both the obstacle and the free space to the side, the robot can decide, based on its geometric freedom, whether to climb over the obstacle or to advance around it. Figure 1.3(a) shows the possible local paths for a robot confronting an obstacle. And in order to make this decision, the robot needs to be

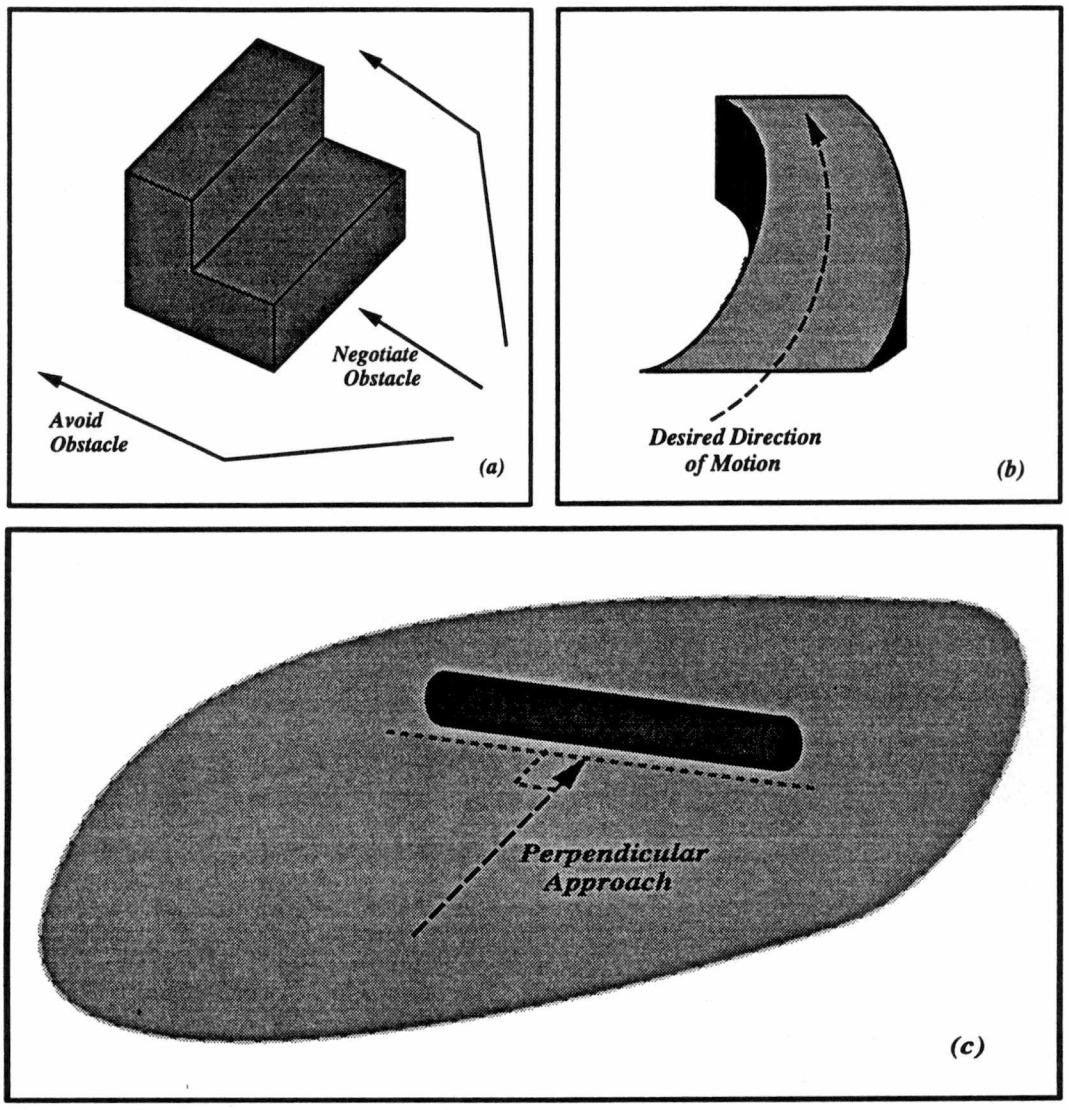


Figure 1.3: *Different issues in robot navigation in unknown environment. In figure (a) a decision needs to be made to move on a uniform surface. Figure (b) portrays the need for taking a turn on a ramp. And figure (c) shows the importance of robot positioning with respect to the obstacle.*

aware of the surface geometry of the obstacle. If the obstacle is negotiable, the robot can approach for climbing.

1.2.2 Make Turns while Negotiating Obstacles

Another need for 3-D surface information emerges when a mobile robot is in the act of negotiating an obstacle. The obstacle surface might have permissible and non-permissible regions for the motion of the robot. The obstacle could also be of "twisted" type where the need to make a turn can occur for the robot. In Figure 1.3(b), such a need is shown in terms of a ramp structure that is spiral as well as gradually increasing in height. This sort of a situation can be very common when confronting uneven terrains such as ditches and ramps. Usually, in moving through such an uneven surface, the gradual change of slope in the path occurs not along a straight line but along some curvature generally defining the direction of motion. In order to follow the horizontally curved and gradually changing uneven surfaces, a robot would definitely need surface contour approximation to continue navigation.

1.2.3 Approach Perpendicular to the Obstacle

Approaching perpendicular to the obstacle in consideration is an extremely important issue if the robot is to successfully climb the object. If the robot's approach is not "straight ahead", or perpendicular, when the robot starts climbing, the robot risks a good chance of tipping over and cause expensive damage to itself and the surrounding environment. In other words, the likelihood of an "accident" happening rises immensely. Therefore, safe navigation over obstacles depend highly on the

approach to the obstacle by the mobile robot. In Figure 1.3(c), such a situation is characterized. There is a log on the ground that has appeared as an obstacle to be negotiated. If the robot approaches the log at a sharp angle, there is a possibility that the force acting at the center of gravity of the robot cannot be neutralized by the reactions because there may not be enough contact towards the front of the robot. And when that happens, the robot could tip over on its side and thus fail to perform its job.

1.3 Obstacle Negotiation with R2M2

A Reactive Obstacle negotiation algorithm is successfully implemented on the Robotics Research Mobile Manipulator(R2M2) available in the laboratory. R2M2 is a tracked mobile robot capable of moving on uneven surfaces. The robot is initially designed to be a teleoperated system, but further and current research is geared towards automating some of the tasks performed by a teleoperator with many difficulties. Figure 1.4 is a picture of the mobile robot R2M2.

For example, the teleoperator is usually sitting at a different location than the robot itself. That means the robot is probably not in the line of sight of the teleoperator. Now, the only sensory feedback that the operator gets is the video picture taken from the robot by a camera on R2M2. Therefore, the operator is trying to venture through a three dimensional world by looking at a two dimensional video pictures. Often, the operator misinterpretes the available information while trying to guess the depth information that he or she wants the robot to react to. This is a difficult task

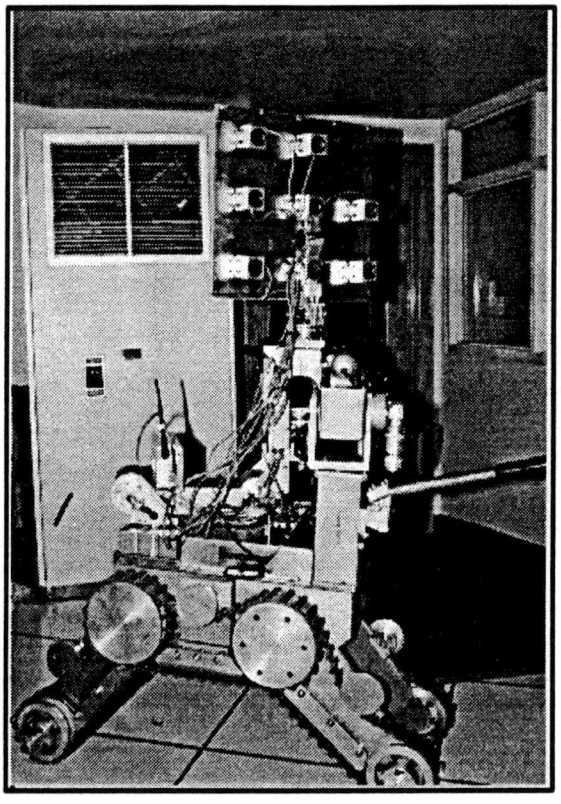


Figure 1.4: *The robot R2M2.*

unless range data is obtained and processed for decision making. Inconveniences for the operator include navigating on a path that has sudden changes on its surface—namely obstacles in the way, stairs, ditches, etc. If the operator approaches such a difficult navigation path, he or she would be enormously aided by the option of automatic obstacle negotiation. By the press of a button, the robot can make the decisions of adjusting its body to the uneven terrain while moving. After the path clears out, the operator can regain control of the robot and continue through his or her desired mission.

The motion of R2M2 is based on the three moving tracks on its structure. The track structure is divided into three relative positions. There is a front track and a rear track which scroll forwards and backwards as well as move upwards and downwards. The tracks on the main body of the robot can, however, only scroll forwards and backwards. Therefore, there are additional attributes associated with the front and the rear tracks compared to those of the main body track.

If the robot needs to climb over a higher object, the front tracks are raised first. This allows the front mobile tracks grab on the higher obstacle. When the robot climbs on the obstacle, the front and rear tracks could be adjusted through the up and down motions to set the body in the appropriate position for navigation on that terrain.

An obstacle negotiation algorithm based on a cubic polynomial was simulated [8] in a graphical environment [5]. That algorithm used 15 simulated range points with three simulated inclinometers on the tracks of R2M2. Global path coordinates were calculated using both the simulated ultrasonic sensor and the simulated inclinometer

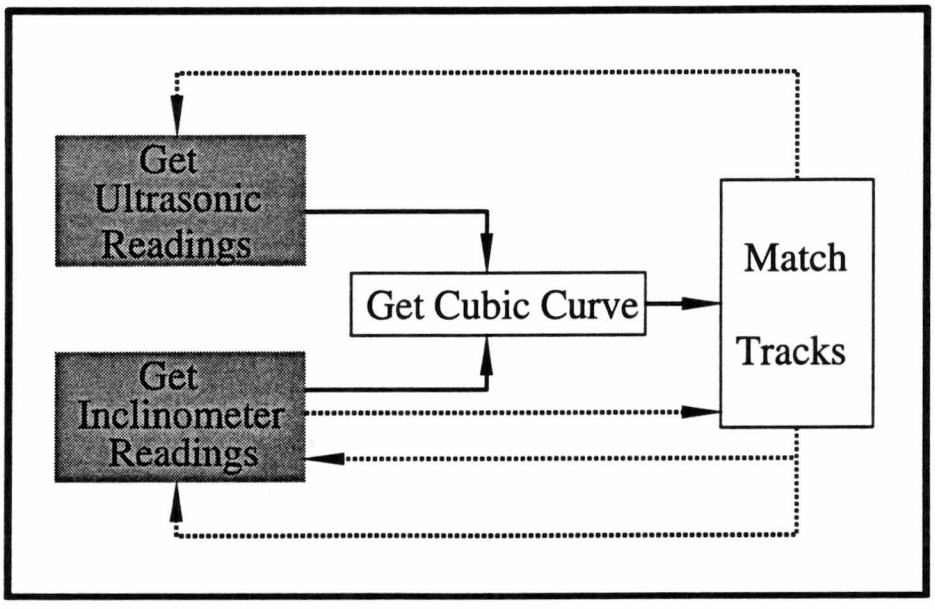


Figure 1.5: *The robot R2M2.*

readings. A cubic curve was fit through the path, and the tracks were matched at their joints with inclinometer feedback. An outline of that algorithm [8] is illustrated in Figure 1.5.

1.4 Summary

Insufficient data can be interpreted through interpolation techniques. Robot navigation is one issue where data interpolation can be utilized to solve some of the problems encountered by a mobile robot. Automatic navigation of a robot in unknown environments requires integrating sensors with the robotic system to collect

information about the environment, transforming the collected information to knowledge about the environment, making decisions based on the knowledge, and finally, using sensory data feedback to upgrade the decisions. Interpolated data can be applied in order to make reactive decisions for a mobile robot, using only localized information about the environment and feedback of sensory data.

The problems involving surface interpolation and automatic navigation of mobile robots are attempted to solve in this thesis through developing surface interpolation algorithms and by using polynomials to guide a mobile robot. Graphical user interfaces for a mobile robot and for the surface interpolation techniques are also built. Therefore, the accomplishments of this thesis can be narrated by the following items:

- Development of surface interpolation algorithms using polynomials, with rigorous statistical analysis,
- Evaluation and implementation of polynomial based reactive obstacle negotiation of a mobile robot, and
- Development of graphical user interfaces for teleoperation of the mobile robot R2M2 and for surface analysis and viewing in three dimensions.

It is also emphasized that the surface interpolation techniques developed can be utilized for a variety of range data analyses such as environment mapping, recognition, and surface characterization. The versatility of the algorithms developed is, therefore, imoprtant and valuable.

CHAPTER 2

The Ranging System

As emphasized thus far in this thesis, the surface generation technique is based on range data. The distances are measured from the plane containing the sensor array to the various points on the surface being studied [9]. Ultrasonic range finders were used [10] [11] [12] in order to obtain the range information for the experimental results of surface interpolation reported in this thesis. Polaroid ultrasonic ranging buttons were integrated with a TRC multiplexing board which consisted of a Motorolla 6811 microprocessor and the ultrasonic transmit-receive circuitry.

2.1 The Ultrasonic Range Sensors

The range finders work based on the Time Of Flight(TOF) information of the ultrasonic sound waves. That is, a range button transmits a pulse of ultrasonic sound and then waits for the sound to come back. The sound wave bounces off of the object in its path, if there is an object in the path, and comes back to the sensor. What the sensor realizes is the time difference, or the *Time Of Flight*, between the instance it transmitted the sound, and the instance it received the sound. The velocity of the sound is a constant, and therefore, the time difference between the transmit instance and the receive instance is proportional to the distance of the object the

sound reflected off from.

A gold plated thin plastic foil is the part of each transducer which converts electrical energy to sound energy. An AC voltage of a certain known frequency is applied in order to force the foil vibrate at that particular frequency and transmit sound at that frequency. Coherent phase conditions are used to trim the beam of the transmit and receive sound waves. Also, the phase pattern is well defined so that the responding sound wave's frequency-domain shape (or the "lobe" pattern) is known.

One important aspect in ultrasonic ranging is the signal strength in the echo. The echo depends on the shape of the object and its surface geometry. Different portions of the object reflect sound in different strengths. That means, the sound absorption quality of the object is important in determining the distance with accuracy. In addition, the signal strength depends on the frequency of the sound. The reflected sound from a particular object at a certain frequency might be weak, but the same object can reflect a strong signal if sound were transmitted at a different frequency. The setup for the ranging system used had a transmission frequency of 50 KHz.

2.2 Principle Behind Ultrasonic Sound Based Ranging: The "Chirping" Concept

The ultrasonic sound range sensors operate on a popularly known concept called the "Chirping" concept to measure distances from the sensor to an object. Animals such as bats utilize this concept with great efficiency. An ultrasonic sound pulse, or a "chirp", is transmitted from a sensor, then the sensor waits for the sound to come

back. The time in between is, as mentioned earlier, called the *Time Of Flight(TOF).* Let this time of flight be denoted as Δt. Now, the velocity of the sound is a known quantity, and let this be denoted as v. According to Figure 2.1, the distance traveled by the sound before the echo is heard is roughly (but not exactly) twice the distance of the object from the sensor. If this distance is labeled as d, then we have the following relationship according to Newton's equation of motion:

$$\begin{aligned} 2d &\cong v.\Delta t \\ d &\cong \frac{v.\Delta t}{2} \end{aligned} \tag{2.1}$$

And Equation 2.1 is the one that translates the TOF information into range data.

But, as also viewed in Figure 2.1, the actual distance traveled by the ultrasonic sound is not exactly equal to the ideal twice the distance between the object and the sensor. Rather, in almost all the cases, the distance traveled would be a little different from the actual desired distance since the travel of the sound wave depends on the incident and the reflective angles on the object, the object shape, and any other sort of interference between the sensor and the object. In Figure 2.1, the distance that the sound travels is actually larger than the desired distance.

In addition to the error added by the incident and reflective angles, there are additional problems caused by the multiple reflection of the wave from the same object. The ultrasonic sound from the sensor is transmitted along a cone with an expanding angle of about 30 degrees. Therefore, there could be multiple waves reflecting off the same object. And as a result, the echo heard by the sensor is the first wave reflected off of the object, and not necessarily the sound wave that would determine the correct

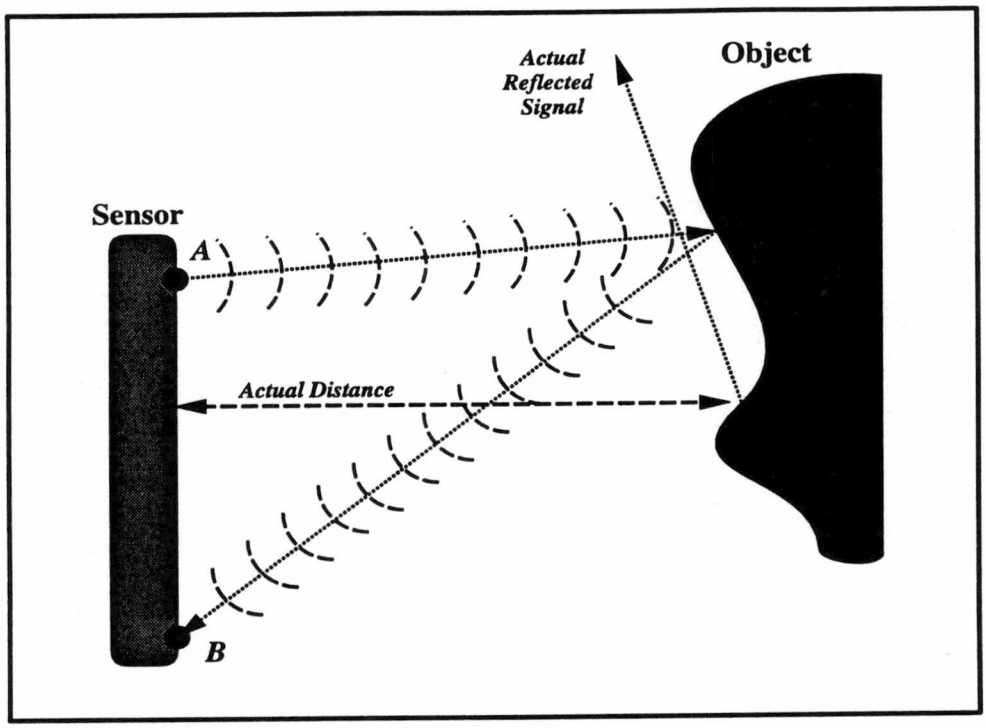


Figure 2.1: *The "chirping" concept. The sound wave travels from point A of the sensor, reflects off of the object in the way, and finally returns to point B of the sensor. Note that the distance traveled by the sound wave is not exactly equal to twice the distance of the object from the sensor.*

distance. Also, in Figure 2.1, the wave that could measure the actual distance is shown to be reflected away and never received by the sensor.

There are some instances when the obtained data can be significantly erroneous. For instances, ultrasonic sound based ranging perform poorly when the sensors are looking at simple corners of a room. The incident waves may reflect off one wall, then bounce off the other wall, and finally come back to the sensor. This way, the distance measured would be significantly greater than the actual distance of the object, in this case the two walls meeting to create the corner, from the sensor. Figure 2.2 illustrates this fact. The sound bounces from wall A first, then gets trapped in the reflection from wall B, and then finally, comes back to the sensor. The sensory module, however, is ignorant about the multiple reflection, assumes only one reflection, and measures the distance according to Equation 2.1. This is true for concave corners.

The convex corner of a wall can also confuse the ultrasonic range sensors. This incident is portrayed in Figure 2.3. The sound transmitted could easily be reflected by a convex corner at such an angle away from the sensor that the sensor might never hear the echo. Therefore, the distance measured in such a case is interpreted by the electronic circuit as some distance beyond the maximum or time-out distance permitted by the circuit. In the actuality, however, the distance was only a corner, perfectly in the range of the sensor.

Nevertheless, despite all the difficulties imposed by practical situations, the ultrasonic sound based ranging has proven to be a reliable source of range information within distances between six inches and 35 feet. The data, unquestionably, is noisy, but not noisy enough such that range analysis cannot be performed from this data.

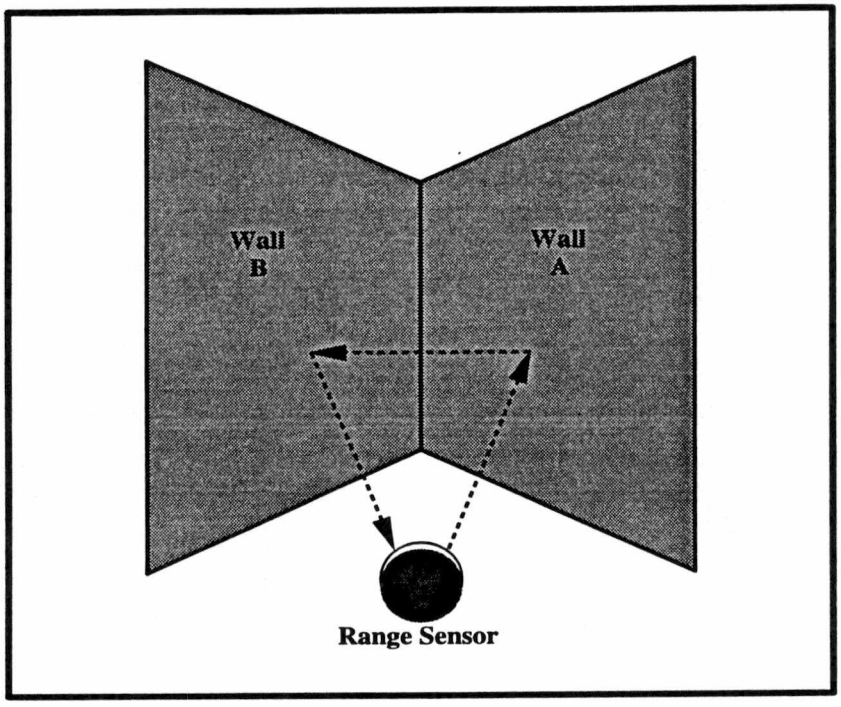


Figure 2.2: *The multiple reflection of the ultrasonic sound as it reflects off two surfaces before the sensor hears the echo.*

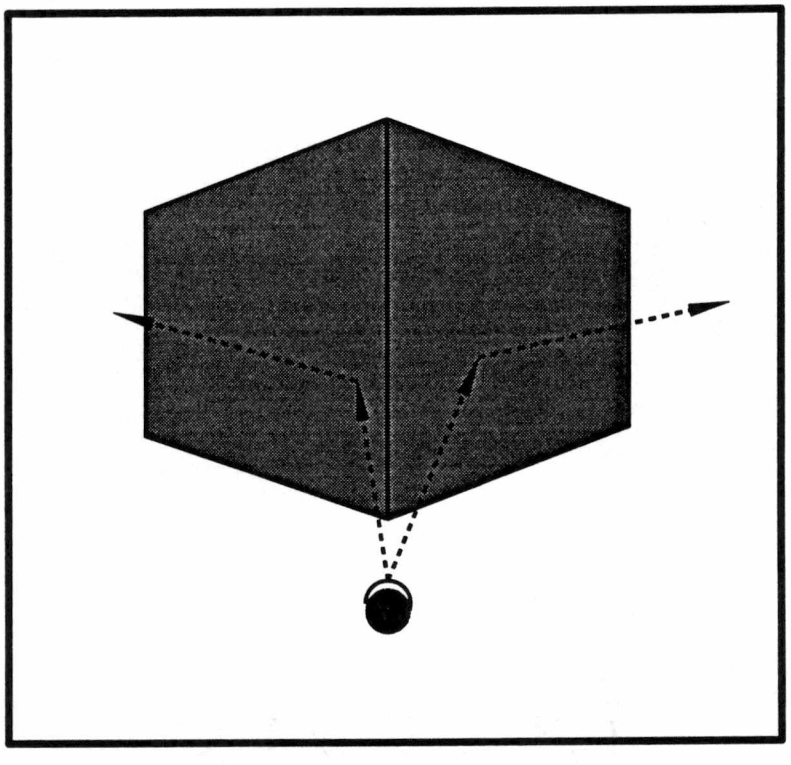


Figure 2.3: *Range sensing being mislead by a convex corner adjoining two walls.*

2.3 Description of the Sonar Ranging Module

The Texas Instrument ranging module used was the SN28827 chip [13]. Without any additional interface, this module is capable of driving a 50-KHz, 300-Volt transducer. With an additional simple interface, this module can measure distances ranging from 6 inches to 35 feet. This module is equipped for multiple echo detection through external blanking inputs. The noise reduction and the side lobe detection in the signals are furnished through the digitally controlled gain and the variable bandwidth amplifier. This chip is designed to perform between temperatures of 0°C to 40°C with the sonar transmit output of 16 cycles at 49.4 KHz.

In Figure 2.4, the component layout of the SN28827 module for the sonar transmit-receive circuitry is shown [13]. The power(VCC) is applied first. Then a 5 milliseconds waiting period must elapse in order for all the internal circuits to reset. After reset, the initiate(INIT) signal is brought high, and 16 pulses at 49.4 KHz with an amplitude of 300 V excite the transducer to cause transmission of ultrasonic sound waves. The dc bias voltage is kept at 150 Volts even at the end of transmission for optimum operation according to the manufacturer. After the transducer is fired, the receive(REC) signal is kept inhibited for a certain period of time by an internal blanking signal. This is done so that the continuing vibration or the ringing of the transducer after transmission is not detected as a received signal. The time of inhibition can be ended by taking the BINH signal high, and this inhibition time is varied depending on the minimum distances that the transducer would be required to measure.

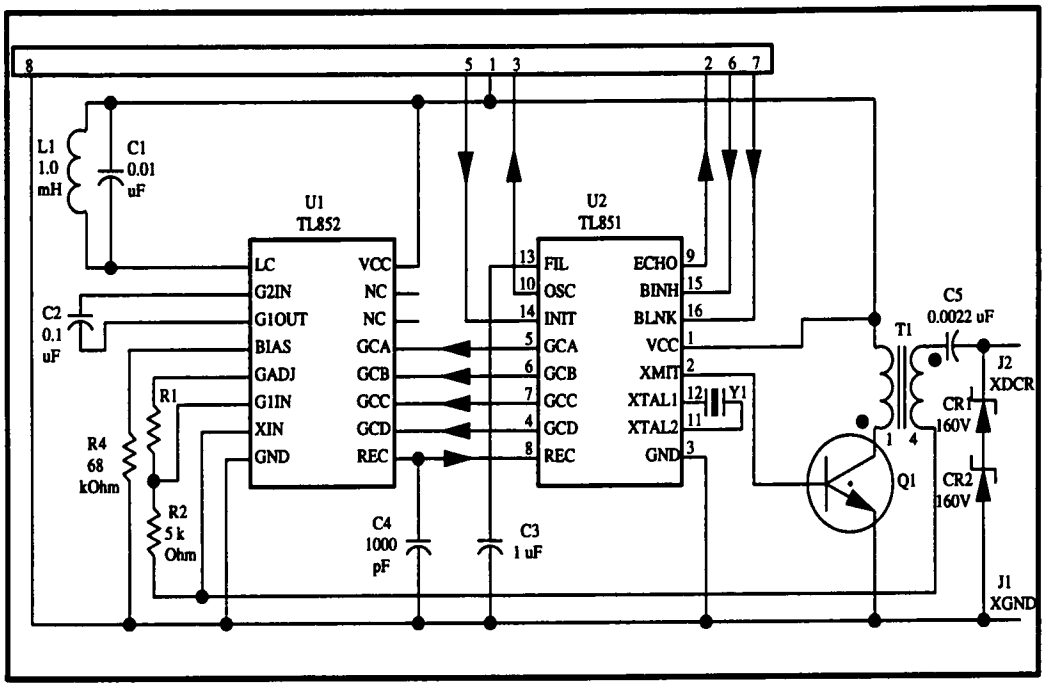


Figure 2.4: *Component layout of the SN28827 module for the transducers.*

The utrasonic sound wave travels one foot and back in about 0.9 milliseconds. The returning signal is referred to as the echo output signal. The echo signal, upon receiving, is amplified in order to make it appear as a high-logic-level echo output. The time difference between the INIT going high and the echo(in the diagram, ECHO) going high is proportional to the distance of the object in the path of the sound wave transmitted by the transducer.

Another important aspect is the receiver amplifier gain. This issue is important because of the variation in the signal strength in the sound waves. Transmitted signals get weak as they travel further and further. As a result, the received signal is also weak when an object is detected at a greater distance. The receiver amplifier is, therefore, clocked to have higher gain as more and more time elapses before the

received signal is detected. The higher gain compensates for the attenuation in the signal. For this particular module, the maximum gain is achieved at about 38 milliseconds. However, even with this approach, especially for objects 35 feet or further away(which corresponds to 38 ms), sufficient gain might not be achieved. In that case, additional gain could be obtained by varying R1 in Figure 2.4. But there is a limitation to this variation in order for reliable operation of the module to be achieved.

2.4 Integrated System

The previously discussed ranging module was integrated with the transducers and the host computing system for experimentation reasons. Basically, there are four modules that make up the system for interpolating surfaces. They are as follows:

- The host computer
- The remote host
- The ranging module(SN28827 and Motorolla 6811)
- The transducers

The host computer is the place where the surface generating software exists. The host computer has all the necessary data available through the other three modules and generates the continuous surface function. In this project, the host computer was a Silicon Graphics workstation available in the laboratory. The remote host, however, is the key element between the computation at the host end and the data collection at the transducer end. This remote host responds through Remote Procedure

Calls(RPC) from the host, and communicates with the ranging module through a wireless serial connection. The ranging module is controlled by this remote host, and it reports data to the main host upon RPC request. This method of operation keeps the sensors alive and activated at all times while the host gets the input data from the remote host whenever needed. The VxWorks processor "penguin" was used as the remote host because this processor is available in the laboratory and is extremely prompt for RPC. The standard serial communication between the remote host and the ranging module is done at 9600 baud with 8 bit data packets having no parity and one stop bit. Two wireless modems were used to furnish this wireless communication. The ranging module, however, only controls the transducers by firing them in a disciplined fashion. The transmit-receive operations are done through the TRC board while the Motorolla 6811 multiplexes among all the sensors in determining which sensors are to be fired, read, and reported. And finally, the transducers themselves transmit ultrasonic sound upon proper excitation and receive the reflected signals, if any.

For clarification, a step-by-step outline of how the ranging system is operated is described here. Figure 2.5 is an outline of how the different elements of the integrated sensory system are connected. The dotted connection is the wireless connection, and the solid connections are the wired connection. The RPC client VxWorks runs the program that controls the sensory module in parallel with the surface interpolation program. Therefore, step 1 for the integrated system is to spawn the RPC client software and to run the host program. Then, in step 2, the host asks for new data set. The remote procedure call is made at this stage, and the VxWorks fires off the

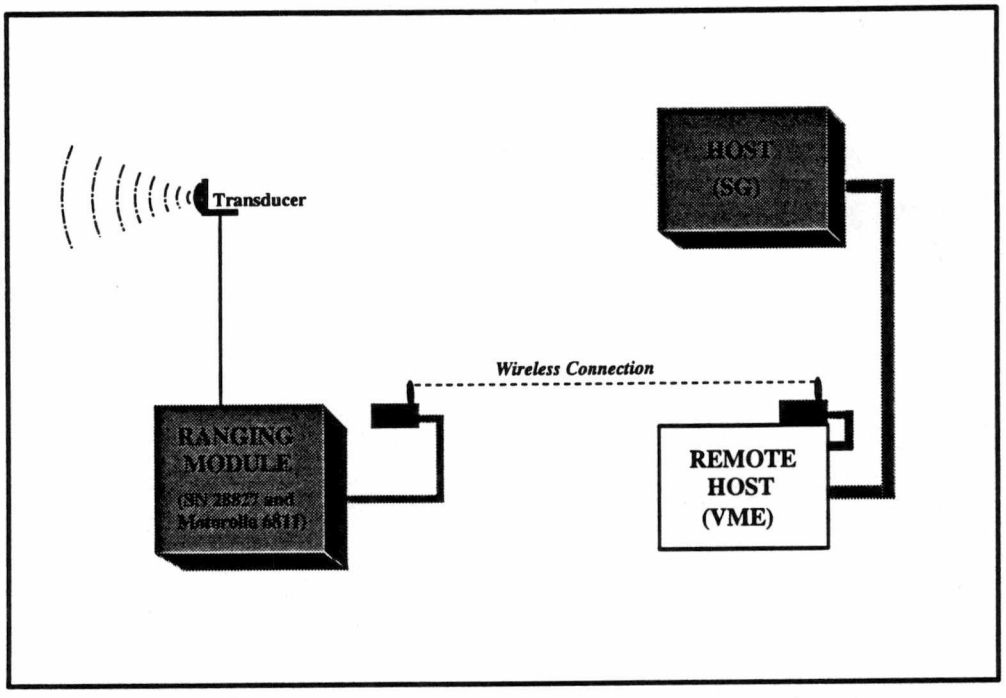


Figure 2.5: *Integrated hardware system for surface generation.*

sensors through the wireless connection. The received data is fed back to the RPC client. That data, as step 3, is available to the host by return of the remote procedure which was just called.

2.5 Summary

Ultrasonic sound based range sensors are selected to be mounted on the robotic subsystem R2M2. The Polaroid sonar ranging buttons are integrated through a TRC transmit-receive circuit using a SN28827 chip and a Motorolla 6811 microprocessor along with the other necessary circuit elements. The sensory subsystem is connected to the host computers on the robotic subsystem end with a wireless serial link. There-

fore, R2M2 is upgraded with an integrated sensory subsystem and also with sophisticated computing capabilities.

CHAPTER 3

Issues in Sensor Positioning

Since this study of three dimensional surfaces is based on carefully utilizing the two dimensional interpolation techniques over a surface of interest, organized sensor placement is important for calculating accurate coordinate positions of various points on the surface [14]. It is desirable that depth information remains along straight lines on the plane in the two dimensions containing the sensors. Each run of the two dimensional approximation algorithm thus provides an interpolation of the surface along a line on the surface. Multiple runs of the two dimensional techniques in different directions set up a basis for interpolating the entire surface scanned by the range sensors. Therefore, the sensor placements should be such that straight lines could geometrically be obtained in different directions on the sensor-plane. This prerequisite of sensor organization would be referred to as Straight-line Sensor Placement(SSP). The more different directions for two dimensional interpolation are created, the better approximation of the entire surface is possible. Thus multi directionality of the SSP is desired in this interpolation method.

Several different approaches to the sensor positioning were pursued. There are certain criteria that need to be satisfied for any particular type of sensor placement worth considering. The straight line sensor placement (SSP) and the repeatability of the placement geometry are essential to verify, and, therefore, it is very difficult to be

innovative in this respect. There are two different basic repeatable geometric structures considered because both of those structures have straight edges and are capable of sub-dividing into their original but smaller shapes. These geometric structures were squares(or rectangles) and hexagons.

The first among them is a square plane containing the sensors, named Square Array Organization(SAO). Thus sensor readings could be interpolated as pixels of an image and any image processing techniques could be applied in interpreting the missing information. Among other approaches, the Hexagonal Organization of Sensory Template(HOST) is helpful as multiple straight lines are achieved around a certain sensor on the plane and the structure is fairly repeatable.

3.1 Square Array Organization(SAO)

In the Square Array Organization of sensors, as shown in Figure 3.1, the sensors are placed in rows and columns. The number of rows and columns are kept the same, thus providing a square geometric shape. The basic structure of the SAO is shown in Figure 3.1 (a).

There are a few characteristics of this Square Array Organization. As seen in Figure 3.1, the rows and columns definitely create the straight lines that are required to fit the 2-D polynomials in order to perform this 3-D surface interpolation technique. These rows and columns provide lines at 0°,90°, and 180° of angles. The diagonals, which also lie along straight lines, provide the directions of 45 and 135 degrees. Thus multi directionality of the Straight-line Sensor Placement(SSP) is achieved.

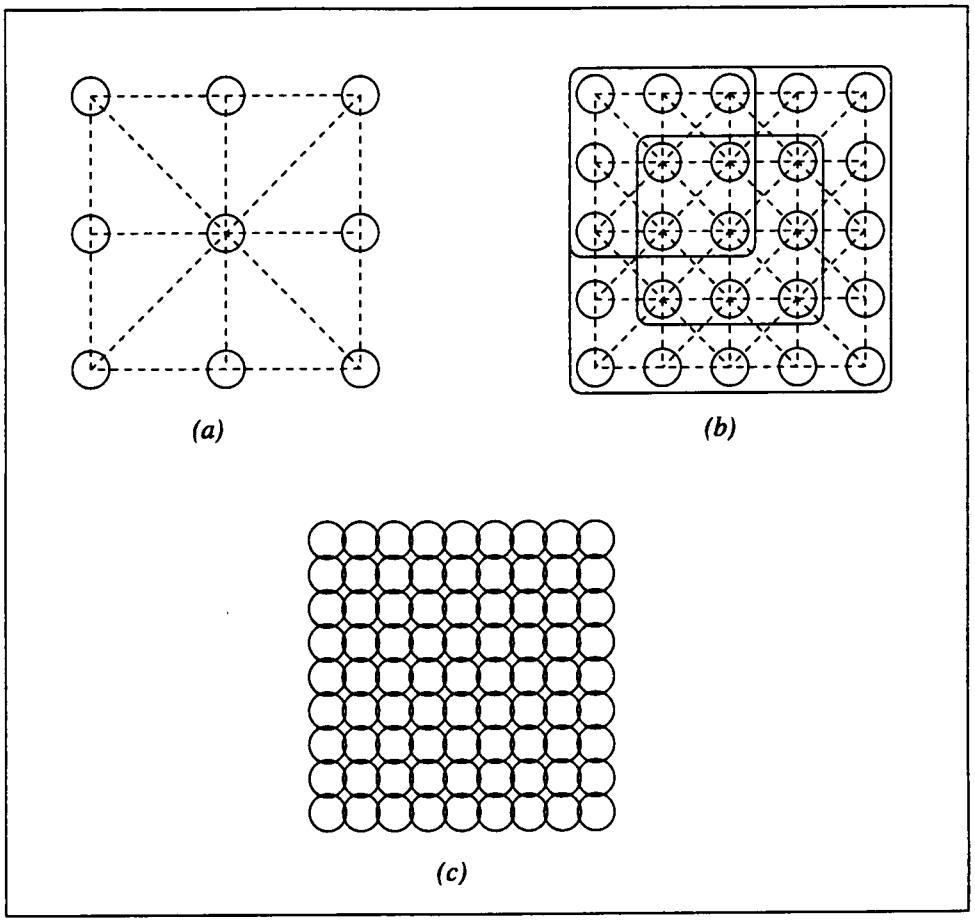


Figure 3.1: *The Square Array Organization(SAO) of the sensors. (a) The basic organization in a 3 by 3 array, (b) the masking after upsampling the data, and (c) the repeatability of the SAO.*

Another attribute of this SAO structure is its reproductness. Once the first set of algorithms are run to interpolate points in between the actual readings of the sensors, the basic square structure with equal numbers of rows and columns is created once again, only with more rows and columns the the new version(Figure 3.1 (b) and (c)). In addition, mini structures of the same arrangements are created within the global primary structure. This is shown in Figure 3.1 (b). Thus SAO behaves as a reproductive structure within the scope of this 3-D surface interpolation technique.

An inherent advantage of the Square Array Organization is the error reduction due to the repeated use of the same sensor reading. Even in the primary SAO structure, some of the same sensory readings are used to interpolate along rows and columns, as seen in Figure 3.1 (a). And in the later stages of the reproduction level, the mini SAO structures are contained within the global structure, causing more than one use of the same data point. Also, the mini structures themselves overlap on one another, causing more reruns of the scanned points. Thus error reduction statistically takes place by averaging multiple interpolated values at the same coordinate on the surface under consideration.

3.2 Hexagonal Organization of Sensory Template(HOST)

The Hexagonal Organization of Sensory Template(HOST) is another type of sensor placement that can be considered. The HOST meets the two crucial requirements of this project—the placement of sensors along straight lines and the repeatability of the basic structure.

The HOST has the structure of a simple hexagon, exactly like a benzene ring structure that Organic Chemists are familiar with. Figure 3.2(a) shows this basic hexagonal structure. It could be noted that sensors can be placed at every corners, or junction of two lines, and they could be placed anywhere in between along a straight line. Also, as shown in Figure 3.2(b), the structure is fairly repeatable. The reason the term "fairly" is used is because this structure is not entirely repeatable like the SAO structure described earlier. There are six parallelograms that are created in the reproduction or subdivision of this structure. These parallelograms have to be taken care of by some other method than the repeatable one desired for this project. Each one of the seven hexagons reproduced in the division of the original structure could also be subdivided into seven hexagons and six parallelograms. If the need for looking into the parallelograms is not important, this is a good sensor placement technique to use.

3.3 Choosing One Sensor Array Type: The SAO Structure

Even though different repeatable structures were studied for as potential candidates for the study of this research, only one sensor organization was practical to pick since the theory developed in this project would ultimately be implemented on hardware. It is difficult to build multiple hardware platforms for testing an underdeveloped technique because nobody knows whether the theory would actually be useful or not before it is developed. It is also good practice to study the options carefully, and finally pick the best seemingly best option for practical implementation.

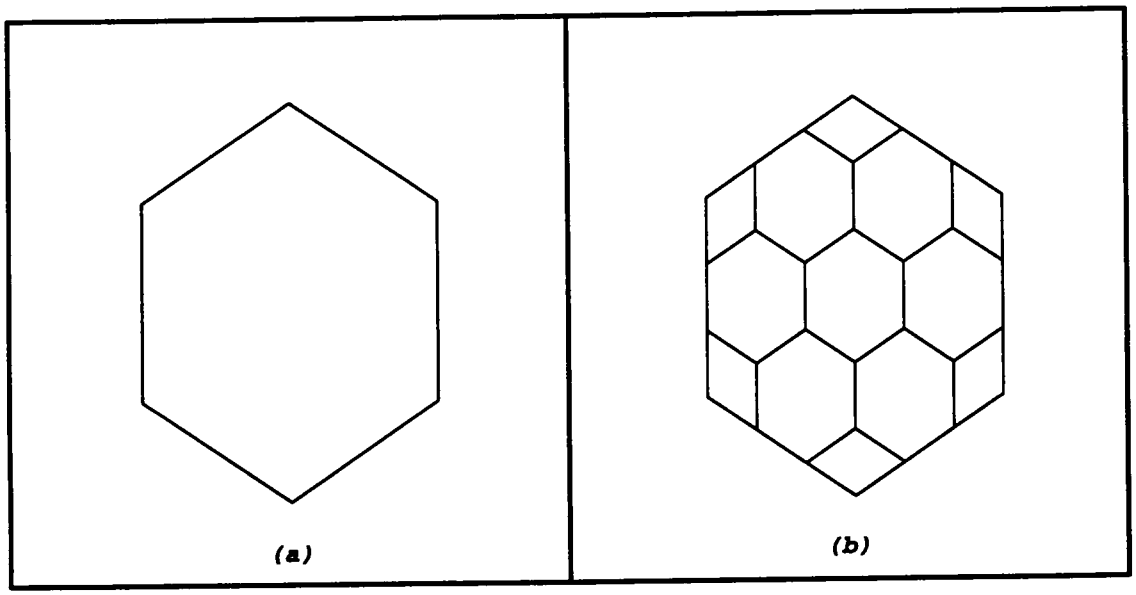


Figure 3.2: *The Hexagonal Organization of Sensory Template(HOST). Figure (a) shows the basic structure, and figure (b) illustrates the repeatability of the basic structure.*

The Sensor array which was picked for this thesis was the Square Array Organization(SAO). The reasons behind this choice are listed below:

In the SAO structure

- *rows and columns are easier to identify than the HOST,*
- *there are more repeatability in the subdivision whereas the HOST has areas where parallelograms are created instead of hexagons,*
- *hardware implementation requires simpler assembly than the HOST, and*
- *it is simple to look at local regions, by conventional masking methods of isolating square regions.*

It is not, by any means, implied that the HOST would be impossible to implement. It is rather emphasized that both from the hardware and software perspectives, it is easier to implement the SAO placement type. That is why the SAO was chosen instead of the HOST for sensor placement.

3.4 Summary

Since data interpolation through polynomials is the heart and soul of the surface approximation algorithm developed in this project, appropriate sensor placement played an important role in the effectiveness of the algorithm. A square sensor placemet technique named SAO(Square Array Organization) is chosen to meet the constraints that would be posed by the polynomials and masking requirements of the algorithm developed.

CHAPTER 4

Curve Interpolation

Data interpolation is the discipline where information is processed based on a table of data or functions [15]. Interpolation techniques attempt to identify the correspondence between the discrete information available, and based on these correspondences, information at unknown or unavailable areas are approximated.

Many times, the data available from a system remains insufficient for it to arrive at conclusive results with reasonable accuracy. When this available data is not enough to process through the system, the data needs to be interpolated [16] at places where information is not available. The system, if the interpolation is in coherence with the real data, is often able to process this extended information to output acceptable results. This is why, from an engineer's point of view, data interpolation is important.

Data interpolation could be achieved from two different approaches. The first one among them is upsampling the data at specific locations of interest. The other approach provides a functional description of the data where interpolating data can be achieved at any location desired within the boundary of the functional description.

For applications of interpolation techniques in approximating surfaces, most people tend to take the second approach defined in this thesis—an approach that deals with a general functional description of the entire surface within a certain boundary. Dierckx takes an approach to surface fitting by evaluating a proper set of coefficients to

fit a spline function which would describe the surface. Hall [17], [18], however, takes a slightly different approach to approximate surfaces by splitting the surface under consideration into sub-intervals or sub-surfaces. He calls this procedure a CNC(Computer Numerically Controlled) approach. As the basis of surface fitting, two dimensional interpolations also become crucial in the context. Loach talks about the continuity of spline functions and their improvement in convergence resulting from a certain set of free "knots" or data points in the polynomial. Mulansky, in accordance with Loach, discusses the better performance of the Chebyshev polynomials having free knots.

4.1 3-D Surface fitting

In the Department of Computer Science at KU Leuven, Belgium, Dierckx, Leemput, and Vermeire [1] fitted a surface through a set of scattered points in 3 dimensions. Splines with proper triangulation were used for this surface fitting job. The depth map, referred to as $Z(x, y)$, is first obtained and then smoothed in order to derive the final description of the surface. The smoothing factor was the weighted sum of the squared residuals. The triangulation method was carried out through an adaptive way where all the knots of the spline were determined.

Dierckx [1] decided to break the surface up in triangles, and the surfaces for all the different triangles were combined to obtain the surface description. Bernstein-Bézier representation of the polynomials were used in this study [2]. The polynomials were Bernstein polynomials and the coefficients were the Bézier ordinates.

For better understanding of Hall's approach, the definition of the Bernstein Poly-

nomial and their Bézier coefficients are listed here. The Bernstein Polynomial of degree d is described as:

$$B^d_{i,j,k}(\tau) = \frac{d!}{i!j!k!}\tau_1^i\tau_2^j\tau_3^k \tag{4.1}$$

Here, the τ symbols have the following definition,

$$\begin{aligned} \tau_1 x_1 + \tau_2 x_2 + \tau_3 x_3 &= x \\ \tau_1 y_1 + \tau_2 y_2 + \tau_3 y_3 &= y \\ \tau_1 + \tau_2 + \tau_3 &= 1 \end{aligned} \tag{4.2}$$

If the Bézier coefficients with respect to the proper triangle is $b_{i,j,k}$, then the Bernstein-Bézier polynomial is expressed as:

$$p(x,y) = \sum_{i+j+k=d} b_{i,j,k}B^d_{i,j,k}(\tau) \tag{4.3}$$

This was the basis for Dierckx's work. He applied this adaptive triangulation method using the Powell-Sabin splines. The Powell-Sabin splines are described in the following manner:

$$S^\tau_d(\Delta) = s(x,y) \mid s\epsilon C^\tau(\Omega), s_{|\rho_j}\epsilon\prod_d, j = 1,...,T \tag{4.4}$$

Here, the given conforming triangulation Δ of Ω with triangles ρ_j are described.

Without going into further detail of this particular approach, it can be understood that a lot of updating in the complex triangulation process is needed to be done before each of the polynomials in a particular triangle can be determined. Also, the evaluation of the polynomials themselves involve extensive calculations for their

coefficients, the expressions of which coefficients are not listed here to avoid the perplexity each one imposes. Therefore, this approach could be considered as one which is computationally demanding. For applications such as robot navigation and obstacle detection, the computational burden is a concern since a lot of other computation involving the robot kinematics, dynamics, local and global positioning is achieved in parallel with this sensory aspect of the robot. Therefore, this type of approach for designing the sensors on a robot can be considered as complex.

Nevertheless, Dierckx's method has one aesthetic aspect to it which is highly commendable. The polynomials determined are carefully trimmed at places to remove sharp edges and abrupt changes in the contour. The residual smoothing provides the surface with an artistic characteristic. Also, with proper triangulation in the surface domain, the accuracy of the interpolations for gaussian distributed surfaces could be reduced to about 5%. This is a significant strength of this interpolation technique.

3-D surface approximation also has practical applications such as material removal from optical surfaces [18]. This work was inspired by the need of polishing and cleaning up lenses in machines.

> *A flawed mirror blurs the vision of the $1.5 billion Hubble telescope, imperiling its hopes of seeing the edge of space and the beginning of time.(Newsweek, July 1990)*

A computer Numerically Controlled(CNC) study has supplanted, to a certain extent, the human efforts for grinding and polishing such surfaces [17]. Mathematical models were developed for the kinematics of the operation tools and also for the material

removal profile for the surface under consideration. One of the key parameters in the CNC approach is the material removal rate of the machine. And the determination of this material removal rate requires the workspace profile along with the speeds of the points on the contact pad in a certain time interval. Even though this study was conducted more towards solving the problem of material removal rate from the optical surfaces, the rate depends on parameters such as the pressure between the pad and the workpiece and the relative speed between the pad and the workpiece. The working surface was assumed to be stationary, but the surface contour was still needed to be estimated to some extent. The material removal rate was determined based on a parameter called the material removal profile space, a description which was clearly influenced by the contour of the surface. The contour information of the surface, in this case, was interpreted through the pressure data between the removal pad and the surface itself.

3-D shape reconstruction is another area in which surface information becomes crucial. Surfaces from multiple range images is one method of solving this structure reconstruction and representation problems [19] [20]. Instead of extracting and matching features and thus constructing the 3-D shape, all the surfaces of an object from the multiple range images were constructed, and then they were finally fused to determine the structure of the object. Octree-splines were used in order to find the range data from the sensor to the surfaces, and the minimization of the least squared technique helped reduce the error introduced by the splines. Range data analysis also include developing operators for computer vision [21].

The fast distance computation [19] was achieved in four distinguished steps, much

similar to the approach taken in this thesis. Surface point octree-splines were constructed first [22], followed by the subdivision selection on the surface, corner distance computation [3], and finally the crack elimination or the continuation enforcement. In this method, each surface was first constructed and later smoothed to reduce errors and avoid sharp edges.

One of the central ideas behind any interpolation procedure is that the interpolation model derived comes from a statistical basis. That is, the available information is indeed statistical information about the signal or the function. Therefore, some mathematicians and scientists often take statistical approaches for interpreting information. Tanemura [23] uses a directional interaction approach in order to statistically model point objects in space. This may not have direct implications on 3-D structures since he deals with point objects, but nevertheless it is another method of interpreting statistical information to come up with solution models for given problems.

4.2 2-D Polynomial Interpolation

2-D polynomial interpolations are often the basis of the 3-D functions describing surfaces. At the least, the basic principles for interpolation are much easier to understand in two dimension than in three dimensions. The purpose of this thesis project was to expand some chosen 2-D algorithms into 3 dimensions in order to obtain surface approximation. And since this expansion required modification and additional complexity in the context of developing the new algorithms, relatively computationally easy algorithms received more attention than their more complex counterparts. The

basic algorithms looked into were the Lagrangian interpolation, the Chebyshev polynomial, the Linear interpolation, the Cubic Hermit and Cubic Spline interpolations, etc. These methods are briefly discussed here in terms of their prior applications.

Mulansky [24] describes a method of using Chebyshev polynomials in the spline fashion. He decided to take the spline approach so that he could free up the knots guiding the polynomial through the plane. Spline functions, as it is common knowledge to the people involved with data interpolation, are very flexible and fit even the nonsmooth functions efficiently. Mulansky followed the notion of tangent cone and came up with a new sign rule for the spline theory. In his opinion, free knots give more flexibility for the splines than fixed knots do. This comment is supported by this thesis, where a fixed knot at the boundary was impossible to achieve and free boundaries were considered.

The construction constraints of cubic spline polynomials are described by Murchuk [25]. the constraints require that the piecewise cubic polynomials have continuous derivatives up to the second order. the conditions also require the polynomials to go through the discrete given points and their second derivatives at the boundaries be zero. This is the type of cubic spline polynomial that is generally used for interpolation. The spline technique used in this thesis, as described in Chapter 5, have conditions very similar to the ones described by Murchuk.

2-D polynomial derivation techniques are extensively discussed by Bearden [26]. The Lagrange polynomial, according to Bearden, is oscillatory with fewer data points and more stable with larger quantity of available points. Even in a stable polynomial, however, Lagrange polynomials are oscillatory at their endpoints. The cubic spline

polynomial described by Bearden has numerous conditions associated with it. Bearden's analysis in this respect is quite similar to that of Mulansky [24], as mentioned earlier. Bearden also seems to prefer the natural boundary conditions for general purpose implementation of cubic spline algorithms.

CHAPTER 5

Interpolation Algorithms Basics

Approximations at different points on the surfaces were done through several mathematical interpolation techniques [15] [26] [25]. The mathematics is based on two dimensional data, which means that the techniques deal with curves on a plane or polynomials [27]. The algorithms used are: Lagrange Algorithm, Piecewise Linear Algorithm, and Cubic Spline Algorithm. The reason more than one algorithm were studied was because a comparative study could be done about the performances of these algorithms on a three dimensional surface using the 2D to 3D expansion of the interpolation techniques. Brief discussion of the algorithms used for surface approximation are provided in this chapter.

5.1 The Lagrange Algorithm

The Lagrange Interpolation method is based on point data on a curve and the derivative information at those points. This sort of interpolation tends to be oscillatory at the endpoints and more stable in the middle of the interpolation coordinates. The reason is very simple. It is hard to find derivatives towards the endpoints of the interpolation coordinates. In fact, for discrete data set, the derivatives at the endpoints are not even defined in most cases. Therefore, some other approximation

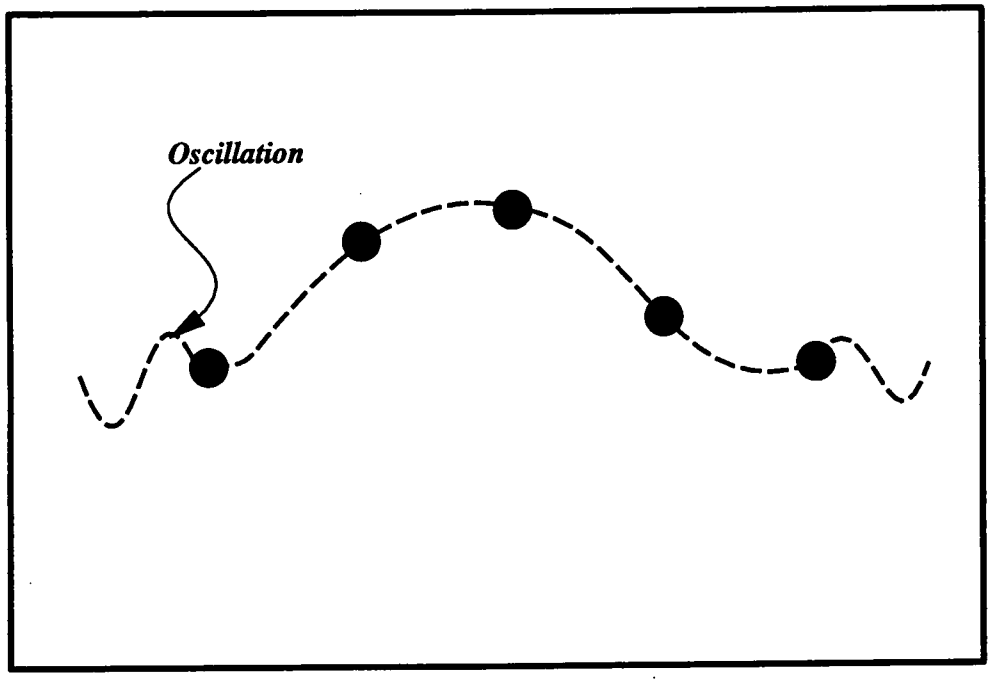


Figure 5.1: *The Lagrange interpolating polynomial. Note the oscillation at the end-points. The dark dots are measured data points.*

method would be needed at the endpoints in order to find the derivative information. This approximation introduces additional errors at the starting point of the algorithm, and therefore, the curves at the endpoints tend to deviate further from the true measured values.

The algorithm generates a polynomial which goes *through* the available data points, and then tries to *approximate* along the rest of the polynomial's range(Figure 5.1). There are some shortened term- definitions that would be made during the course of describing this algorithm.

Lagrange Polynomial is defined to be $P(x)$ in this description. If the actual mea-

sured value at any position x_k is $f(x_k)$, then the goal of the polynomial is to achieve $P(x_k) = f(x_k)$. For the application of this project, which is Surface Interpolation, the major objective was to minimize the difference of $P(x_k)$ and $f(x_k)$. Therefore, the accuracy of the surface interpolation technique depends on the following term:

$$P(x_k) - f(x_k)$$

For a set of measured data points $(x_0, f(x_0)), (x_1, f(x_1)), ..., (x_n, f(x_n))$, the Lagrange Polynomial is defined as,

$$\begin{aligned} P(x) &= f(x_0)L_{n,0}(x) + f(x_1)L_{n,1}(x) + ... f(x_n)L_{n,n}(x) \\ Or,\ P(x) &= \sum_{k=0}^{n} f(x_k)L_{n,k}(x) \end{aligned}$$

Here, $L_{n.k}(x)$ is defined by the following,

$$\begin{aligned} L_{n,k}(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)...(x-x_{k-1})(x-x_{k+1})...(x-x_n)}{(x_k-x_0)(x_k-x_1)(x_k-x_2)...(x_k-x_{k-1})(x_k-x_{k+1})...(x_k-x_n)} \\ &= \prod_{i=0, i\neq k}^{n} \frac{(x-x_i)}{(x_k-x_i)} \end{aligned}$$

Here, The $\prod$ symbol represents the usual product of the terms assigned to the $\prod$ symbol.

It is pretty evident that the degree of the polynomial is the highest degree in the term $L_{n,n}(x)$. And from the definition of $L_{n,k}(x)$, there are n product terms associated with $L_{n,n}(x)$ for $n+1$ points. And, therefore, the *degree* of the Lagrange Polynomial derived by $n+1$ data points would be n.

5.1.1 Lagrange Algorithm Computation Technique

The Lagrange Interpolation uses a computation method called Newton's Divided Difference(NDD) method. This NDD is basically a quotient of two difference measures. For example, the slope of a straight line is a divided difference. That is, if a straight line goes through points (x_1, y_1) and (x_2, y_2), the slope is defined by

$$slope \;=\; (y_2 - y_1)/(x_2 - x_1)$$

which is a divided difference of the sets (x_1, y_1) and (x_2, y_2).

It could be assumed that $P_n(x)$ is an nth degree Lagrange Polynomial that approximates some pre-defined function $f(x)$. The NDD method enables one to derive coefficients a_0 through a_n if the polynomial is expressed as follows,

$$P_n(x) \;=\; a_0 + a_1(x - x_0) + ... + a_n(x - x_0)(x - x_1)...(x - x_{n-1}) \quad (5.1)$$

The computation of the coefficients starts with a sorted data set. That is, the independent coordinates of the data sets should be sorted in an ascending manner in order to start the procedure of deriving the polynomial. For simplicity purposes, it could be assumed that $(x_0, f(x_0))$, $(x_1, f(x_1))$, ... , $(x_n, f(x_n))$ are n data points sorted in an ascending way in the independent x direction. This assumption states that x_0 is the smallest data and $x_n$is the highest data available in the x direction. The coefficient derivations are done by using equation 5.1 and plugging in the x coordinates of the points in that equation and solving for the coefficients left in that equation.

If x_0 is substituted for all the x's in equation 5.1, the coefficient a_0 could be evaluated.

$$a_0 = P_n(x_0) = f(x_0)$$

Similarly, by replacing all the x's with x_1's, and by using the value of a_0, it could be derived that

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

Thus, substituting x_2 for x in equation 5.1, and also using the terms a_0 and a_1, the expression for a_2 is:

$$a_2 = \frac{f(x_2) - 2f(x_1) + f(x_0)}{x_2 - x_0}$$

$$\textit{In other words, } a_2 = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

From now on, in computing the Lagrange Polynomial coefficients for any i^{th} point, $f(x_i)$ would be referred to as $f[x_i]$. Thus, the divided difference of $(x_i, f[x_i])$ and $(x_{i+1}, f[x_{i+1}])$ is defined as

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} \tag{5.2}$$

For clarity, these divided differences are assigned with orders depending on which level the differences were calculated at. The function values or the measured values(the $f[x_i]$'s) are called the *Zeroth Divided Difference*, the divided difference between $(x_i, f[x_i])$ and $(x_{i+1}, f[x_{i+1}])$ is called the *First Divided Difference* $f[x_i, x_{i+1}]$, the divided difference between $(x_k, f[x_k, .., x_i])$ and $(x_{k+1}, f[x_{k+1}, .., x_{i+1}])$ (where k *is less than* i) is similarly called the $(i-k)^{th}$ *Divided Difference* $f[x_k, ..x_i]$.

The computation of the divided differences could be understood better if the data is arranged in a tabular or array fashion every time a new difference is calculated. The

use of the Newton's Divided Difference in the Lagrange Interpolation is for generating the Lagrange Polynomial coefficients. The procedure is simple. All the measured data are put in a column next to the actual coordinates arranged in a column referring to those measurements. This way, if one is working with n sets of data, one has n members in the two columns to the left(corresponding to the coordinates and the measurements at those coordinates), and has $n-1$ members in the newly created column. Similarly, the next column is created by using NDD in the previous column and the first column members that correspond to that previous column. Accordingly, $n-2$ members are created in the newest column. Thus, every time a new column is created, that new column has one less member than the previous column. This results in a cone shaped table which is called the Newton's Divided Difference Table(NDDT). And the members at the edge of the cone, either at the top or at the bottom after excluding the member corresponding to the column at the most left(which is the coordinate information only), are the coefficients of the Lagrange Polynomial(Table 5.1).

By careful observation of this computation method, one can realize that for n sets of available data sets to generate a polynomial, n coefficients are generated by the Lagrange computation process. Thus, the polynomial is of degree n. Since the basis of the derivation of the coefficients are Equation 5.1, the derived coefficients a_0 through a_n refer to the coefficients in Equation 5.1.

This particular NDD just used in determining the Lagrange Polynomial is called the Backward Divided Difference because for computing $f[x_i, x_{i+1}]$, the set $(x_{i+1}, f[x_{i+i})$, which was considered later, is used for the initial basis of measuring

x_0	$f[x_0] = a_0$			
		$f[x_0, x_1] = a_1$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2] = a_2$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3] = a_3$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	...
		$f[x_2, x_3]$		$f[x_0, x_1, x_2, x_3]$
...	...		...	...
		...		...
x_{n-1}	$f[x_{n-1}]$		$f[x_{n-1}, x_n, x_{n+1}$	
		$f[x_{n-1}, x_n]$		
x_n	$f[x_n]$			

Table 5.1: *Newton's Divided Difference Table.*

discrepancies in both the numerator and the denominator of the right side of Equation 5.2. The other kind of NDD would be the Forward Divided Difference which would use the first set as the set of reference in measuring differences. Therefore, the Forward Divided Difference of the expression for the NDD for the sets $(x_i, f[x_i)$ and $(x_{i+1}, f[x_{i+1})$ would be

$$f[x_{i+1}, x_i] = \frac{f[x_i] - f[x_{i+1}]}{x_i - x_{i+1}}.$$

As easily observed here, the value of the NDD does not change for particular sets of data. Which one of the two Divided Differences to use in ones algorithm depends on how data is available to the user while the processing is going on. Mathematically, for Lagrange Polynomial coefficients determination, forward or backward divided difference does not matter. In this project, the backward divided difference is used.

5.2 Piecewise Linear Algorithm

The Piecewise linear approximation is by far the easiest algorithm to comprehend. The central idea of this algorithm is to set up a straight line between every pair of points next to one another in the coordinate frame. Then, by using the straight line equation, the algorithm interpolates any range data from the given coordinate information.

The Piecewise Linear Polynomial is expressed as $PWL(x)$ as follows:

$$PWL(x) = mx + b$$

This is the definition of a straight line where m is the slope of the line, b is the intercept

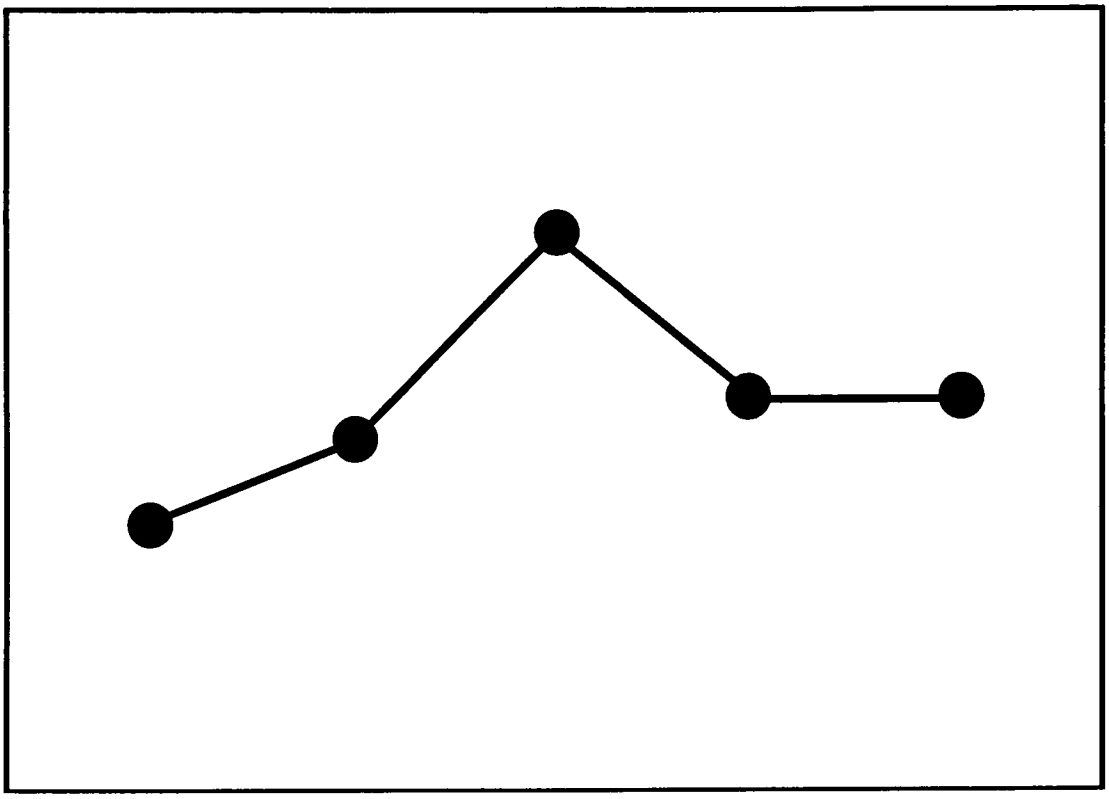


Figure 5.2: *The Piecewise linear function going through a set of data points. This linear polynomial simply draws straight lines between pairs of points.*

in the dependent axis(in this case, the $PWL(x)$ axis). In general, a piecewise linear polynomial can be viewed as data points interconnected by line segments. This is illustrated in Figure 5.2.

There is one limitation to this polynomial. This polynomial interpolation would fail when two samples are taken from the same coordinate. It is because the slope of the line in that case becomes infinity. It is true that if the sample magnitudes are the same, the zero difference in the numerator would overcome this infinity slope problem, but when the range sensors are involved, chances are, when the two samples are taken from the same coordinate, their values would not be the same. The range sensors

use the *time of flight* information to measure distances, and the real environment, in most cases, would show different readings even at the same coordinate if they are are taken at different instances. Therefore, one needs to be careful that sensor placement is done in order to avoid two sensor readings at the same coordinate.

5.2.1 Piecewise Linear Algorithm Computation Technique

Since the algorithms are defined in the two dimensional domain, this algorithm would also be described in the 2-D context. If one assumes that two known points are given to be (x_1, y_1) and (x_2, y_2), and a third point in between them is (x, y), the definition of slope provides us with the following equation,

$$\begin{aligned} \frac{(y - y_1)}{(x - x_1)} &= (y_2 - y_1)(x_2 - x_1) \\ Or, \quad y &= y_1 + (x - x_1)(y_2 - y_1)/(x_2 - x_1) \end{aligned}$$

One can note that the above is the equation of a straight line. Thus Piecewise Linear Interpolation interpolates along the line connecting neighboring nodes or points.

5.3 Cubic Spline Algorithm

The Cubic Spline Algorithm is a powerful method of approximating 2-D functions. From the name of the Algorithm, Cubic Spline, it is pretty evident that cubic or third degree polynomials are used in order to interpolate data points. The power of this algorithm lies in the principle same as the Piecewise Linear algorithm's principle. That is, the full scale of the independent part of the data set(the x direction on

an (x, y) plane if $y = f(x)$) is divided into small sub-intervals. Then a separate polynomial is evaluated for each sub-interval; in this case, the polynomials evaluated are third degree polynomials. The reason a third degree polynomial is evaluated in every sub-interval is because cubic polynomials are the simplest polynomials which have continuous derivatives throughout and are able to change directions in both increasing and decreasing directions of the interpolated values. In other words, cubic polynomials can "swing" both up and down in a standard cartesian (x, y) plane. Figure 5.3 shows how a Cubic Spline Algorithm generally interpolates between two points.

From the definition of the Cubic Spline algorithm, and also from Figure 5.3, one is able to note that Cubic Spline polynomials allow sufficient flexibility in not only having the continuous first derivative, but also having continuous second derivatives. From basic calculus theory, first derivatives are measures of the change in a certain function, whereas the second derivatives are measures of the "inflection points", points which separate the concaving-upward part of the function from the concaving-downward part of the same function. This continuous second derivative quality provides Cubic Spline polynomials with the "swing" capability. However, this polynomial does not guarantee that the derivatives, at the actual data points, would be the same as the derivatives in the original function. Nevertheless, the Cubic Spline interpolation does go *through* the original data points in each interval and in the global interval as well. In summary, the Cubic Spline interpolation in two dimensions look like a flexible rod smoothly bent at places to go through the actual measured data points [26].

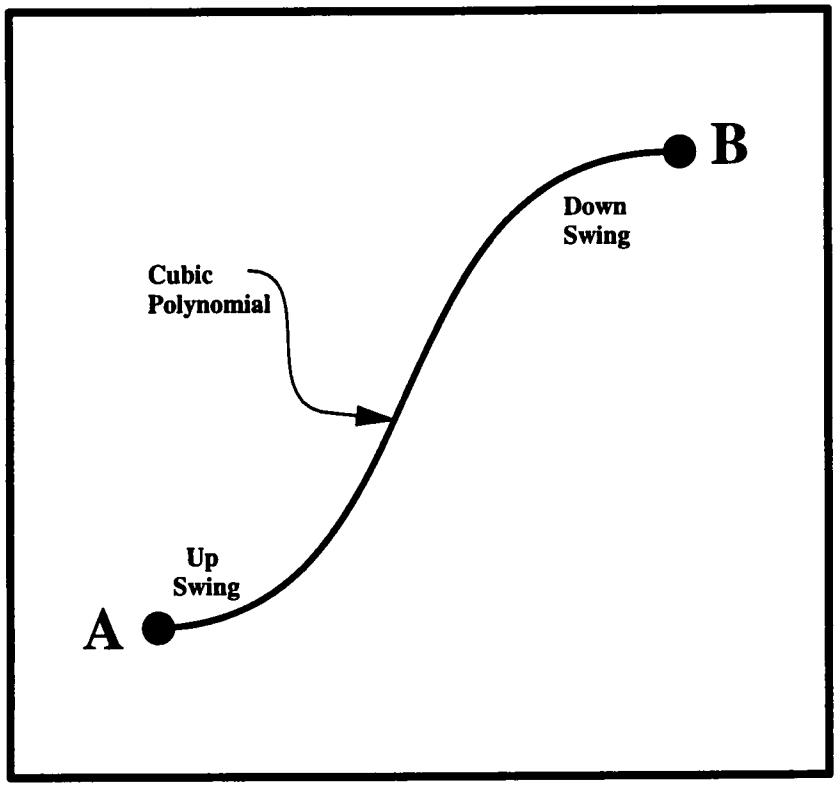


Figure 5.3: *The General Cubic Spline Polynomial connecting a sub-interval from point A to point B. The "upswing" and "downswing" are labeled in the diagram. This figure illustrates the polynomial's capability to adjust itself to faster changes in the data points through multi-swing features.*

As stated earlier, a Cubic Spline interpolation could be described as a set of third degree polynomials. In the two dimensional plane, if the n global intervals are $[x_0, x_1], [x_1, x_2], ..., [x_{n-1}, x_n]$, then there will be n cubic polynomials namely $CS_{0,1}(x), CS_{1,2}(x), ..., CS_{n-1,n}(x)$ derived, and the global interpolating polynomial would be

$$CS(x) = \left[\begin{array}{ll} CS_{0,1}(x), x_0 \leq x \leq x_1 \\ CS_{1,2}(x), x_1 \leq x \leq x_2 \\ ..... \\ ..... \\ CS_{n-1,n}(x), x_{n-1} \leq x \leq x_n \end{array} \right]$$

Conditions of the Cubic Spline Polynomial:

The basis of the Cubic Spline polynomial is developed on certain conditions. These conditions provide the polynomial with its smooth shape, its ability to go exactly through the true data points, its "swing" capability, and the other third degree polynomial attributes associated. The conditions are summarized below:

1. If the interpolating polynomial is $CS_{i,i+1}(x)$ *and the true function is* $f(x)$*, then* $CS_{i,i+1}(x_i) = f(x_i)$.

2. $CS_{i+1,i+2}(x_{i+1}) = CS_{i,i+1}(x_{i+1})$.

3. $CS'_{i+1,i+2}(x_{i+1}) = CS'_{i,i+1}(x_{i+1})$ *for* $i = 0, 1, 2, ..., n$.

4. $CS''_{i+1,i+2}(x_{i+1}) = CS''_{i,i+1}(x_{i+1})$ *for* $i = 0, 1, 2, ..., n$.

Conditions #1 and #2 establish the polynomial's path through the true data-points. Condition #1 makes sure that the interpolating polynomial includes the beginning point in each sub-interval in its path, whereas condition #2 assures the polynomial via the end-point of the same sub-interval. Condition #3 forces the slope of any two consecutive polynomials to be equal. This assures a smooth junction of the connecting polynomials. The last condition emphasizes on the concavity of the polynomials at the junction. By setting the second derivatives equal between connecting polynomials, it is made certain that there is no conflict in the concavity of the global interpolating polynomial, the global function which is a union of all the sub-interval polynomials.

5.3.1 Boundary Conditions for Cubic Spline Polynomial

There are two kinds of boundary conditions for the Spline algorithms in general, including this Cubic Spline algorithm [24] [28] [16]. The boundary conditions are named as follows:

1. *Natural Boundary, and*
2. *Clamped Boundary.*

A short description of these boundaries are given here.

Natural Boundary: The *Natural* boundary condition is the most reliable and easy condition to satisfy when the actual function describing the data points are unknown. That is, if there are no functions defined in order to describe the global behavior of the actual data, the *Natural* boundary condition is the "natural" way

of invoking interpolation function behavior at the endpoints. These end-points are called the "boundary" of the function in mathematical terminology. All this condition imposes is that the initial second derivative at the first sub-interval be equal to the final second derivative at the very last sub-interval. If the first piece of the interpolating Cubic Spline polynomial is $CS_{0,1}(x)$ and the last piece of the interpolating Cubic Spline polynomial is $CS_{n-1,n}(x)$, then the *Natural* boundary condition imposes

$$CS''_{0,1}(x_0) = CS''_{n-1,n}(x_n)$$

This condition forces the polynomial to avoid oscillations at the actual measured points by making the first derivatives of the two Cubic Spline functions to be equal at their junction. **This is the boundary condition used in the scope of this project.**

Clamped Boundary: The *Clamped* boundary condition invokes more constraints on the interpolating polynomial. This type of boundary condition needs the end-point derivatives to be equal to the end-point derivatives of the original function. If the interpolating Cubic Spline polynomials at the end-points are $CS_{0,1}(x)$ and $CS_{n-1,n}(x)$, and the original function is $f(x)$, then the *Clamped* condition invokes that the following be true:

$$CS'_{0,1}(x_0) = f\prime(x_0)$$

$$\text{and, } CS'_{n-1,n}(x_n) = f\prime(x_n)$$

That means, in order to impose *Clamped* boundary condition, the original function describing the data set must be known, or the end-point derivative values or their accurate approximations must be known. However, this raises a question about the applicability of this boundary constraint. If the function describing the data set is already known, there need not be many reasons to interpolate anymore. Actually, the derivative information at the endpoints should be enough to satisfy this condition. Therefore, a different method of interpolation may be used in order to impose *Clamped* boundary condition in the Cubic Spline approach. But, chances are, the derivative information obtained by another algorithm(such as Lagrange, or Linear) are of low resolution, and thus the *Clamped* condition could lead one to an erroneous approximation of the curves.

5.3.2 Computation Technique of Cubic Spline:

The two dimensional Cubic Spline polynomial in the interval $[x_i, x_{i+1}]$ is defined by the following function:

$$CS_{i,i+1}(x) = a_i + b_i(x - x_i) + c_i(x - xi)^2 + d_i(x - x_i)^3 \tag{5.3}$$

Equation 5.3 is the heart and soul of deriving the polynomial coefficients at each and every interval. The computation technique follows the four conditions listed earlier in this section along with satisfying Equation 5.3. First of all, by setting all the x's equal to x_i's, and by following condition #1, a_i is derived,

$$a_i = CS_{i,i+1}(x_i) = f(x_i)$$

Moving along the list of conditions, if condition #2 is applied,

$$\begin{aligned} a_{i+1} &= CS_{i+1,i+2}(x_{i+1}) = CS_{i,i+1}(x_{i+1}) \\ Or,\ a_{i+1} &= a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 \end{aligned}$$

The term $(x_{i+1} - x_i)$ is repeated commonly in this discussion. Therefore, it would be referred to in the shortened terminology as v_i. Therefore,

$$v_i = x_{i+1} - x_i \tag{5.4}$$

Now, all the other coefficients except for the a_i are undetermined until this stage. By taking derivatives on both sides of Equation 5.3, the b_i could be derived. After differentiating both sides of Equation 5.3, it is obtained,

$$CS'_{i,i+1}(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \tag{5.5}$$

By plugging in x_i for x in Equation 5.5, and by using Equation 5.4, it is evident that,

$$b_i = CS'_{i,i+1}(x_i)$$

If condition #3 is applied at this stage,

$$b_{i+1} = b_i + 2c_iv_i + 3d_iv_i^2 \tag{5.6}$$

If further steps are taken to differentiate Equation 5.5, the following is obtained,

$$CS_{i,i+1}(x) = 2c_i + 6d_i(x - x_i) \tag{5.7}$$

And, substituting x_i for x, c_i can be defined as

$$c_i = \frac{CS_{i,i+1}(x_i)}{2} \tag{5.8}$$

By applying condition #4, the future c_i could be defined,

$$c_{i+1} = c_i + 3d_i v_i \tag{5.9}$$

The only unknown coefficient is now d_i. And one derive d_i easily by solving Equation 5.9.

$$d_i = \frac{c_{i+1} - c_i}{3} \tag{5.10}$$

These coefficients need to be iteratively determined within every sub-interval of the Cubic Spline polynomial. After deriving all the a_i, b_i, c_i, and d_i's, the global view of the Cubic Spline polynomial is obtained.

5.4 Comparing the Algorithms

The three algorithms, the Lagrangian, the Linear, and the Cubic Spline, can be compared with one another on the same grounds. The computational difficulties vary from one algorithm to the other—the Linear interpolation being the simplest while the Cubic Spline the most complex among the three. But, interestingly, none of these algorithms require extensive calculations by any means. That is why instead of comparing their computational methods, only their performances along curves would be compared. As one can see in Figure 5.4, The Lagrange polynomial tends to oscillate towards the end points. But, throughout other regions, Lagrange polynomial is a smooth polynomial interpolating through the data points and is a one-piece polynomial throughout. The linear interpolation, as expected, is monotonously increasing or decreasing between each pair of points. This polynomial is strictly based on the

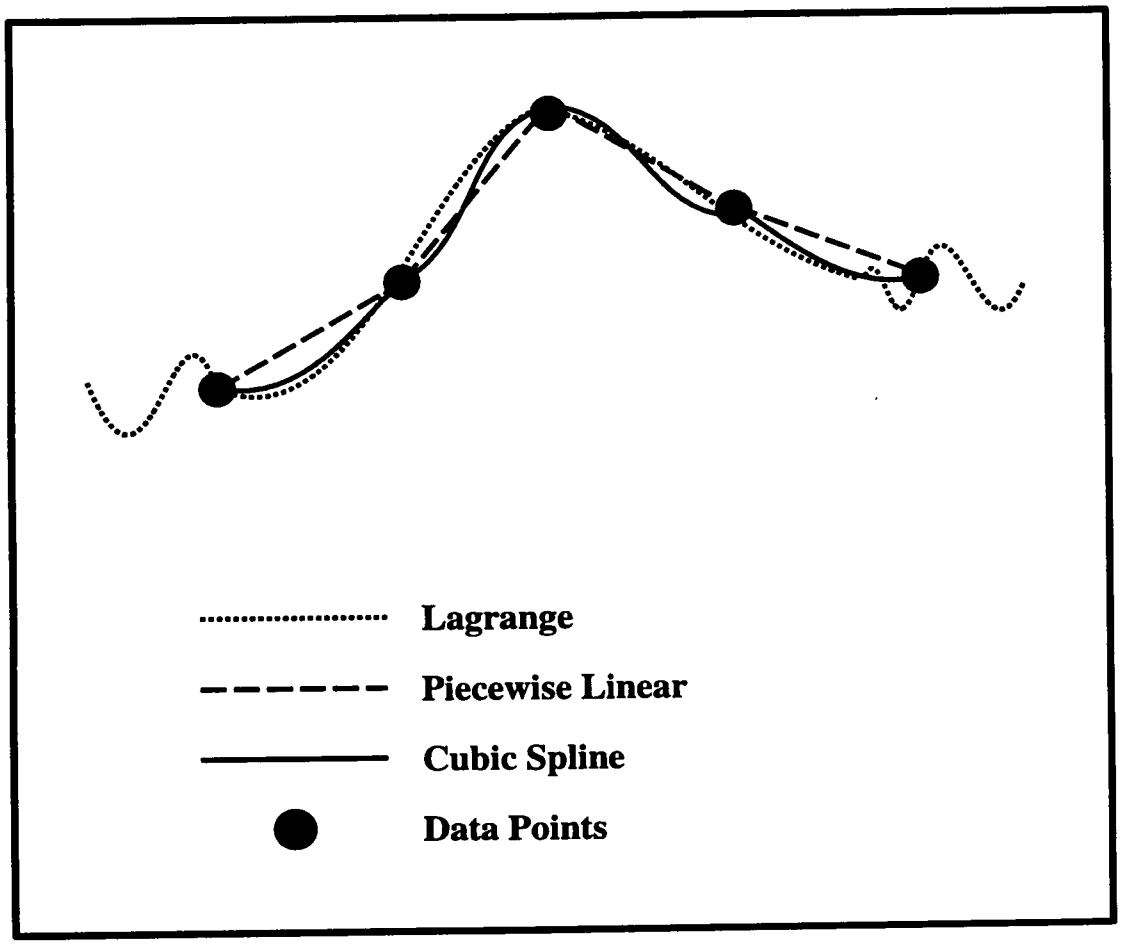


Figure 5.4: *Comparison among the three polynomials going through the same points.*

positions of each data point. Therefore, it does not bias towards the rate of change of the data at each point. The cubic spline polynomial, however, has both upswing and downswing in each interval. This provides the polynomial with low derivatives near each real measured point.

As far as to which polynomial is the best approximation, none can answer that question with confidence. That information is dependent on the original function itself. If the original function tends to be linear, the linear approximation would obviously be the best fitting curve. Likewise if the original function is oscillatory at the ends and smooth towards the middle, Lagrange is the prime candidate. The performances of these algorithms expanded in the 3-D context would therefore be depended on the actual surface the polynomial is trying to interpolate. Also, as focused in this thesis that robot navigation is the reason behind this surface study, navigation may be fitted better for a particular type of algorithm even though that algorithm might not have the best accuracy analytically.

5.5 The Chebyshev Polynomial

The Chebyshev polynomial is a familiar polynomial in designing digital filters [29] and thus is common knowledge to people in the areas of Digital Signal Processing(DSP) and Image Processing(IP). In the DSP and IP environments, Chebyshev polynomials have shown consistent performance even with inconsistent and noisy data sets and for higher degree filter implementations. The passband ripple in the digital filters using Chebyshev polynomials hardly ever causes any problems for practical

applications such as digital noise removal and audio signal processing. Therefore, it is reasonable to think about Chebyshev polynomials in other practical applications such as surface construction discussed in this thesis.

The Chebyshev polynomial sets are orthogonal polynomials with respect to certain weight functions influencing the polynomials' nature [26]. For ease of understanding, this polynomial would also be described in 2 dimensions. The Chebyshev polynomials would be denoted as $T(x)$ throughout this discussion.

The Chebyshev polynomial is an iteratively constructed polynomial. In each iteration, the degree of the polynomial increases. Each stage is also dependent on the two previous stages. In simple mathematical terms, the Chebyshev polynomial can be described by the following:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{5.11}$$

Here, n is a particular iteration in the determination of the polynomial. As it can be easily seen, the entire polynomial determination would be characterized by the approximation of the first two initiating polynomials $T_0(x)$ and $T_1(x)$. These selections are also dependent on the basic function with respect to which these polynomials are orthogonal to one another. Orthogonality of the polynomials is directed by specific conditions. A good test of orthogonality in the interval [-1,1] with respect to a weight function $w(x)$ would be,

$$\int_{-1}^{1} \frac{[T_n(x)]^2}{w(x)} dx = \frac{\pi}{2} \tag{5.12}$$

The more explicit definition of the orthogonality property of the Chebyshev polynomial could be used to determine the initial or starting stage of a Chebyshev polynomial

of a later stage. Let the weight function, with respect to which the Chebyshev polynomials have orthogonality, be denoted as $w(x)$. Now, orthogonality requires that the following be true,

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{w(x)}dx = 0 \tag{5.13}$$

Here the n and the m are two different stages of the Chebyshev polynomial. Now depending on the weight function, the $T_0(x)$ and $T_1(x)$ are chosen. A popular practice is to use the derivative or the multiple of the derivative of the weight function as one of the functions, and adjust the other one to satisfy orthogonality.

Chebyshev polynomials, however, require a specific set of coordinate location restrictions at which the data need to be obtained [24]. This optimal spacing of points enable the polynomial to reduce the error compared to other approximation techniques such as the Lagrange interpolation method. Also, Chebyshev polynomial helps reduce the degree of the polynomial with minimum loss in accuracy. That is, a set of data points interpolated by an n^{th} degree Lagrange polynomial can be interpolated by a $< n^{th}$ degree Chebyshev polynomial with greater or equal accuracy. However, the optimal spacings of the coordinate locations are required, and this constricts the use of Chebyshev polynomials where spatial sensor positioning is not recommended to be absolutely strict, or geometrically difficult to calibrate.

Nonetheless, the optimal spacing method of the Chebyshev node determination is briefly mentioned here. If the Chebyshev nodes are referred to as x_k's($k = 0, 1, 2, ..., n$), then the expression for x_k is,

$$x_k = cos(\frac{2k-1}{2n}\pi) \tag{5.14}$$

It is not difficult to imagine that it would be rather difficult to find a method of running this Chebyshev algorithm in multiple directions to match with the sensor positioning of the approach taken in this thesis. That is one of the major reasons why this polynomial was not expanded to three dimensions.

5.6 Summary

Polynomial based surface interpolation technique is the major emphasis in this thesis. Three interepolative polynomials—the Lagrange polynomial, the Piecewise Linear polynomial, and the Cubic Spline polynomial—were studied thoroughly for implementation. These three algorithms showed potentials for interpolating through varied sets of data points within the limitations posed by surface generation method outlined in Chapter 6. Another popular algorithm in Electrical Engineering called the Chebyshec polynomial, used extensively in designing digital filters, was also reviewd; but Chebyshev algorithm could not be applied in this project because this particular algorithm contradicted with some of the requirements of the surface approximation technique discussed in this thesis.

CHAPTER 6

Surface Generation

Three Dimensional Surface generation is the principal concentration of this study. Interpolation techniques described in Chapter 5 are used with the sensor placement technique in Chapter 3 in order to generate surface information from the insufficient range data available [3] [19] [30] [31] [32]. The reason the data available is referred to as insufficient range data is because only a few sampled points on the surface are considered as the input to the system, and a continuous function for the entire surface region is generated from that set of input data. Subsequently, the depth information of *any* point on the surface under consideration could be obtained from the continuous function generated from the interpolation techniques over the careful sensor placement geometry of the system. In summary of the Surface Generation technique, it could be said that **the input to the system is a set of discrete data representing the depth measurements at different points on the surface, and the output of the system is a continuous function interpolating the depth measurement at every point on the region of interest.** Figure 6.1 illustrates the system from a block diagram input/output point of view.

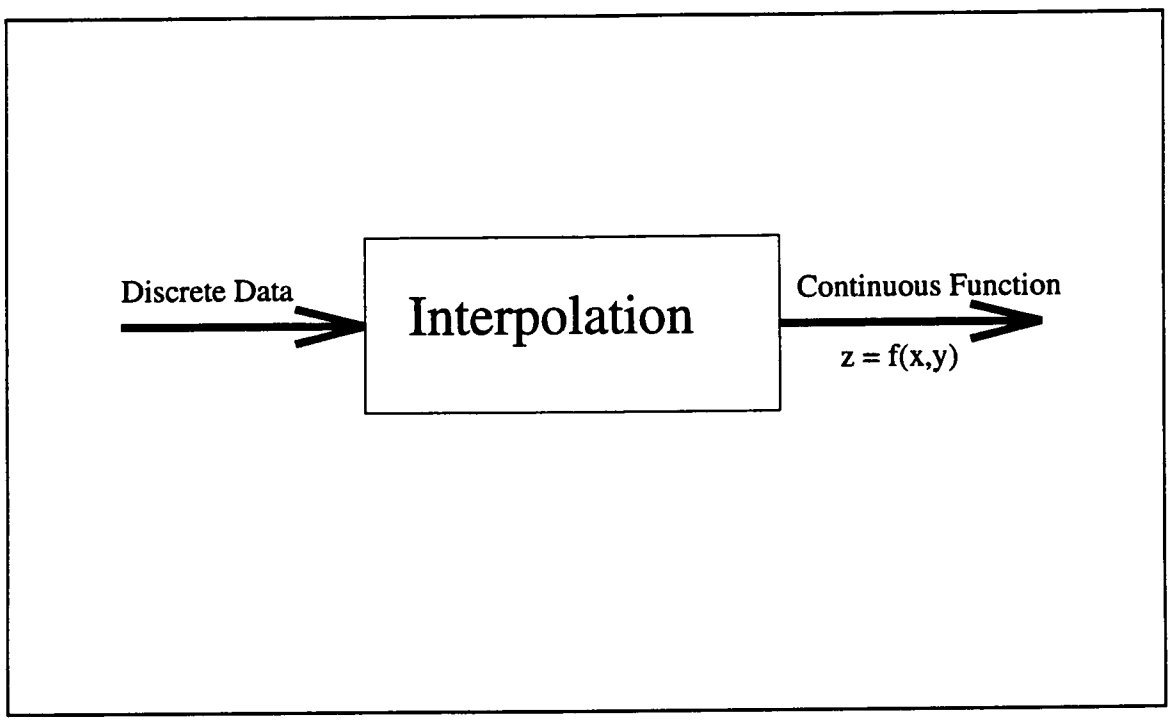


Figure 6.1: *The system block diagram from an input/output(I/O) perspective.*

6.1 Technique involved in Surface Function Generation

The continuous function describing the 3-D surface is obtained by running the 2-D interpolation algorithms described in chapter 5 on the sensor placement described in chapter 3. Therefore, the 3-D surface generation could be thought of as a careful and thoughtful combination of the 2-D interpolation technique and the sensor placement geometry. This combined procedure of obtaining a continuous function to measure surface depth could be achieved in the following steps:

1. *Choose a 2-D interpolation algorithm.*

2. *Run the 2-D interpolation algorithm over the entire surface of interest by rows, columns, and diagonally distributed data points. Thus, the data are upsampled.*

3. *Organize the upsampled data in rows, columns, and diagonals in order to regenerate the Square Array Organization(SAO).*

4. *Mask a region of interest by separating a square region from the larger square data set.*

5. *Create the Weight Vector(WV) to assign Row, Column, and Diagonal weights for the generating polynomials in those directions.*

6. *Create the Polynomial Vector(PV) by computing polynomials inside the mask, and take the scalar product of the Weight Vector and the Polynomial Vector to generate the continuous depth function inside the mask.*

7. *Move the mask on the entire surface region, repeat steps 4, 5, and 6.*

8. *Combine all the continuous functions generated in step 7, and use arithmetic averaging to generate a single function describing the entire surface.*

The result of the above steps is a continuous depth map, z = f(x,y).

For the experiment base of this particular project, a 3 by 3 array of actual measured data were used. These 9 datapoints were described in the usual Cartesian *(x,y,z)* coordinates' notation where the z direction, as described earlier in this chapter, refers to the depth of the surface at the corresponding *(x,y)* location.

As the very first processing on the data set, the points were normalized in both the x and the y directions in order to get free from the confusion of the sensor placement, a placement which could have been implemented as a rectangular organization instead of a square organization. The lower limit of either the x or the y direction was 0.0 and the upper limit was decided to be 1.0. These numbers were chosen in order to standardize a minimum and a maximum limit on the kinds of data available to the system.

Then each of the rows were driven through a 2-D interpolation algorithm to generate two extra points in each row. After completing through the rows, the same method was followed for the columns and the diagonals. Thus, an upsampled total data set with 25 total points are created. Among these 25 points, 16 are newly generated or interpolated points, and the other 9 are the original measured points. Data sets can be expanded to 49 data points by upsampling even further, but, usually, the test points need to have reasonable inter-spacings such that the errors in the upsampling technique do not overwhelm the adaptive nature of the polynomial interpolation methods. This expanded set of data is the obtained upsampled data which play an important role in generating a continuous description of the surface contour. Surface contour is generated by updating some parameters which are described later in this

chapter.

It would be easier to describe how a continuous depth function is generated within a mask. In order to do so, two terms, the Polynomial Vector(PV) and the Weight Vector(WV), need to be defined.

Polynomial vector(PV):

The Polynomial Vector(PV) is a vector which has polynomials as its individual elements at different directions. Each element of this PV vector is associated with a certain angle measurement corresponding to the angle at which the 2-D interpolating polynomial lies in the square array organization of the sensory mask. Therefore, the considered directions in this system are the 0^o angle for the rows, the 90^o angle for the columns, and the 45^o angle for the diagonals. This is illustrated in Figure 6.2. There could also be a 135^o angle for the diagonals, but the output generated from that set of information would be redundant since all the upsampling regions are already covered with the three angular directions. As a result, the Polynomial Vector(PV) is defined to be the following for n directionalities:

$$\vec{PV} = [P_1\ P_2 ... P_n]$$

And for the purpose of this project, the polynomial Vector is:

$$\vec{PV} = [P_{row}\ P_{col}\ P_{dia}]$$

Here P_{row}, P_{col} and P_{dia} refer to the interpolating polynomials in the directions of the row, column and diagonal respectively.

Weight Vector(WV):

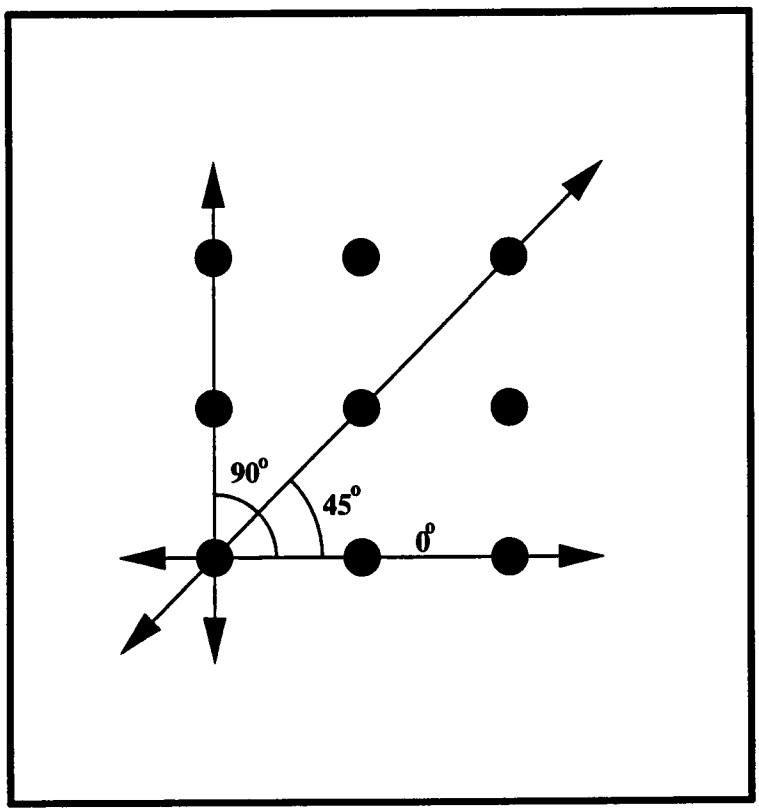


Figure 6.2: *The Polynomial Vector Directions in a 3 by 3 square array. The direction angle for rows are* $0°$*, for columns the angle is* $90°$ *, and for diagonals it is* $45°$*.*

The Weight Vector is defined to be a vector having n elements representing the weight or importance of n elements in some other vector. The arithmetic summation of the n elements of the WV is 1.0. Therefore, once this Weight Vector(WV) is put through a dot product or a scalar product with the associated n dimensional measurement vector, an independent resultant magnitude is achieved. This magnitude is a depth-measurement of the representation of a particular coordinate or variable by the Weight Vector and the associated measurement vector. In this project, the Weight Vector(WV) assigns importance to the different polynomials in the Polynomial Vector(PV). The resultant of the dot product of the WV and the PV is the depth measurement of the surface under consideration.

The Weight Vector(WV) could be expressed in n direction space by the following:

$$\begin{aligned} \vec{WV} &= [W_1\, W_2 \ldots W_n] \\ \sum_{i=1}^{n} W_i &= 1.0 \end{aligned}$$

In this particular study, however, WV is defined as:

$$\begin{aligned} \vec{WV} &= [W_{row}\, W_{col}\, W_{dia}] \\ W_{row} + W_{col} + W_{dia} &= 1.0 \end{aligned}$$

The Polynomial vector is achieved by running the 2-D interpolation algorithm along rows, columns, and, diagonals. Therefore, the first element of the Polynomial vector is $P(x)$ since the row coordinates vary in the x direction. Accordingly, the second element of PV is $P(y)$, and the third element is $P(x,y)$.

The Weight Vector is determined based on shortest distance measurement. If the desired coordinate to approximate depth is (x_d, y_d), then the point closest to

(x_d, y_d) in the upsampled data set is identified. For example, let this distance be $minimum[dist((x_d, y_d)(x_i, y_i))]$. Here, the *'dist'* function is the cartesian distance defined as,

$$dist = \sqrt{(x_d - x_i)^2 + (y_d - y_i)^2}$$

The (x_i, y_i) pair represents all the upsampled and original points in the square array.

Now, if the closest point in the upsampled data is (x_n, y_n), the three polynomials—the row polynomial, the column polynomial, and the diagonal polynomial—through that closest point is identified. These are the three elements in the Polynomial Vector(PV). The distance of the destination coordinate (x_d, y_d) from these polynomials are measured. In order to make this part of the discussion clear, these measured distances would be called d_{row}, d_{col}, and d_{dia}(Figure 6.3). The weight elements of the Weight Vector(WV) is achieved through the Lever Law, which is basically a linear weighting procedure as follows:

$$Weight\ in\ row,\ W_{row} = \frac{d_{row}}{d_{row} + d_{col} + d_{dia}}$$

$$Weight\ in\ column,\ W_{col} = \frac{d_{col}}{d_{row} + d_{col} + d_{dia}}$$

$$Weight\ in\ diagonal,\ W_{dia} = \frac{d_{dia}}{d_{row} + d_{col} + d_{dia}}$$

It is evident that $W_{row} + W_{col} + W_{dia} = 1.0$, which was the basis of the Weight Vector(WV).

Now that the Polynomial vector and the Weight Vector are calculated, one can use the dot product in order to define a continuous description of the surface. The

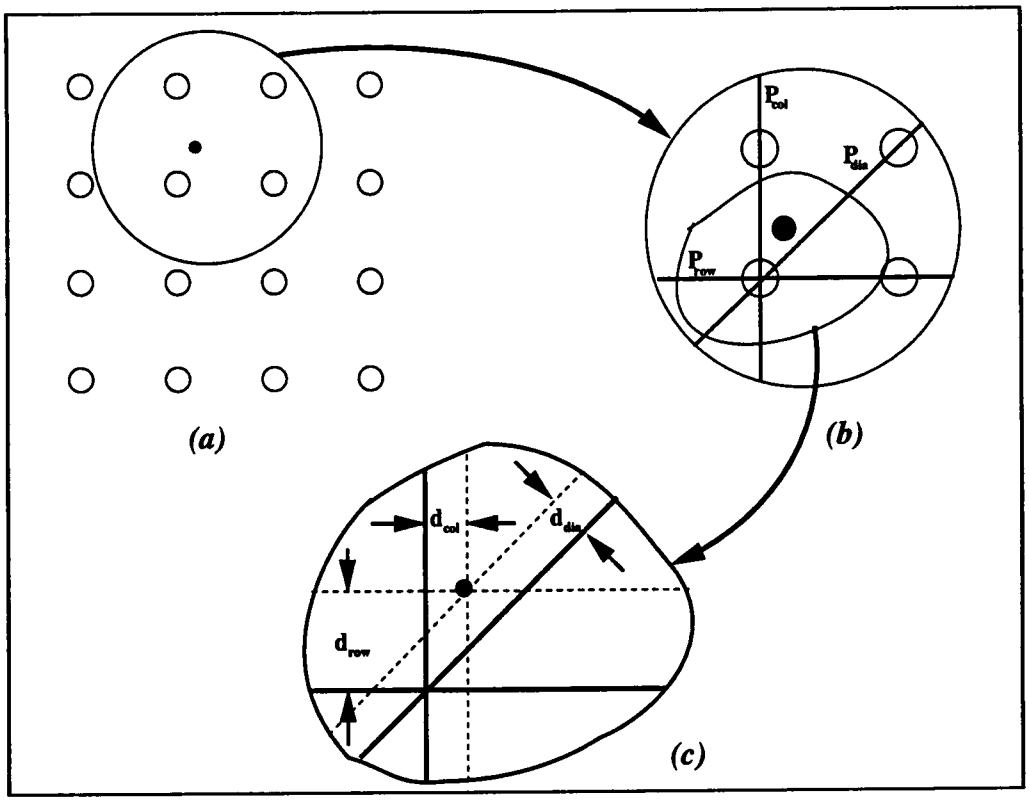


Figure 6.3: *The elements of the Weight Vector(WV) generation. The dark dot in the images show the location at which depth is to be estimated. (b) is a zoomed version of a circular region in (a). Figure (c) is a zoomed portion of (b) where the WV component generating elements* d_{row}, d_{ol}, *and* d_{dia} *are shown.*

depth function, as described earlier, would be defined by **z(x,y)**. And the expression for the depth at any point (x, y) on the surface is:

$$z(x, y) = \vec{PV}.\vec{WV}$$

This is an equivalent expression of the following:

$$z(x, y) = P(x)W_{row} + P(y)W_{col} + P(x, y)W_{dia}$$

6.2 Masking

Before performing the above mentioned operations, the need to mask on the region of interest is very important. Masking could be done in more than one stage. Once the upsampled data is obtained, the region around the point of interest could be even upsampled further. Also, additional masking could be done to center the point of interest under the mask. This probable multiple masking procedure is shown in Figure 6.4.

Based on the above discussion, it could be understood that by updating the Polynomial Vector(PV) and the Weight Vector(WV), the surface depth could be interpolated within the scope of the accuracies offered by the basic data interpolation algorithms. The masking procedure could also be updated. The initial mask, in all cases, is the entire region under consideration. Then subsequent smaller masks can be achieved until the polynomial vector and the weight vector do not change significantly. For the purpose of this project, once the change in subsequent PV and the WV are less than 15% to 20%, the mask-updating procedure was stopped. Figure

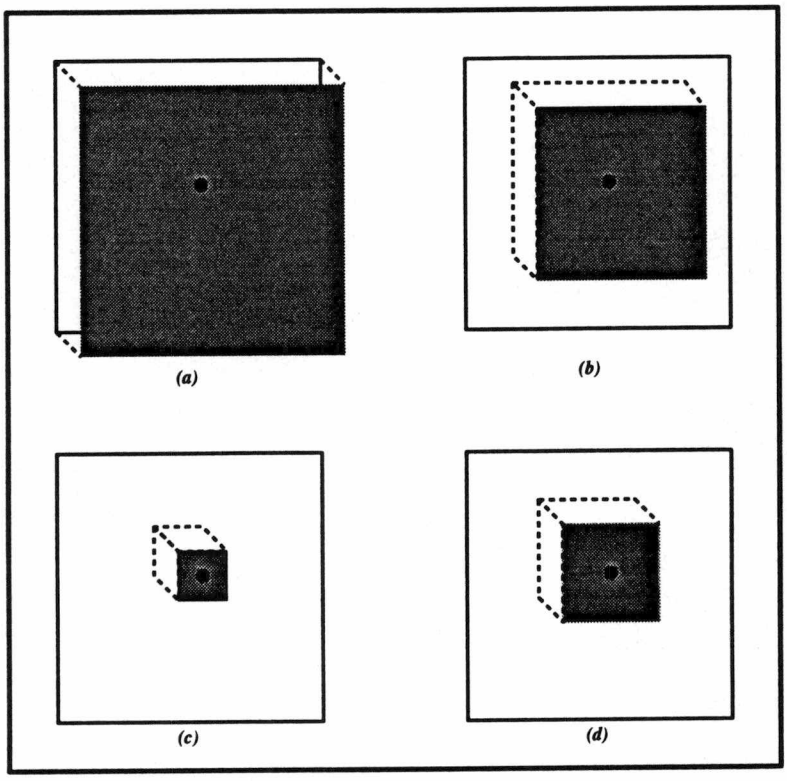


Figure 6.4: *The updating of the mask. The dark dot in each figure represents the goal point at which depth is to be extracted. Figure(a) is the first step of masking where the entire region is considered. (b), (c), and (d) are the smaller masks around the goal point to get better polynomial and weight vectors.*

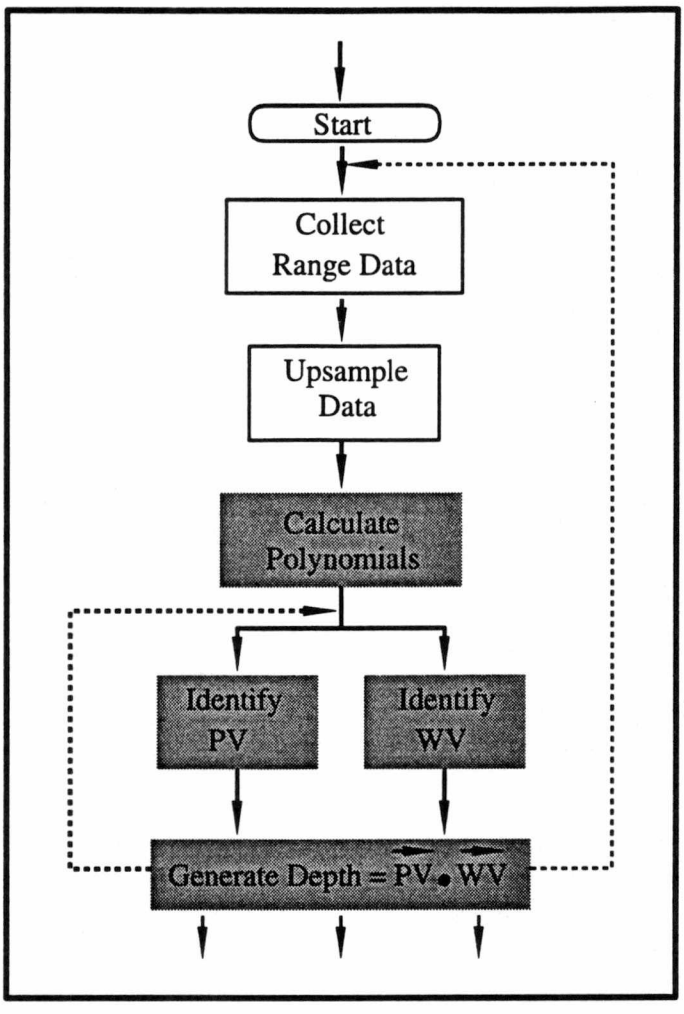


Figure 6.5: *The algorithm: the major steps to generate depth.*

6.4 shows how the mask-updating method reduces the size of the mask itself. Reduction in size of the mask refer to the PV elements being closer to the goal point, and, therefore, more accurate interpolation. The entire surface interpolation method could be viewed as in figure 6.5.

6.3 Summary

Surface interpolation is achieved through a vector based approach using computation of polynomials and some additional but computationally simple steps. Polynomial Vectors(PV) and Weight Vectors(WV), as defined in this chapter, are computed first. Then a scalar product between the PV and the WV generates the interpolated depth which is seeked to determine the surface contour.

CHAPTER 7

Experimental Validation Using Hardware

A significant amount from this thesis project has been implemented and tested on real hardware. The reason is simple. As an engineer it is always the goal of a researcher to *try out* theoretical options in real life situations. Even though it is true that theories cannot always be implemented on hardware without any modification what so ever, it is a valuable experience to even modify some of the theories and hardware implement them.

The hardware implemented portions of this thesis include three different areas of robot navigation in obstacle ridden environments. The tests were mainly designed for the available mobile robot in the laboratory R2M2, which is a tracked mobile robot. The first among these is the heart and soul of this thesis, and that is the surface interpolation from real range data collected. The second is the approach of the robot towards an obstacle, making sure that the robot does not tip over and have an accident. The last, but certainly not the least, is the reactive automatic obstacle negotiation along a straight line. Reactive systems have been utilized successfully in the past for robot navigation [33] [34].

7.1 Surface Interpolation from Real Range Data

The surface interpolation technique used in this thesis is based on the Square Array Organization(SAO) of sensors. Chapter 2 describes this SAO sensor placement required for surface interpolation. This type of sensor placement is implemented synthetically, but a real sensor placement on a square plate is also built where the ultrasonic range sensors are mounted.

A square steel plate of about 2 feet on each side holds the range sensors. A 3 by 3 array of range buttons were mounted on this steel plate. The sensors are read and controlled by the circuits and organization described in Chapter 3.

There is one parameter that would limit the implementation of the SAO structure, and that is the inter-distance between the sensors. The sensors need to be mounted on the plate in such a manner that the inter-distances between the sensors could be changed for experiment purposes with ease. This obviously limits the easiest mounting method of using brackets and screws to mount the buttons on the plate. But on the other hand, the brackets are required to hold the range buttons. Therefore, the brackets holding the sensors needed to be attached but not bolted on the plate. Small sized magnets were used to hold the brackets on the plate, but this arrangement also kept the sensors free to be replaced at desired different locations without any difficulty.

7.2 Turning of R2M2 on Obstacles

At one stage of the experimentation process, ultrasonic range sensors were mounted on both sides of the front tracks of R2M2. This additional integration of sensors gave R2M2 capabilities of looking at three dimensional surfaces from its front tracks instead of the 2-D curve along a line. The information obtained was processed to get interpolated surfaces, and that interpolation result was used to guide the robot along its navigation path.

The robot was initially placed in front of an obstacle in such a manner that only one of the side track-sets of the robot was lined up with the obstacle. The other side had a free navigation path. It is obvious that the robot would tip over on one side if it tried to negotiate the obstacle if it approached the obstacle straight forward from such a position. After guiding through surface interpolation, R2M2 was successful in realizing that its body position needed to be adjusted if successful negotiation is to be achieved.

The robot initially started climbing straight ahead, but soon it realized the absence of obstacle on one side. Therefore, it backed up, turned toward the direction of the obstacle, and tried further to negotiate. Therefore, by the virtue of surface interpolation, R2M2 was able to go towards the center of the obstacle by taking automated turns by itself.

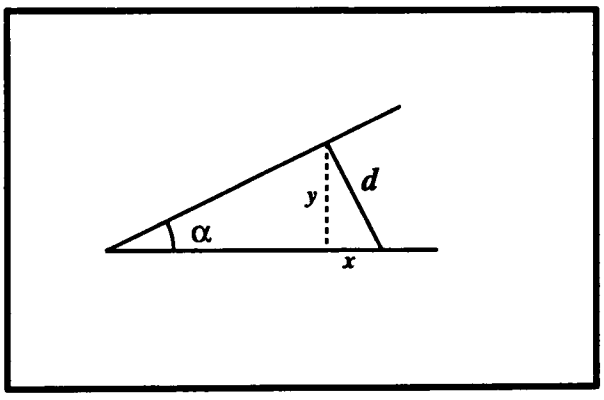


Figure 7.1: *Incorporating range and inclinometer information.*

7.3 Implementation of the Obstacle Negotiation Algorithm on R2M2

The reactive obstacle negotiation algorithm is based on two dimensional polynomial data interpolation. There is an array of ultrasonic range sensors mounted along one side of the R2M2 robot. Range data from those sensors are interpreted through a third degree polynomial. The track positions of the robot is informed by three inclinometers mounted on each of the tracks.

The robot received data from the ultrasonic range finders and the inclinometers. The range finders along with the inclinometer information provided points in two dimensions on the path of motion according to the following equations:

$$x = d.sin\alpha$$

$$y = d.cos\alpha$$

Here d and α are the range and the inclinometer information as seen in Figure 7.1.

Now, the coordinate information obtained would be used to set up a third degree polynomial through the path points, and the relative slopes at the revolute joint

positions of the robot would be calculated in order to learn how further the tracks need to move to be able to match the robot's body to the path. The inclinometers would provide the angle of the tracks at all times.

The above was the initial setup. But, due to the low resolution in the inclinometer readings, and due to the errors involved in the inclinometer readings because of the jerkings in the robot motion, inclinometers were taken off of the robot. Therefore, the current setup is based on the ultrasonic range sensors only.

Since range data are the only information available to the system at this point, relative slopes, instead of absolute slopes of the track inclinations are measured. Therefore, the robot is now trying to take any third degree polynomial generated by the sensory data, and move the tracks in such a manner that the curve gets closer and closer to a straight line. When the curve gets very close to a straight line, it indicates that the tracks are more or less matched with the terrain, and that maximum traction is achieved between the path and the tracks of the robot.

Figure 7.2 delineates a few different stages during the reactive obstacle negotiation based on a 2-D curve interpolation. Figure 7.2(a) shows the reactive track adjustments of R2M2 encountering a log or a beam on the floor. Figure 7.2(b) is a picture of R2M2 going down a ramp. Figure 7.2(c) and (d) shows the behavior of R2M2 while approaching and also while climbing stairs. These pictures are drawn in the SAVIC(Simulation Animation Visualization and Interactive control) software [35] and are obtained from there. The simulation of the obstacle negotiation algorithm is implemented by adding onto the SAVIC software and the simulation of R2M2 associated with it. The actual hardware, the robot R2M2 itself, was also controlled by

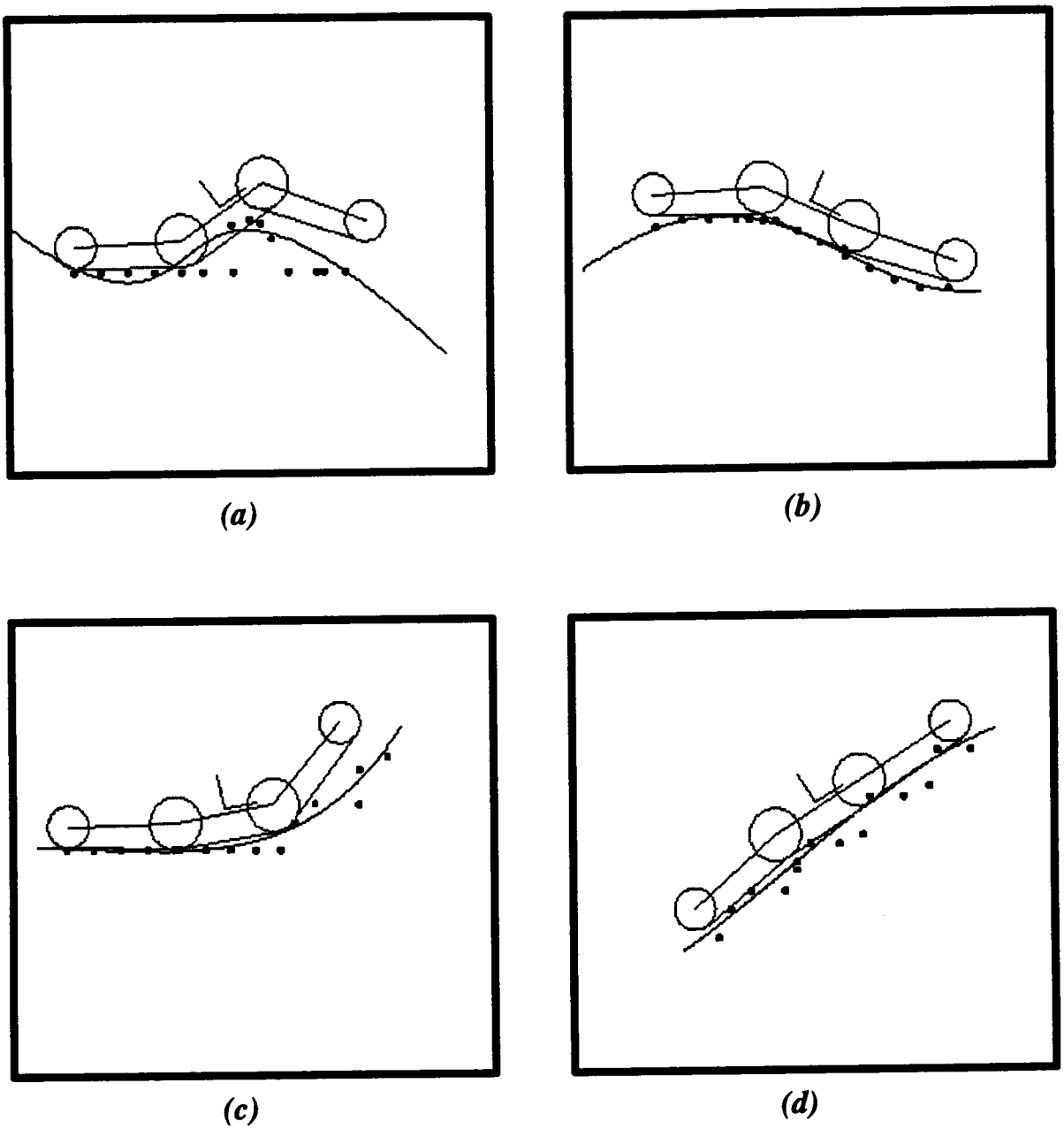


Figure 7.2: *Two dimensional obstacle negotiation of R2M2. Figure(a) shows climbing over a log as seen by the measured or dotted points. Figure(b) is negotiating a ramp, while Figures (c) and (d) are climbing stairs. Note how a curve is fit through the data points in each figure.*

the automatic obstacle negotiation separately. The real robot is not sensitive to the shapes of the obstacles or the arrangement or sequence of different obstacles in its path. The only limiting factor is that the physical dimension of the obstacles need to be such that the robot can negotiate the obstacles without violating its geometrical and mechanical structure.

7.4 Summary

A series of experiments with real harware are completed in order to test the polynomials and the surface approximation techniqe. A hardware setup for SAO(Square Array Organization) is built, and real surfaces are interpolated through the algorithm developed here. The results are qualitatively analyzed and show robustness on various surfaces having different contours. 3-D interpolation is transported on R2M2 and successful turns are taken by R2M2 in approaching an obstacle properly. Also, polynomials are used in guiding the robot automatically negotiate obstacle courses with obstacles placed arbitrarily.

CHAPTER 8

Graphical User Interface

A user interface for the viewing and organizing of data was developed as a part of the package presented in this thesis. The purpose of this user interface was to make the software and hardware more accessible to any user, even the ones not familiar with the software or hardware packages used in the project. Simple viewing area enables the user to view any three dimensional plot chosen to be viewed, and the control panel on the side provides the user with a variety of options of choosing data types, interpolation techniques, noise analysis, and other options of such sort. X-Windows toolkit and the Motif utilities were used to design and realize the interface.

Telerobotic systems are also another area where user interfaces become important [36]. The mobile robot R2M2 has teleoperation capabilities. Therefore, a graphical user interface is built for easier access of an operator to the robot from the computing system available in the laboratory.

The description of developing the user interface can be split into two areas in which the majority of the efforts were expended. These are the Drawing Area for 3-D mapping, and the Control Panel for making resources available. A brief discussion of each is provided here.

8.1 The Drawing Area

X-Windows utilities have the techniques available to a programmer to create a drawing area and draw lines, points, and various sorts of curves on the two-dimensional plane containing the drawing area. Therefore, viewing lines and points is only a matter of patience in creating the proper widgets and sending commands for drawing. The key to viewing 3-D surfaces lies in the transformation of 3-D data available into 2-D data projected on a plane. In other words, the 3-D to 2-D mapping is essentially the critical issue in viewing 3-D structures. The approach taken in establishing the Drawing Area, therefore, is based on converting the 3-D data to an isometric view of the structures on the 2-D plane. There could be other 3-D viewing such as perspective viewing chosen. But the reason isometric views were chosen was because the equations involved are fairly simple, and no perspective viewpoint needs to be specified. And since the interest of this project is to understand 3-D surfaces, a generalized isometric description is sufficient.

The user interface was planned to be a plane frame or box with the drawing area along its left side and occupying most of the window. The control panel was designed to be at the right side with the utility widgets. The equations that transform the 3-D (x, y, z) data into the 2-D (x, y) data are described here.

As described earlier, the mapping of the volumetric data was based on the isometric views of three dimensional objects. In order to understand these equations, one must be aware of the fact that both the x and the y coordinates in the three dimensional world is mapped into the (x, y) coordinates of the two dimensional world.

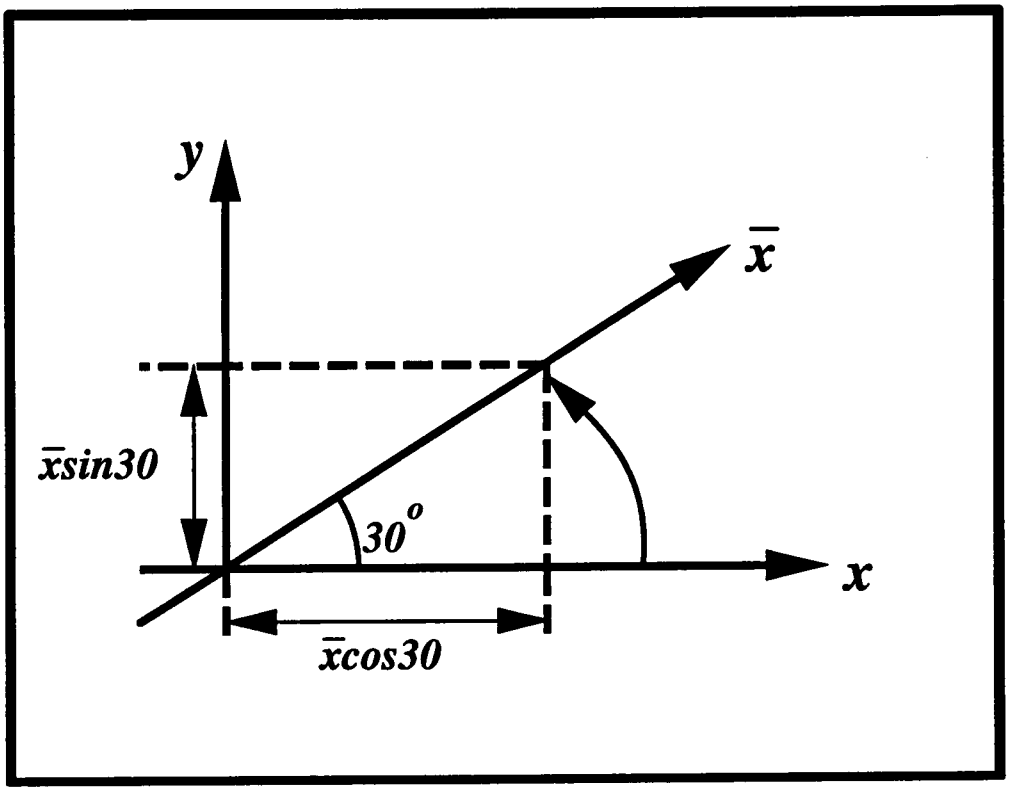


Figure 8.1: *The transformation of only one coordinate(in this case, the x coordinate) of the 3-D world into the 2-D world.*

Also, the depth, or the z coordinate, of the 3-D world is a combination of the actual z parameter and the actual x and y parameters now mapped into the (x, y) coordinates in the plane of drawing.

Figure 8.1 is useful to understand the mapping that takes place in the drawing procedure. The mapping of only one coordinate can be discussed at first. The x coordinate is considered in this case. The 3-D world coordinates are assumed to be $(\bar{x}, \bar{y}, \bar{z})$, and the drawing coordinates are assumed to be (x, y). The viewing angle is chosen to be 30° as it is commonly used in isometric viewing.

Figure 8.1 shows that for an angle of 30^o, the world coordinate $\bar{x}$ transforms into

$$x = \bar{x}.cos30^o \tag{8.1}$$

Consequently, the world coordinate $\bar{x}$ generates a y component on the drawing plane as follows,

$$y = = \bar{x}.sin30^o \tag{8.2}$$

Thus either the x or the y coordinate in the 3-D environment generates both x and y coordinates on the drawing plane.

Now, if one looks at Figure 8.2, the transformation of both the $\bar{x}$ and the $\bar{y}$ coordinates together becomes more comprehensible. The way the drawing plane is spread out, the x coordinate on the drawing plane becomes a difference between the contribution in the x direction by the world $\bar{x}$ and $\bar{y}$ coordinates. Also, the y value on the plane becomes a summation of the contribution in the y direction by the world $\bar{x}$ and $\bar{y}$ coordinates. The transform equations are as follows,

$$x = \bar{x}.cos30^o - \bar{y}.cos30^o \tag{8.3}$$

$$y = \bar{x}.sin30^o + \bar{y}.sin30^o \tag{8.4}$$

Also, the height in the world coordinate translates into a summation in the y direction of the transform equations.

$$y = \bar{x}.sin30^o + \bar{y}.sin30^o + \bar{z} \tag{8.5}$$

In the above discussion, the origin was not considered to be at $(0,0)$ coordinates on the drawing plane. But, the drawing plane has its own origin set up, and to

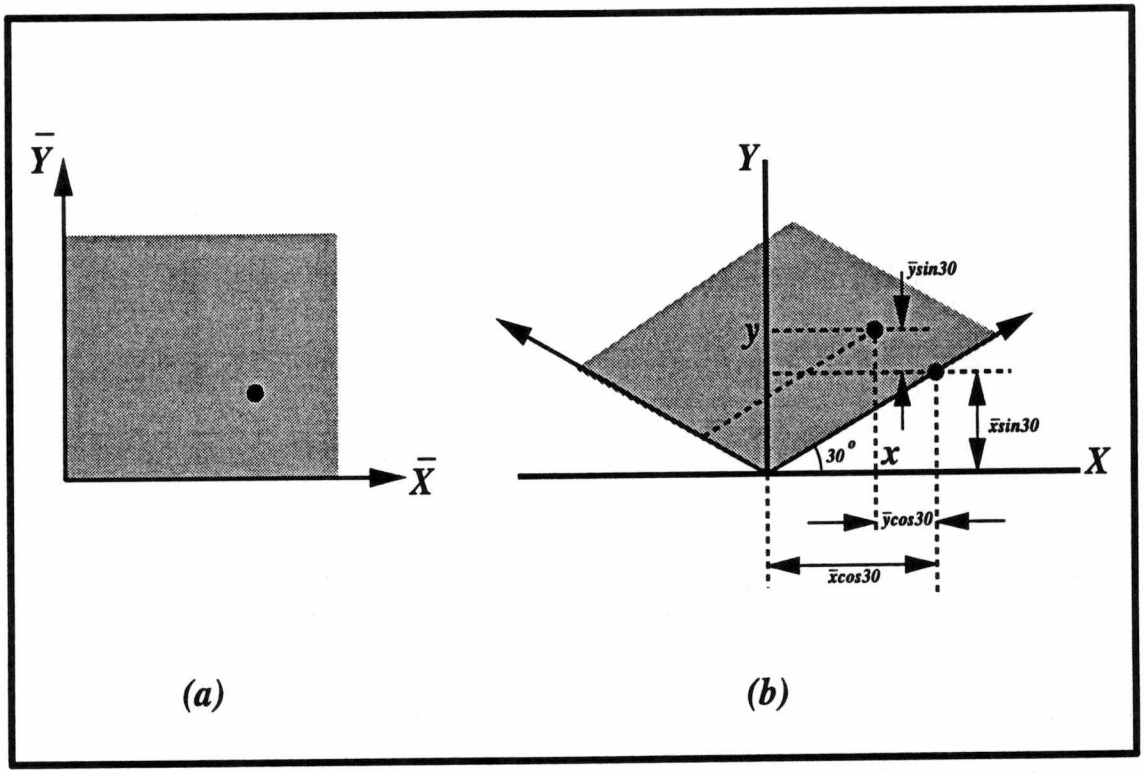


Figure 8.2: *An (x,y,0) point being mapped on the drawing plane as (x,y).*

position the plots at the desired places, a simple translation in the coordinates is required. Therefore, the equations become, with a translation of X and Y in the x and y directions respectively,

$$x = X + \bar{x}.cos30^o - \bar{y}.cos30^o \tag{8.6}$$

$$y = Y + \bar{x}.sin30^o + \bar{y}.sin30^o + \bar{z} \tag{8.7}$$

These are the equations that were implemented in the X-Windows drawing system for plotting 3-D surfaces.

8.2 The Panel for Utilities

A control panel was built using Motif widgets for the several utilities available in the software developed for this thesis. The panel provides the user with simple pushbutton and scrolling choices along with drawing erasing and noise analysis capabilities. Also, the robot R2M2 was integrated with his interface to perform automatic obstacle negotiation with the push of a button. In addition, the Square Array Organization(SAO) of the real hardware was also integrated with this interface.

The control panel is located along the right hand side of the interface *XMir*. The top of the panel includes a scrolling window which has synthetic surfaces available to the user. These synthetic surfaces include exponential, quadratic, planer, stair-step, ditch, and several other surfaces. The user can simply scroll vertically along this list of surfaces and choose one function by highlighting the name of the function. Figure 8.3 shows the control panel of *XMir* where the scrolled list can be seen at the very

top.

To the bottom of the scrolled list, there are three pushbuttons providing selections for the type of interpolation algorithm to be used. Obviously, in accordance with this thesis, these buttons are labeled *Lagrange*, *Linear*, and *Cubic Spline*. The user can click on any of these buttons in order to choose the basic interpolation type he or she wants to explore. This is also illustrated in Figure 8.3.

The next set of buttons consists of a *Draw* button, a *Clear* or erase button, a *Noise Analysis* button, and a *Quit* button(Figure 8.3). The *Draw* and the *Clear* buttons respectively draws the original and interpolated chosen function approximated with the selected algorithm basics. The *Noise* button provides the user with an option of adding zero mean white gaussian noise to the input data to the system. And the *Quit* button enables one to exit *XMir* at any time.

At the bottom of the panel, the real hardware controls are located. There are three control options available in this section of the interface. One of the buttons is for Initializing the ultrasonic range sensors regardless of whether they are located on the SAO plane or on R2M2. The second button is for turning on or activating all the sensors. When this *Activate* button is pushed, the range sensors start getting readings and send the distance measurements through the cooperation of the RPC(Remote Procedure Call) server and client. The last button of the panel is one to turn on Automatic Obstacle Negotiation capability of R2M2. The sensory data is incorporated and the control signals are sent to R2M2 when this option is utilized through the interface. When R2M2 is turned off, the normal functionality of the interface returns.

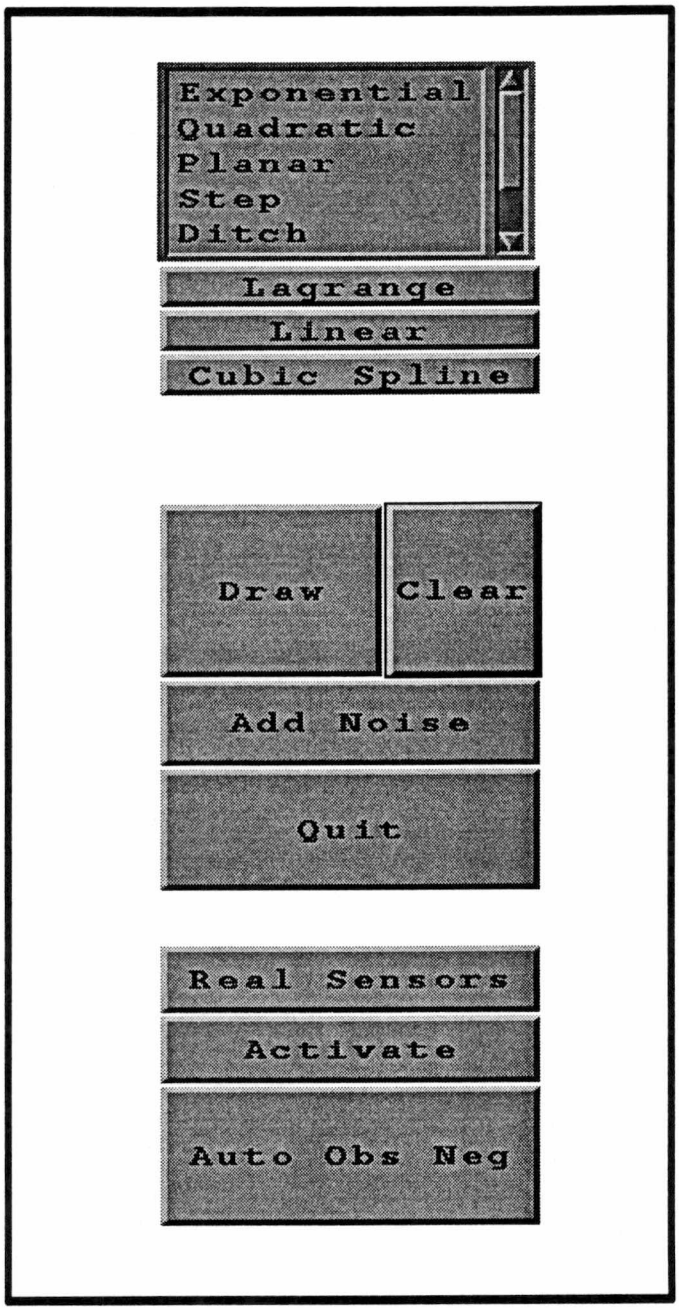


Figure 8.3: *The control panel for XMir.*

All of the control widgets were developed in Motif. Figure 8.3 is a diagram of the entire control panel which takes care of both the surface interpolation and the automatic obstacle negotiation.

8.3 Real Time graphical Controller for R2M2

The Robotic Research Mobile Manipulator(R2M2) system was originally a teleoperated system. There is a controller with R2M2 which has joysticks and switches to control the tracks and various capabilities of the arm of the robot. This real controller can be used for both the motion control and arm control of the mobile manipulator.

There was a graphical controller built for interfacing with R2M2 during earlier research activities [35] [5]. That interface was developed using the Graphics Library(GL) utilities of the Unix system. But there was a significant delay in that particular interface. Even after the graphical joysticks were activated, the robot would not immediately respond to the commands. It would follow the commands accordingly after a split second or two. However, this small delay in the updating process of the GL is enough to cause reasonable discomfort for the teleoperator.

Consequently, a graphical controller named *R2M2* was developed through X-Windows utilities(Figure 8.4). This new controller has all the motion and arm control capabilities of R2M2 as the previous one. However, this particular controller *R2M2* has faster graphics updates and thus performs real time control operations for a teleoperator. This additional graphical controller can be run in parallel with the automatic capabilities added to R2M2 in order to earn benefits such as repositioning the

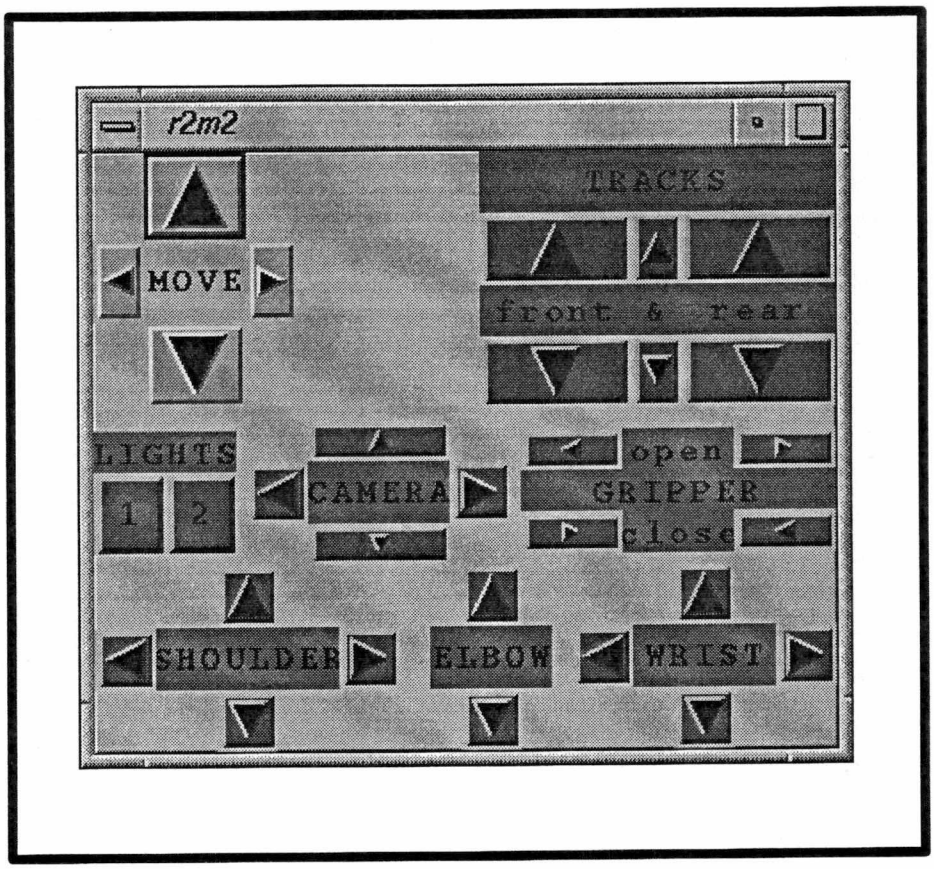


Figure 8.4: *The graphical controller "r2m2" for the robot R2M2.*

robot at any desired position in a teleoperated mode. Therefore, graphical controller provides the user with transparency between the automatic and the teleoperation modes of R2M2.

8.4 Summary

Graphical user interfaces are built in order to have the software easily available to any user. The user, with the aid of a user interface, does not have to be aware of the issues in running the options of the software package through a keyboard. Rather, he or she can view a panel on the computer screen, press buttons, use scrollbars,

arrows, and other graphical utilities with the mouse clicks, and thus run the program appropriately. The user can also view surfaces with three dimensional isometric wireframe structures and get visual representations of the results obtained. There is also an interface built for the teleoperation capabilities of R2M2. This graphical controller acts the same way as the real controller with joysticks provided with the robotic subsystem.

CHAPTER 9

Results Using Surface Interpolation

The major goal of this thesis was to approximate unknown surfaces by sampling a few points on the surface using a ranging device. The interpolation technique is developed in the scope of the project. The technique is tested extensively on known surfaces. Basically, the developed technique was used to approximate surfaces which are already defined by some function. The surfaces, both the known surface and the interpolated surface, were plotted using the user interface *XMir* to visualize the effectiveness of the approximation. Also, statistical error estimation techniques were used in order to understand the quality of the approximation techniques. In addition, the error estimation provided the opportunity to compare the basic polynomials in the larger perspective of surface interpolation.

9.1 Plotting Surfaces

The home-brewed user interface *XMir* is used to plot the surface data available to the system. The 3-D data is mapped into the 2-D viewing environment according to the isometric representation of 3-D structures. A wireframe drawing approach is taken in viewing the surfaces. The surface drawn at the left of the drawing area in the interface is the original surface, and the one at the right is the approximated

surface. This way, the user can look at the two surfaces and visually get a qualitative feedback of the strength of the interpolation technique of this thesis.

For example, Figure 9.1 is a description of the way the surfaces are graphically represented. The two surfaces at the left are the original exponential surfaces described by

$$z = e^{-4x^2-4y^2} \tag{9.1}$$

The plots at the right of the screen, however, are the ones generated by approximating the surfaces located towards their left sides using the different methods described in this thesis.

9.2 Error Estimation

When data interpolation and surface approximation is the central thought of a project, it is sometimes difficult to talk about errors in a broad sense. What is meant by "a broad sense" is the fact that the error involved in the interpolated data depends largely on how the original data is distributed with respect to the available data to the interpolator. The same data set can be available to the interpolator for two different surface structures if the sample points on both the surfaces happen to have the same depth. Therefore, by using only limited additional information, the interpolator is likely to approximate the same resulting surface for the two different original surfaces. Considering this, the error estimations with the interpolation approaches become case and application specific.

A statistical approach [37] [23] [38] [39]towards error estimation is taken to eval-

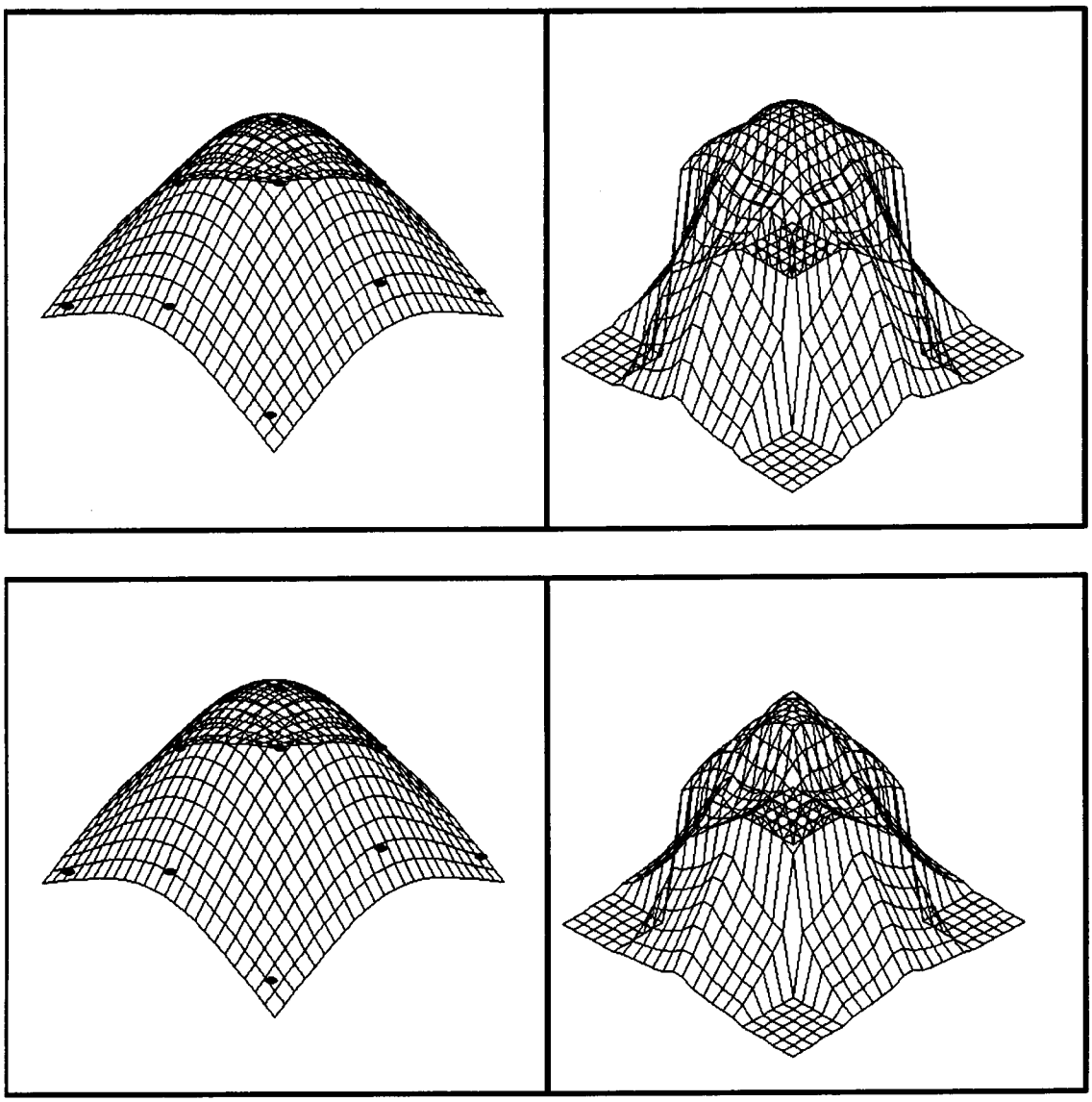


Figure 9.1: *An approximation result. The figures at left show original surfaces of the exponential type. And the figures at right show the interpolated or approximated surfaces after initially sampling only 9 points from the original surfaces as shown by the dark dots. The top-right diagram is the result of a Lagrange polynomial based interpolation, and the bottom-right surface is the Piecewise Linear based interpolation of the respective surfaces at left.*

uate the results obtained for these experiments. Therefore, results are compared between a set of points representing a set of coordinates on the original surface and a set of points representing the same set of coordinates on the approximated surface. The mean error and the standard deviation of the error are reported in this chapter.

The mean value for any statistical random variable x is defined as the *Expectation* of x and is represented by Equation 9.2.

$$E\{x\} = \frac{x_0 + x_1 + x_2 + ... + x_n}{n} \tag{9.2}$$

The variance is defined by the Expectation of x and the Expectation of x^2 in the following manner, and is expressed as σ^2,

$$\sigma^2 = E\{x^2\} - (E\{x\})^2 \tag{9.3}$$

The standard deviation is simply the square root of the variance and is expressed as σ.

In order to look at the error distribution in the generated surfaces, an array of evenly spaced points along the rows and the columns were chosen. this distribution of points is upto the user in the context of how many points need to be taken into consideration. After the depth measurements on the coordinates chosen are taken from both the original and the interpolated surfaces, the difference between those two depths represent error at that particular coordinate. Afterwards, when all of the points of interest are individually compared, Equations 9.2 and 9.3 are used to report the error measurements.

9.2.1 Testing on Synthetic Surfaces

An extensive set of qualitative and quantitative analyses was done using the interpolation techniques developed. Functions were assigned to generate a dome, an incline, a stair-step, a ditch, and various other surfaces. Approximations were made on each of these synthetic surfaces using all the three basic algorithms—Lagrange, Linear, and Spline. The tests certainly prove robustness of the algorithms as they are incorporated in this project.

Tables 9.1 through 9.5 show some error analyses of exponential, product, planar, ditch and stair-step surfaces based on the three basic polynomials—Lagrange, Linear, and Cubic Spline. By looking at the tables, it seems that Piecewise Linear approximation yields the best approximation for completely unknown surfaces. But, the performance of the Lagrange based technique is also comparable to the Piecewise Linear based technique. Of the three compared, the Cubic spline results in the most deviation from the actual surfaces. In addition, some figures illustrating those surface approximations are seen in Figures 9.1 through 9.6.

9.2.2 Testing on real surfaces

Real surfaces were also tested using the hardware made according to the Square Array structure. As it can be imagined, there are no excellent methods of measuring absolutely accurate distances of all the required coordinates of the real surfaces, a qualitative, rather than a quantitative, perspective is explored with the tests performed on real surfaces. The real surface approximations using ultrasonic range information

Table 9.1: *Error estimation for an exponential surface described by* $z = e^{-4x^2-4y^2}$.

Error Analysis for a Dome				
Approximation Algorithm	Error			
	Minimum	Maximum	Mean	Std. Dev.
Lagrange	-27.0%	48.0%	1.7%	5.7%
Piecewise Linear	-15.5%	25.0%	0.2%	2.3%
Cubic Spline	-32.7%	216.0%	3.4%	17.9%

Table 9.2: *Error estimation for an product surface described by* $z = x \times y$.

Error Analysis for a Crooked Incline				
Approximation Algorithm	Error			
	Minimum	Maximum	Mean	Std. Dev.
Lagrange	-21.9%	47.0%	2.0%	11.1%
Piecewise Linear	-17.0%	13.0%	0.3%	4.4%
Cubic Spline	-14.8%	55.0%	2.5%	10.7%

Table 9.3: *Error estimation for a plane described by* $z = const.$

Error Analysis for a Plane				
Approximation Algorithm	Error			
	Minimum	Maximum	Mean	Std. Dev.
Lagrange	0.0%	1.0%	0.1%	0.5%
Piecewise Linear	0.0%	0.0%	0.0%	0.0%
Cubic Spline	0.0%	0.0%	0.0%	0.0%

Table 9.4: *Error estimation for a ditch surface type.*

Error Analysis for a Ditch Surface				
Approximation Algorithm	Error			
	Minimum	Maximum	Mean	Std. Dev.
Lagrange	0.0%	203.0%	22.4%	63.2%
Piecewise Linear	0.0%	200.0%	22.2%	62.8%
Cubic Spline	0.0%	200.0%	11.1%	45.8%

Table 9.5: *Error estimation of stair-step surface.*

Error Analysis for a Stair-Step				
Approximation Algorithm	Error			
	MinimumM	Maximum	Mean	Std. Dev.
Lagrange	-31.2%	36.1%	0.3%	14.6%
Piecewise Linear	-25.0%	33.0%	-0.6%	13.7%
Cubic Spline	-90.0%	65.0%	-3.8%	21.6%

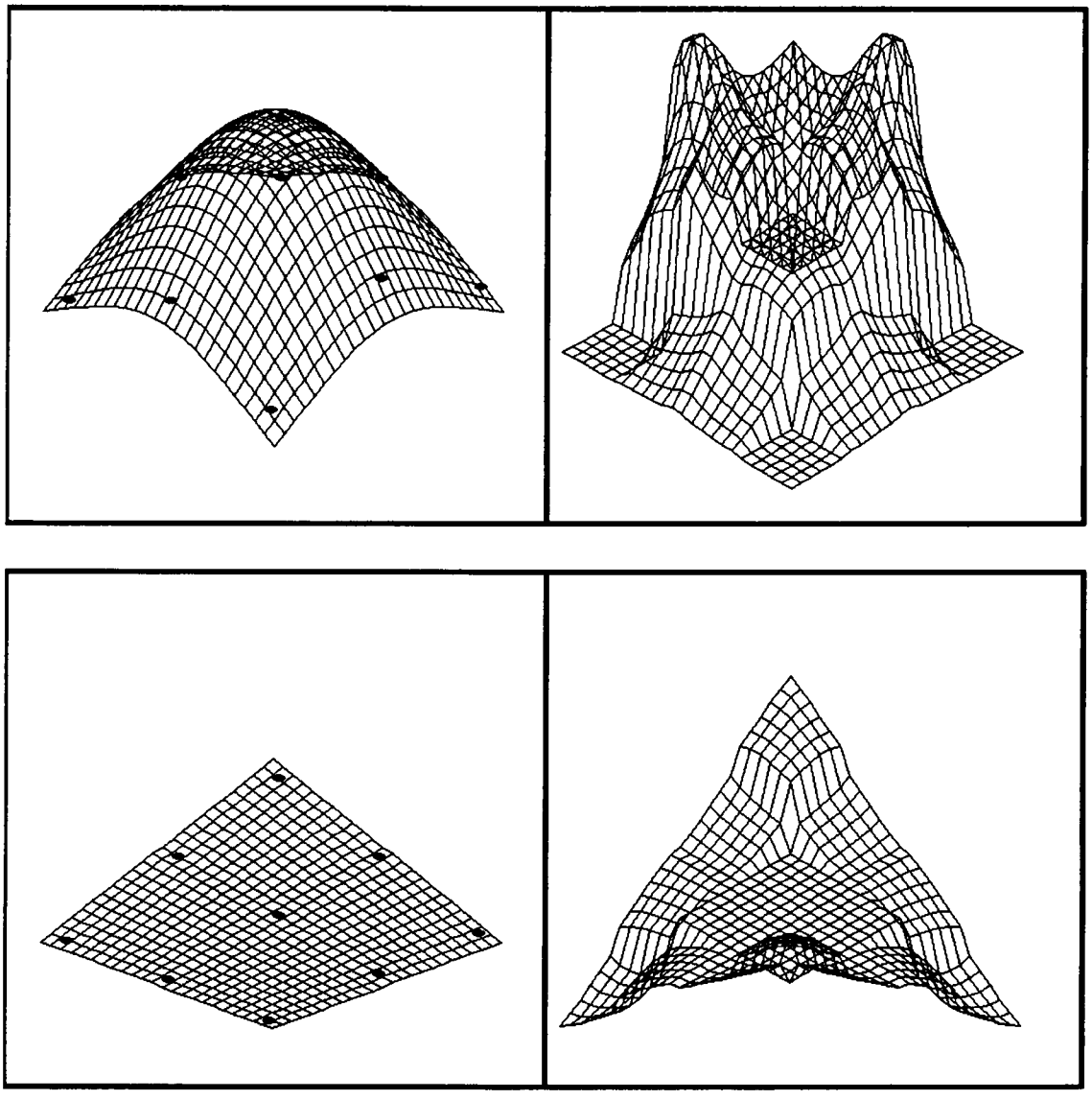


Figure 9.2: *Surface interpolation of an exponential surface using Cubic Spline based approach(on top), and approximation based on Lagrange algorithm for a product surface.*

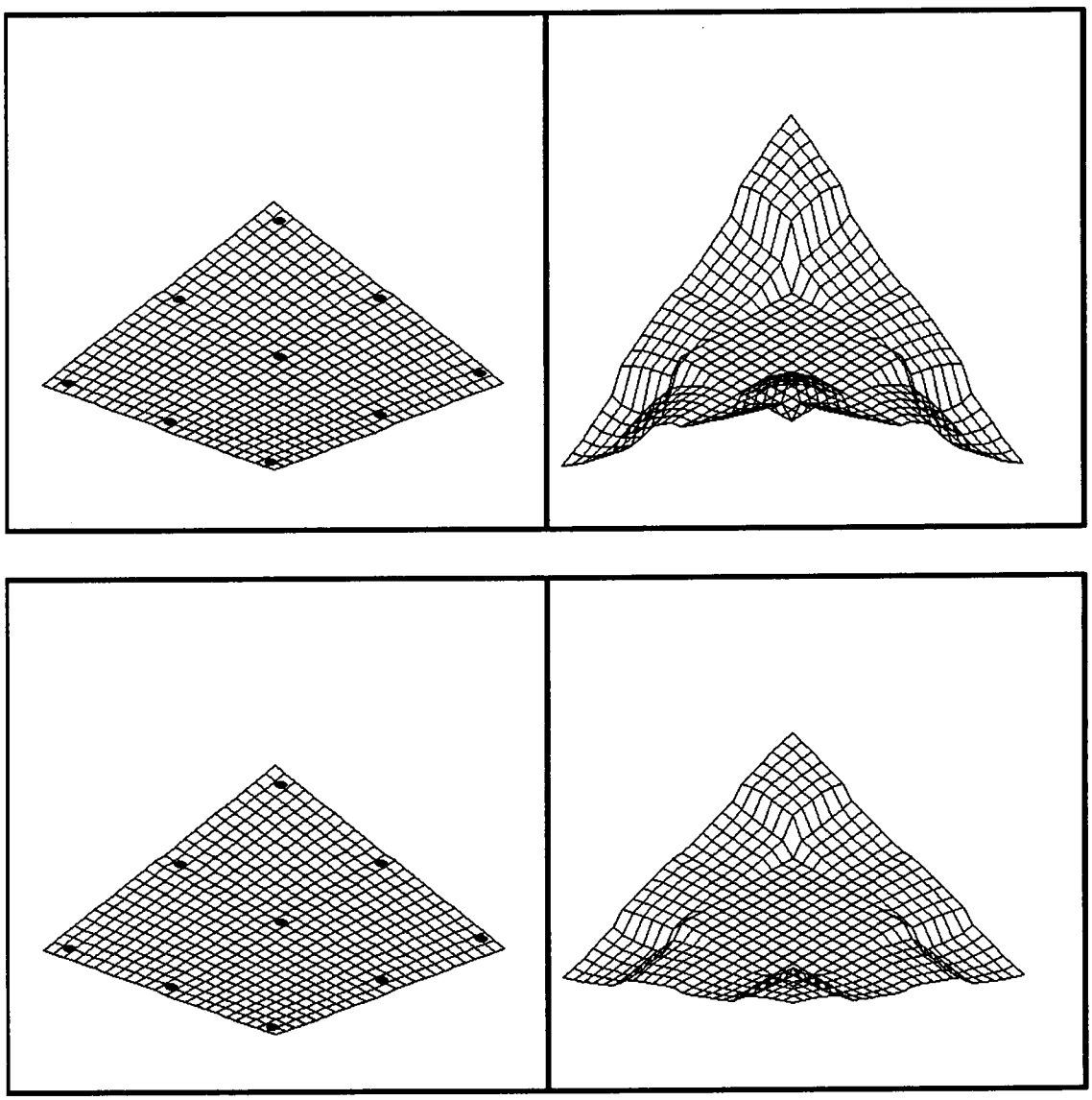


Figure 9.3: *Product surface estimation by Piecewise Linear(top) and Cubic Spline(bottom) algorithms.*

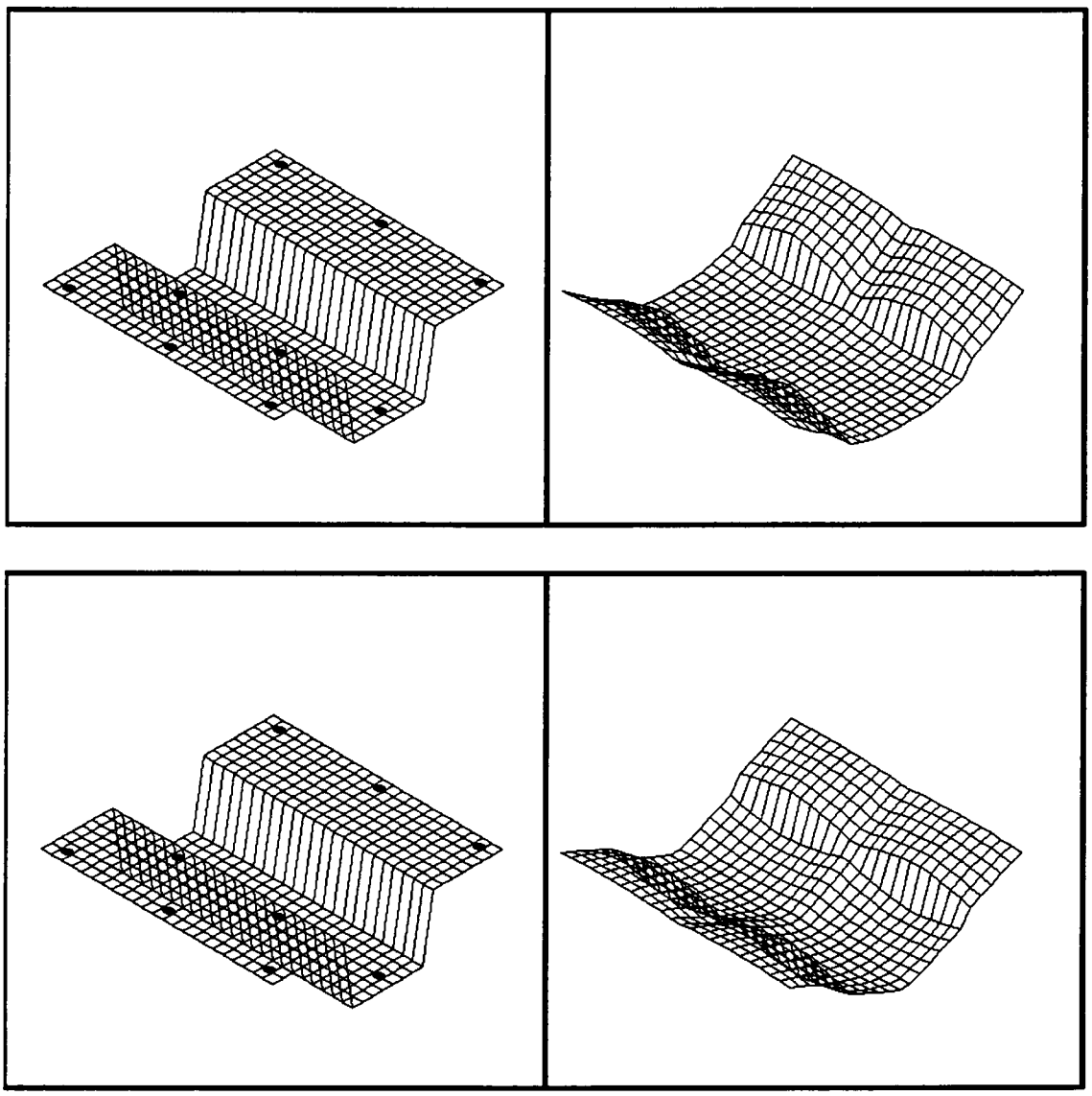


Figure 9.4: *Approximating a ditch with Lagrange based algorithm at top, and with Linear algorithm at bottom.*

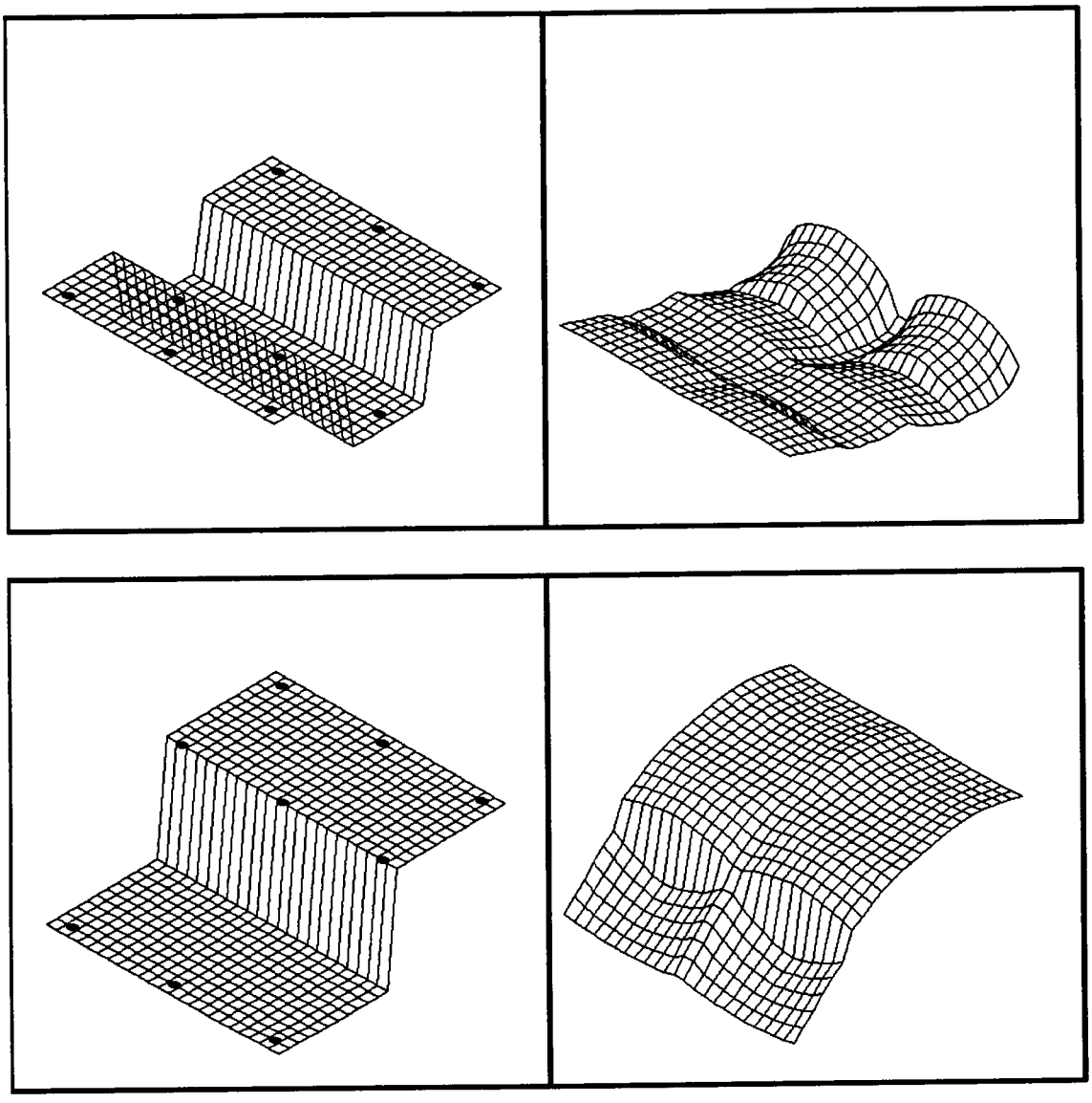


Figure 9.5: *Cubic Spline algorithm on a ditch function, and Lagrange on interpolation on a stair-step.*

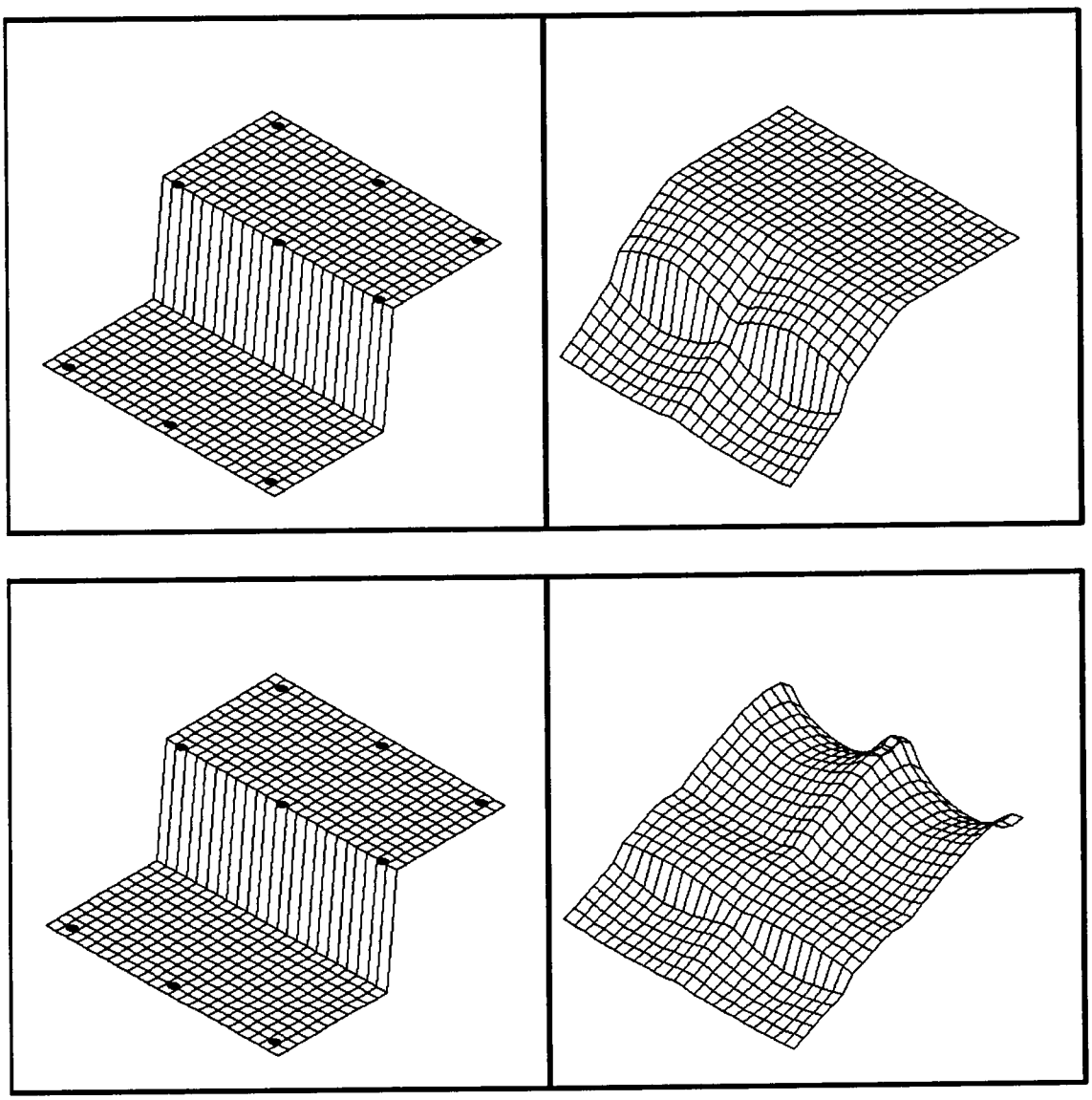


Figure 9.6: *a stair-step function approximated by Linear and Cubic Spline algorithms.*

are shown in Figures 9.7 through 9.9. It is interesting to note that for some of these interpolations, the Cubic Spline approach(especially for the surface where a board was placed in front of a wall) resulted in a better qualitative approximation. As seen in Figure 9.9, the Cubic Spline based approach nicely picked up the board in front of the wall. In the interpolated surface, the board seems to occupy about three quarters of the surface, which was indeed the case when the sensory plane was aimed at the wall and the board.

Through this practical experiment, the versatility of a general surface interpolation algorithm is once again expressed. The algorithm can thus be used for mapping the environment or for 3-D unknown surface characterization. Also, since the algorithm is computationally simple, applications in the real world are probable depending on the need of the user.

9.3 Summary

The surface interpolation algorithm is extensively tested on synthetic surfaces. All three different polynomial based approaches—the Lagrange approach, the Piecewise Linear approach, and the Cubic Spline approach—are included in the quantitative analyses. Qualitative analyses are also done by viewing the original and the interpolated surfaces in three dimensions. Zero mean white gaussian noise of different standard deviations are added to the input data in order to test the robustness and effectiveness of the algorithm. Pictures of real surfaces and the 3-D representations of their corresponding interpolated surfaces are also included here.

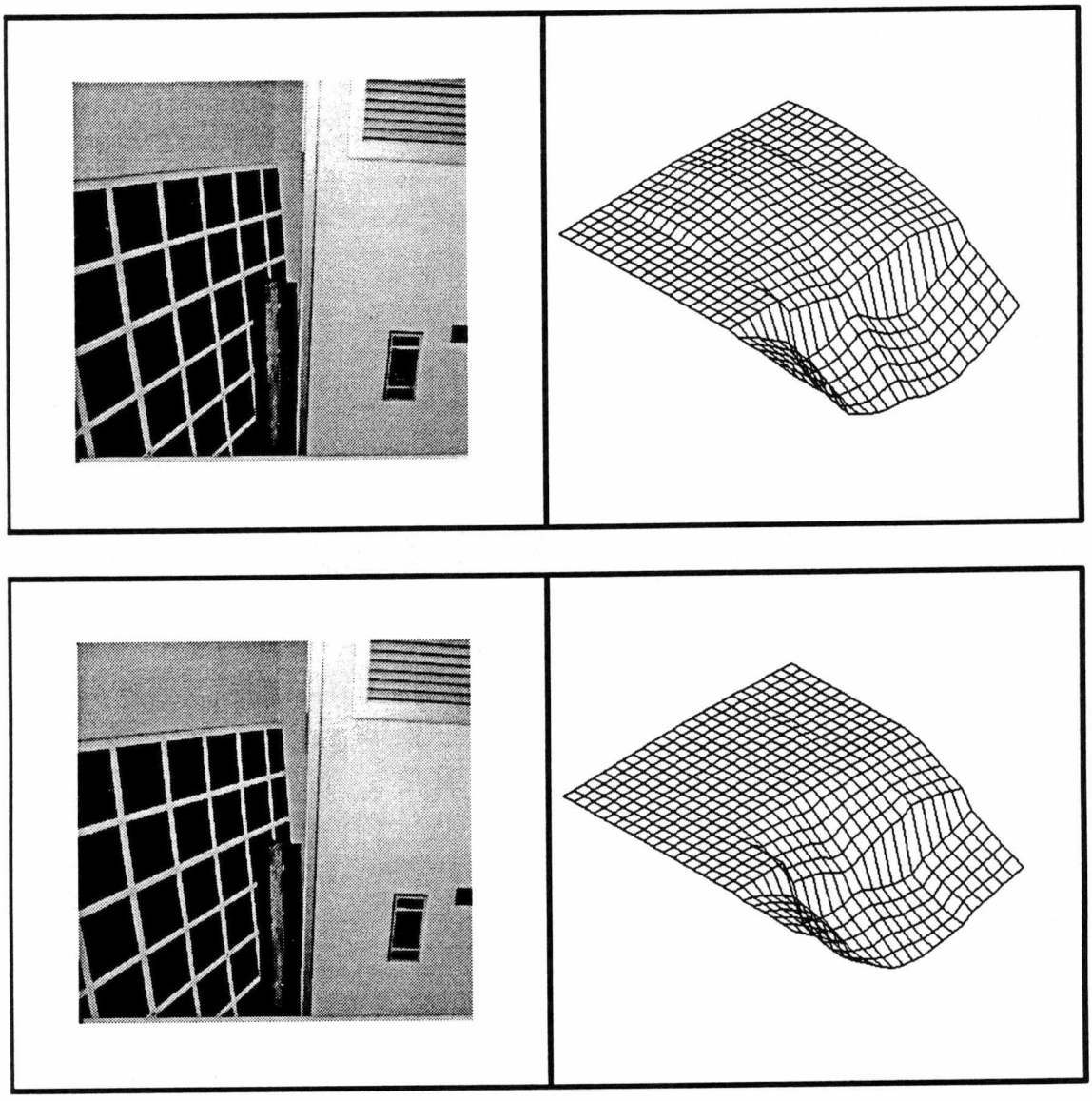


Figure 9.7: *The approximation of an edge using real range data using the Lagrange and the linear interpolations.*

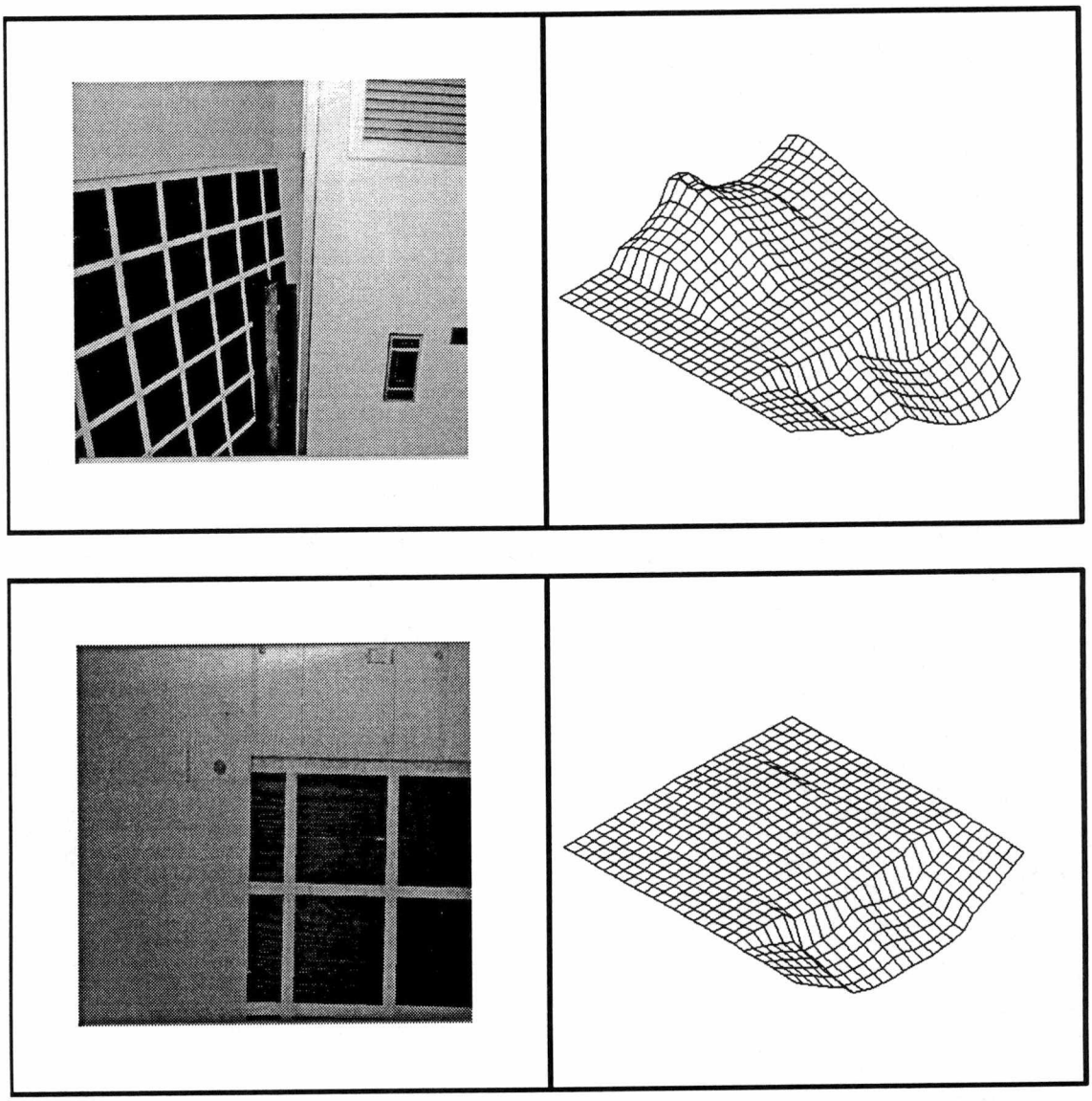


Figure 9.8: *The approximation of an edge using the Cubic Spline approach, and Lagrange based interpolation of a board partially in the view in front of a wall using real range data.*

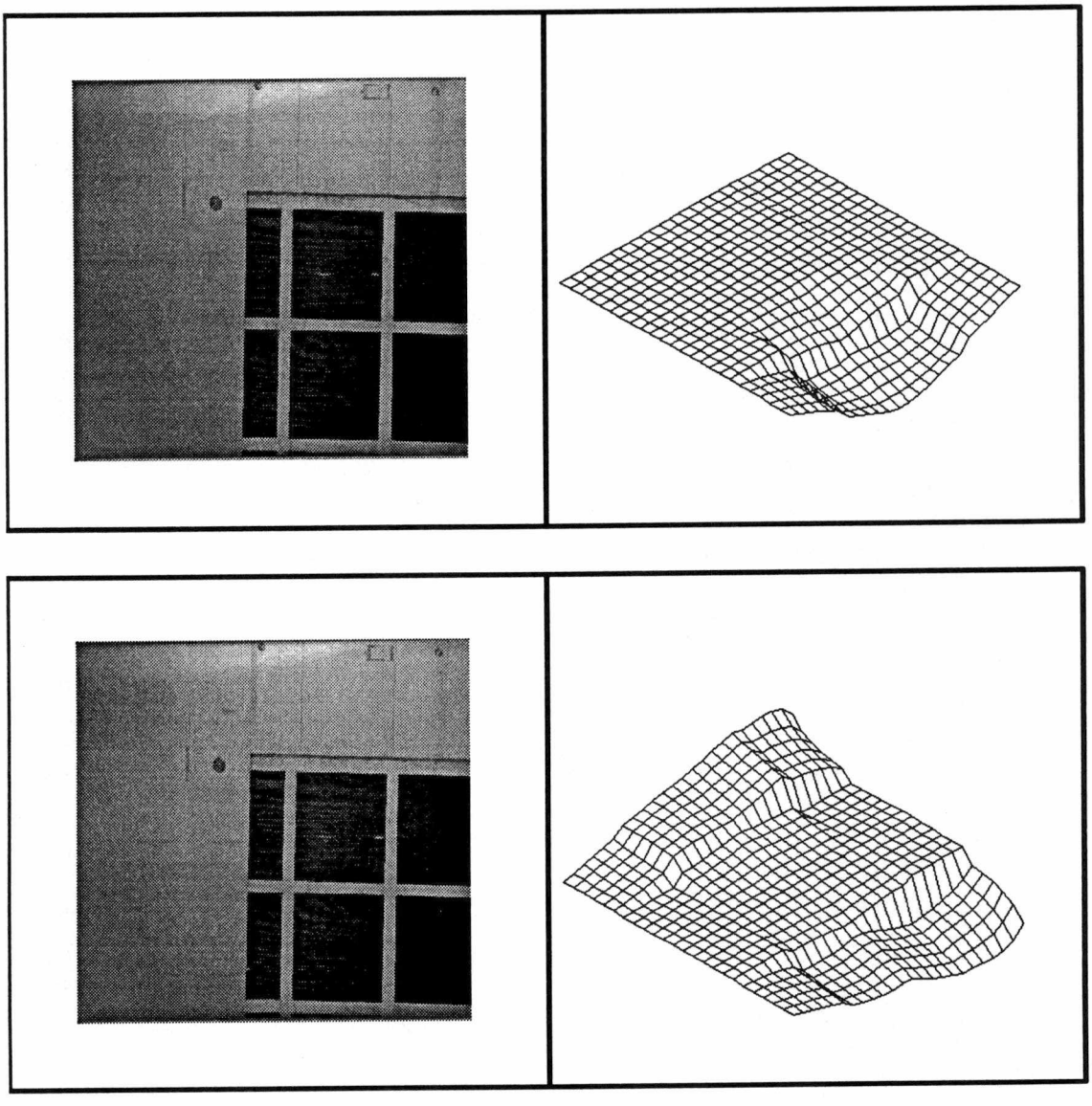


Figure 9.9: *A board placed in front of a wall. Real data approximation using the Linear and the Cubic Spline approach.*

CHAPTER 10

Concluding Remarks and Future Research

The surface interpolation algorithms developed in this project are successful in approximating surfaces with mean errors of less than 22% and with typical standard deviation of less than 15%. In some cases, however, as in the case of a ditch or a beam surface, standard deviation went up to about 50% to 60%. These large standard deviations were there because there were abrupt changes in the surface contour more than once, and the surface interpolation technique fit smooth surfaces through them. The plots presented in this thesis clearly show that the interpolated surfaces can easily be identified with their respective original surfaces. The algorithms, as discussed in earlier chapters, are simple and less demanding in terms of computational necessities, having all the calculations completed within the next data update instance, instances which are only 100 milliseconds apart. Therefore, the goal for achieving simplicity in surface approximation was reached. In addition, the interpolation techniques are robust, since different kinds of surfaces with a variety of contours were interpolated with typical error figures mentioned. The interpolation techniques were practical because R2M2 was able to use the algorithm to center itself on obstacles successfully. Polynomials proved to be good interpolation methods since R2M2 was able to negotiate obstacles using polynomials.

The user interfaces developed are useful in teleoperating R2M2. Also, surface plots

can be achieved and viewed using the interface. One of the possibilities mentioned at an earlier point in the thesis was that a teleoperator can operate the mobile robot, but when he or she runs in front of some obstacle that the robot needs to negotiate, the two dimensional camera view might not be enough for the operator to navigate any longer. That means that the two dimensional camera view does not provide the operator with depth information in order for the operator to make decisions about the track adjustments. The operator can only anticipate the structure of the obstacles and thus risks the possibility of an accident to happen. A simple push of a button can put the robot in automatic obstacle negotiation mode. That way, the robot can navigate on the uneven surface and when it gets back on level ground, the operator can take control of the robot back. The user interface provides this capability also. Therefore, a practical situation confronted by any user can be dealt with the results of this study. Another goal of mobile robots like R2M2 is achieved in this manner.

10.1 Achievements

There were a few distinct accomplishments of this thesis project. These achievements can be itemized as follows:

- Development of surface interpolation algorithms using polynomials, with rigorous statistical analysis,
- Evaluation and implementation of polynomial based reactive obstacle negotiation of a mobile robot, and

- Development of graphical user interfaces for teleoperation of the mobile robot R2M2 and for surface analysis and viewing in three dimensions.

It needs to be emphasized that the surface interpolation algorithms developed are of a general technique that can be used for any range data input. Also, the application of this algorithm need not be limited only in the area of robot navigation. The algorithms can be utilized in environment mapping, recognition, surface characterization, and various such areas associated with 3-D data analysis.

10.2 Future Research Plans

The findings of this study can be used to explore various research interests in the future. The hardware development of the Square Array Organization(SAO) can be mounted on the mobile manipulator's arm and thus let the robot survey around, incorporate the arm's coordinate information with the system, and finally find the best path for the robot to navigate. Mobile robots can thus also be used for mapping the environment. Areas involving range image processing, where a lot more than 9 data points are usually available, can be explored with polynomial fitting methods to approximate the missing information. These are all future research issues. The goal of mobile robotics at present time is certainly not to achieve complete automation, but automating some tasks that are difficult for a remote teleoperator can undoubtedly help mobile robotics reach a higher level of performance.

10.3 Summary

The use of this surface interpolation techniques is dependent on the user's need. Certainly, results obtained through this method developed are not the most accurate compared to all the other surface fitting algorithms. But the basic idea behind this algorithm was to work with severely insufficient data. When complicated interpolation cannot be performed due to the insufficiency of samples, or when the results of interpolation need to be achieved within milliseconds, this particular algorithm is reasonable to test. The interpolation algorithms derived in this thesis can also be expanded by using a different kind of polynomial than any one used here if the user chooses to do so. Therefore, flexibility is another advantage of this interpolation technique. The application in mobile robotics was the area of interest for this thesis, but any area requiring range data analysis can be explored through the surface interpolation algorithm derived here.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] P. Dierckx, V. Leemput, and T. Vermeire, "Algorithms for surface fitting using Powell-Sabin splines," *IMA Journal of Numerical Analysis*, vol. 12, pp. 271–299, 1992.

[2] P. Boulanger, "Automatic extraction of complex surface models from range images using a trimmed-rational bézier surface," *International Society of Optical Engineering*, vol. 2059, pp. 68–78, 1993.

[3] S. Lavallée, R. Szeliski, and L. Brunie, "Matching 3-d smooth surfaces with their 2-d projections using 3-d distance maps," *Geometric Methods in Computer Vision*, vol. 1570, pp. 322–336, 1991.

[4] Intelligent Robots and Systems, *Ultrasonic sensor data fusion for environmental recognition*, vol. 1, July 1993.

[5] C. Chen and M. M. Trivedi, "Savic: A simulation visualization and interactive control environment for mobile robotics," *Series in Machine Perception and Artificial Intelligence: Active Robot Vision*, vol. 6, pp. 133–144, 1993.

[6] M. A. Wolfensberger and D. Wright, "Synthesis of reflexive algorithms with intelligence for effective robot path planning in unknown environments," *International Society for Optical Engineering: Mobile Robotics VIII*, vol. 2058, pp. 70–81, 1993.

[7] R. C. Luo and M. G. Kay, "Multisensor integration and fusion in intelligent systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 5, pp. 901–931, 1989.

[8] Intelligent Robots and Systems, *Reactive locomotion control of articulated-tracked mobile robots for obstacle negotiation*, vol. 2, July 1993.

[9] P. Gendreu, M. Fink, and D. Royer, "Visualization of surface displacement and transmitted field of piezocomposite transducers by optical probing," *IEEE Ultrasonic Symposium*, vol. 1, pp. 541–546, 1993.

[10] J. J. Leonard and H. F. Durrant-Whyte, *Derectet Sonar Sensing for Mobile Robot Navigation*. Klumer Academic Press, 1992.

[11] J. A. Janét, R. C. Luo, C. M. Aras, and M. G. Kay, "sonar windows and geometrically represented objects for mobile robot self-referencing," *Intelligent Robots and Systems*, vol. 2, pp. 1324–1331, July 1993.

[12] Y. Chen and G. Medioni, "Surface description of complex objects from multiple range images," *Computer vision and Pattern Recognition*, June 1994.

[13] *Texas Instruments Data Book.* Post Office Box 225012, Dallas, TX 75265, 1983.

[14] A. Gaskell and P. Probert, "Sensor models and a framework for sensor mangement," *International Society of Optical Engineering*, vol. 2059, pp. 2–25, 1993.

[15] J. F. Steffensen, *Interpolation.* Chelsea Publishing Company, New York, NY, 1950.

[16] I. C. Demetriou, "Least squares smoothing of univariate data to achieve piecewise monotonicity," *International Mathematical Association Journal of Numerical Analysis*, vol. 12, pp. 411–432, 1991.

[17] C. A. Hall and T. A. Porsching, "Approximation methods in computer numerically controlled fabrication of optical surfaces, part 1: Finite dimensional material removal profile spaces," *IMA Journal of Numerical Analysi*, vol. 11, pp. 67–84, 1991.

[18] C. A. Hall and T. A. Porsching, "Approximation methods in computer numerically controlled fabrication of optical surfaces, part 2: modifications," *IMA Journal of Numerical Analysis*, vol. 12, pp. 259–269, 1992.

[19] E. Bitter, S. Lavallée, and R. Szeliski, "A method for registering overlapping range images of arbitrary shaped surfaces for 3-d object reconstruction," *The International Society for Optical Engineering*, vol. 2059, pp. 384–395, September 1993.

[20] Intelligent Robots and Systems, *3D object representation using spherical harmonic functions*, vol. 3, July 1993.

[21] C. V. Stewart, "A new robust operator for computer vision: application to range data," *Computer Vision and Pattern Recognition*, pp. 167–173, June 1994.

[22] H. Samet, *The Design and Analysis of Spatial Data Structures.* Reading, Massachusetts: Addison-Wesley, 1989.

[23] M. Tanemura, "Likelihood estimation of directional interaction," *First US/Japan Conference on the Frontiers of Statistical Modeling: An Information Approacch*, vol. 3, pp. 293–313, 1992.

[24] B. Mulansky, "Chebyshev approximation by spline functions with free knots," *IMA Journal of Numerical Analysis*, vol. 12, pp. 95–105, 1992.

[25] G. I. Murchuk, *Methods of Numerical Mathematics.* Springer-Verlag New York Inc., 2 ed., 1982.

[26] R. L. Burden and J. D. Faires, *Numerical Analysis.* PWS Publishing Company, Boston, MA, 5 ed., 1993.

[27] E. J. Barbeau, *Polynomials.* Springer-Verlag New York Inc, 1989.

[28] P. D. Loach and A. J. Wathen, "On the best least squares approximation of continuous functions using splines with free knots," *IMA Journal of Numerical Analysi*, vol. 11, pp. 393–409, 1991.

[29] S. M. Richie, C. A. Fulton, and B. P. Abbott, "The inplementation of the chebyshev approximation for the synthesis of saw filters," *IEEE Ultrasonic Symposium*, vol. 1, pp. 53–58, 1993.

[30] M. Soucy and D. Laurendeau, "Multi-resolution surface modeling from multiple range views," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 348–353, 1992.

[31] M. Manela, N. Thornhill, and J. A. Campbell, "Fitting spline functions to noisy data using a genetic algorithm," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 549–553, 1993.

[32] M. Hebert, "3-d landmark recognition from range images," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 360–365, 1992.

[33] D. P. Miller, R. S. Desai, and E. Gat, "Reactive navigation through rough terrain: Experimental results," in *International Conference on Artificial Intelligence*, pp. 823–828, 1992.

[34] R. P. Bonasso, H. J. Antonisse, and M. G. Slack, "A reactive robot system for find and fetch taskes in an outdoor environment," in *International Conference on Artificial Intelligence*, pp. 801–808, 1992.

[35] C.Chen, M. M. Trivedi, and M. Azam, "A novel graphical environment for virtual and real world operations of tracked mobile manipulators," *International Society of Optical Engineering*, vol. 2059, pp. 154–164, September 1993.

[36] Intelligent Robots and Systems, *A man machine interface for a mobile robot*, vol. 1, 1993.

[37] T. Lang, "New structure criteria in group method of data handling," *International Journal of Mathematics in Science and Technology*, pp. 456–462, 1990.

[38] M. Omladič, "Average quadrature formula for gausse type," *International Journal of Mathematics for Science and Technology*, vol. 7, pp. 189–199, 1992.

[39] J. S. Bendat and A. G. Piersol, *Random Data Analysis and Measurement Procedures*. John Wiley and Sons, 1986.

VITA

Mir Shariful Azam was born in the city of Chittagong, Bangladesh on the 8th day of June, 1971. He received his Secondary School Certificate(S.S.C.) in 1986 from the Government Laboratory High School in Dhaka, Bangladesh and his Higher Secondary Certificate(H.S.C.) in 1988 from the Dhaka College in Dhaka, Bangladesh. He came to the United States in Fall of 1989 to start his Bachelor's degree program at Belmont University in Nashville, TN. He received his Bachelor's degree in Electrical Engineering from the University of Tennessee at Knoxville, TN in December of 1992. He has been a Graduate Research Assistant at the Computer Vision and Robotics Research Laboratory in the Electrical and Computer Engineering Department at UTK since January of 1993. His interests include problems involving mobile robotics, intelligent and reactive systems, ultrasonic sound based sensors, three dimensional structures and surfaces, graphical user interfaces, and other issues related to robotics and intelligent systems. He is also a member of the Tau Beta Pi, the Eta Kappa Nu, the IEEE and the SPIE.