INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

**THE DISPLAY, COMPARISON, AND REGISTRATION OF
SURFACES IN RANGE DATA FORMAT**

by

JENNY Y.S. LEUNG, B.CS.

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of**

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

November, 2001

© Copyright

2001, Jenny Yuen Sze Leung

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**385 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**385, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-72079-9

Canada

The undersigned hereby recommends to the
Faculty of Graduate Studies and Research
acceptance of the thesis,

**THE DISPLAY, COMPARISON, AND REGISTRATION OF
SURFACES IN RANGE DATA FORMAT**

Submitted by

Jenny Y. S. Leung, B.C. S.

Dr. Frank Dehne

(Director, School of Computer Science)

Dr. B. John Oommen

(Thesis Supervisor)

Carleton University

November, 2001

Abstract

Geometric Modeling involves the pictorial synthesis of an object from an associated computer-based model. It involves defining a geometric model and developing design methods for this model. Modeling the surface is an important aspect of geometric modeling, where an object is represented by its outer skin (boundary) as discrete elements, and these elements are not associated with the architectural structure of the object itself.

Surface registration is the process of aligning surfaces represented in the form of a digital image, and has applications in image processing and analysis. Without objective criteria for measuring the likeness between surfaces, the reliability of surface registration is difficult to assess. This thesis defines different strategies for measuring the likeness between surfaces, so that the comparison and registration of the surfaces can be achieved. The measure of likeness between two digital surfaces is quantified by a *Surface Similarity Measure* (SSM) and the data representation and display methods for surfaces will also be discussed.

All the problems discussed above are very complex if the image is represented in a range-data format, as opposed to the gray-level format. This thesis contains a study of the associated functions, and details a complete functional prototype, the Manitoba Adaptor Registration System (MARS) that supports all these functionalities.

Acknowledgements

I would like to thank my supervisor from Carleton University, Dr. John Oommen, for his valuable guidance during my thesis work. It has been a pleasure to work with him. His direction, ideas and enthusiasm are very much appreciated. Also, his support and advice have made the completion of this thesis possible. I would also like to thank Dr. Oommen for the extra time and effort he put in to help me finish my thesis on time.

I am grateful to my former supervisor from the University of Manitoba, Dr. Derek Meek, for working with me on giving me the ideas and guidelines on my thesis work. He worked with me on fine tuning my thesis work with great patience. I sincerely appreciate his invaluable input and guidance.

As well, I would like to express my appreciation to my loving husband Philippe, who has given me so much support and advice concerning my thesis. I thank my Mom and Dad who encouraged me in times of disappointments, and for their support when I decided to pursue my Masters degree. Finally, I would like to thank my two sisters, Alice and Doris, and my brother Chung, who not only gave me support, but encouragement and understanding. I would like to say a special word of thanks to Chung, who accompanied me while I worked late in the lab, and made the work more enjoyable. This thesis is dedicated to all of them.

I am also very grateful for the financial support provided by the Faculty of Computer Science at the University of Manitoba, and thank the staff of the Computer

Science faculties of both Universities who helped me with all the various administrative requirements.

Table of Contents

Abstract

Acknowledgements

List of Figures

Chapter 1 - Introduction

1.0 Background.....	1
1.1 Purpose of Thesis.....	3
1.2 Contributions of the Thesis.....	3
1.3 Thesis Outline.....	5

Chapter 2 - Surface Representation

2.0 Introduction	7
2.1 Representing Surface Data	8
2.1.1 Mathematical Modeling Function.....	10
2.1.2 Unit Normal and Shape Descriptors List Array (UNSDLA)	12
2.1.3 Image Segmentation	13
2.1.4 Range Data	13
2.2 Range Data as a Surface Descriptor	13
2.2.1 Advantages of Range Data as a Surface Descriptor.....	14
2.2.2 Disadvantages of Range Data as a Surface Descriptor	15
2.3 Converting Scattered Data to Range Data	16
2.3.1 Spatial Resolution and Interpolation.....	18
2.3.2 Scattered Data and Interpolation Methods.....	20
2.4 Conclusion	24

Chapter 3 - Displaying a Range Data Surface

3.0 Introduction	26
------------------------	----

3.1	Gray Level Display	27
3.1.1	Mathematical Definition	28
3.1.2	Enhancement on Gray Level	31
3.1.3	Advantages and Disadvantages	31
3.2	Contour Drawing	32
3.2.1	Mathematical Definition of Contour Lines	32
3.2.2	Finding Contour Lines	33
3.2.3	Finding Contour Regions	37
3.2.4	Advantages and Disadvantages of Contour Display Schemes	38
3.3	Lighting from Different Directions	40
3.3.1	Approximating Normal at a Point	41
3.3.2	Determining the Shading at a Point	42
3.3.3	Advantages and Disadvantages of Shading	45
3.4	Edge Presentations	46
3.4.1	Advantages and Disadvantages of Edge Display	48
3.5	Conclusion	49

Chapter 4 - Comparing Surfaces

4.0	Introduction	50
4.1	Featureless Vs Feature Comparison	52
4.1.1	Advantages of Featureless and Feature Comparisons	53
4.2	Corresponding Points	53
4.2.1	Vertical Correspondence	55
4.2.2	Perpendicular Correspondence	57
4.2.3	Equal Angles Correspondence	62
4.3	Similarity Measure.	64
4.4	Conclusion	67

Chapter 5 - Interactive Registration

5.0	Introduction	69
5.1	Registration Algorithm	71
5.2	Affine Transformation on Range Data	74
5.2.1	Scaling	75
5.2.2	Translation	79

5.2.3 Rotation	82
5.3 Enhancing the Transformation	83
5.3.1 Reversed Pixel Mapping	83
5.3.2 Bilinear Interpolation.....	85
5.4 Conclusion.....	88

Chapter 6 - Manitoba Adaptor Registration System (MARS)

6.0 Introduction	89
6.1 Functions Implemented.....	89
6.2 Display Menu.....	93
6.2.1 GetGrayLevel.....	93
6.2.2 GetContour	95
6.2.3 GetBiLevelContourRegion.....	96
6.2.4 GetShading	98
6.2.5 GetEdge	99
6.3 Transformation Menu	100
6.3.1 SurfaceTranslation.....	101
6.3.2 SurfaceScaling	102
6.3.3 SurfaceRotation	104
6.4 Similarity Measure Menu	105
6.4.1 CompareRangeData	106
6.5 MARS Interface Functions	108
6.5.1 GetFile	109
6.5.2 FileParseToRangeData	110
6.5.3 DisplayImage.....	111
6.6 Conclusion.....	112

Chapter 7 - Summary and Future Research

7.0 Summary and Conclusions	113
-----------------------------------	-----

References	115
-------------------------	------------

List of Figures

Figure 2.1 A parallelepiped shaped lattice representing the domain of data	9
Figure 2.2 Digital Surface Data	10
Figure 2.3 Different sampling intervals are expected in the y-direction	16
Figure 2.4 Image of dice obtained with different resolutions	19
Figure 2.5 Modified Quadratic Shepards' (MQS).....	23
Figure 2.6 Diagram showing MQS with points A, B, C, D	24
Figure 3.1 Dice, tooth and horse data at different numbers of gray levels	30
Figure 3.2 Different contour levels shown by contour lines, each separated by Δh	33
Figure 3.3 A grid square containing 4 data points	34
Figure 3.4 Steps showing how a triangle is chosen and how to get the contour	35
Figure 3.5 Contour line drawing of tooth data.....	36
Figure 3.6 Four contour regions resulted from contour lines in Figure 3.2.....	37
Figure 3.7 Contour lines with same topology can represent 2 entirely different surfaces	39
Figure 3.8 The two contours represent the same picture but have different topologies	39
Figure 3.9 Reflection of light at point P.....	40
Figure 3.10 Normal at C is approximated by vector AB and A'B'	41
Figure 3.11 Normal at C is approximated by vector EF and E'F'	42
Figure 3.12 Maximum and minimum intensities of light L at different angles	42
Figure 3.13 Pictures showing effect of light source from different directions	43
Figure 3.14 Effect of light source from same direction but different angles	44
Figure 3.15 Effect of light with negative angle.....	44
Figure 3.16 A simple step edge.....	46
Figure 3.17 The discrete version of Laplacian operator.....	47
Figure 3.18 Results by applying the Laplacian Operator	47
Figure 3.19 Some examples of edges [PM90]	48
Figure 4.1 Image of Horse Head at different angles.....	51
Figure 4.2 Background identified by region coherence	52
Figure 4.3 Corresponding points D_1 and D_2 taken from S_1 and S_2 respectively.....	54
Figure 4.4 Corresponding points C, P of two curves taken from two surfaces.....	55
Figure 4.5 Corresponding points C, P from two curve segments.....	55

Figure 4.6 Vertical Correspondence	56
Figure 4.7 Vertical correspondence; C, D correspond	56
Figure 4.8 Perpendicular Correspondence	57
Figure 4.9 Search range of P given radius r	58
Figure 4.10 Search space of P given radius r.....	58
Figure 4.11 P, D correspond, D is closest point in C_2 to point P in C_1	59
Figure 4.12 A simple case of more than one corresponding point	60
Figure 4.13 Another case of more than one corresponding point	60
Figure 4.14 Perpendicular intersection from P to C_2 does not fall	61
Figure 4.15 Different results by choosing points $C_1 \rightarrow C_2$ and $C_2 \rightarrow C_1$	62
Figure 4.16 Equal Angles Correspondence	62
Figure 4.17 A point with two equal angle correspondences	63
Figure 4.18 Multiple corresponding points from two curves.....	64
Figure 4.19 Length between corresponding pair D_1 and D_2	65
Figure 4.20 Surface Based Similarity Measure.....	66
Figure 5.1 Relationship of similarity measure and interactive registration	71
Figure 5.2 Magnification of Dice with Scaling.....	76
Figure 5.3 Scaling of image	77
Figure 5.4 Image translation with $T_x = 10$, $T_y = -50$	80
Figure 5.5 Wraparound effect on a large format image.....	81
Figure 5.6 Patch in Bilinear Interpolation	86
Figure 5.7 Rotation of Dice Image with pixel location.....	87
Figure 5.8 Rotation of Dice Image.....	88
Figure 6.1 A Schematic of MARS' main menu	92
Figure 6.2 A Schematic of the Display sub-menus in MARS	93
Figure 6.3 The schematic of the transformation sub-menu of MARS	101
Figure 6.4 The similarity sub-menu for MARS	105
Figure 6.5 The Main "File"- menu for MARS.....	108

Chapter 1

Introduction

1.0 Background

Computer graphics involves the pictorial synthesis of an object from the associated computer-based model [FD90]. *Geometric Modeling* is the basis for computer graphics that involves defining a geometric model and developing design methods. *Surface Representation* is an important aspect of geometric modeling, where an object is represented by its outer skin or boundary. A surface is composed of discrete elements that are not associated with the architectural structure of the object.

Image Processing deals with surfaces in the form of a digital image; in image processing there is no reference to the structural architecture of the object. Algorithms in this field perform analysis of an image, unlike in the field of computer graphics [FD90], where the goal is to visually enhance or statistically evaluate some aspect of an image not readily apparent in its original form. For example, computer graphics involves reconstructing models of 2D or 3D objects from their pictures. A *Digitized surface* can be obtained by a process called *Digitization* which converts a continuous picture into a discrete form [PA82]. Digitized surfaces can be used in the precise implementation of image processing techniques, improving the system's flexibility and usefulness.

Image processing techniques can be grouped into three general areas, depending upon the intended use of the processed image:

1. **Quantitative Restoration:** This involves the description of the picture by a range of numbers so that each pixel on the picture is assigned a number within this range.
2. **Image Enhancement:** This involves the modifications of features of a picture in order to alter its impact on the viewer. It deals with improving the image quality by eliminating fuzziness or enhancing contrast. In general, the numerical representation of the picture will be changed after the enhancement.
3. **Information Extraction:** This phase involves the conversion of pictorial data into information required by the analyst, perhaps with a reduction in the quantity of the information. Some common sub-areas in image processing for information extraction are: (a) Pattern Detection and Recognition which detect and clarify standard patterns and/or find distortions from standard pattern, (b) Scene Analysis, and (c) Computer Vision which recognize and reconstruct 3D models of a scene from several 2D images.

Digital surfaces can also be used for data representation in scientific studies, which permit further analysis using various image processing techniques. Digital surface representation is therefore a very important and relevant field of study applicable to both image processing and computer graphics.

1.1 Purpose of Thesis

The purpose of this thesis is to first of all, show different methods for displaying surfaces. The thesis then defines different strategies for measuring the likeness of surfaces as applicable in surface comparison. We also discuss the transformation and registration of two surfaces. The main focus of this thesis is on surface comparison.

Without objective criteria for measuring the likeness between two surfaces, the reliability of surface comparison is difficult to assess. The measure of likeness between two digital surfaces is referred to as the *Surface Similarity Measure (SSM)*. Data representation, display methods, comparison techniques and the uses of the SSM in surface registration will be discussed in the following chapters.

1.2 Contributions of The Thesis

The aim of this thesis is to show how we can perform tasks that involve the handling of raw image data, thus achieving data analysis and manipulation even though the images are maintained in non-traditional formats. Indeed, the fundamental contribution of this thesis is to demonstrate that most of the functions associated with images maintained in a bit-map or gray-level format can be implemented even if the images are maintained in their "raw" form - as range data.

The first problem we studied was to display images represented in a range data format. This was achieved by converting the range data files into gray-level data files, and

subsequently displaying the latter using display strategies applicable for gray level images. Since it is well known that gray level representations show only *some* aspects of the image in question, our study proceeded along the avenue of displaying other aspects of range data images. To achieve this, we implemented functions to extract edges, trace contour lines and map contour levels, and also demonstrated how numerous features of the raw data could be highlighted by studying the effects of light sources to enhance shading etc.

After being able to display the information contained in images in their raw data format, we proceeded with the next challenge of achieving the comparison between two images that were individually stored in this format. As will be seen, this problem of surface comparison using range data is a difficult task because there are no standardized well-defined criteria for quantifying the measure of similarity between surface images. In this vein, we defined one criterion, the Surface Similarity Measure (SSM) by introducing the concept of the correspondence between points from two data sets. The task of actually evaluating the correspondence involves solving a complex optimization problem, namely an assignment problem, because, in theory, there can be a very large number of ways for determining the correspondence between the two images. We resolved this problem by attempting to determine a correspondence that satisfied some optimality criteria.

The next problem that we tackled was that of actually using the SSM criterion mentioned above to yield surface registration. This is achieved by comparing the two data sets and evaluating the surface registration by computing the SSM between the original

data set and the one it is being compared to. Subsequently, based on the resulting SSM, we can determine how well the range data sets are registered. The entire registration process involves iterations of transformation, comparisons and decisions that are made so as to optimize the SSM, and our intent is to attain the goal of achieving optimized registration.

We conclude this subsection by emphasizing that all of these features have been implemented in a fairly complete prototype utility, the *Manitoba Adaptor Registration System (MARS)*. The design and implementation of this package has constituted the bulk of the work of this project. We are not aware of another package that achieves this because most of the reported image processing software deals with data in gray-level formats and not in the range data formats which we have ventured to work with.

1.3 Thesis Outline

Chapter 2 discusses the kinds of surfaces dealt with, and how they will be represented. In this thesis, *Range Data* is chosen as the data representation for surfaces. The advantages and disadvantages of range data and several methods of interpolation will be discussed in detail in this chapter.

Chapter 3 deals with displaying surfaces. Several methods are presented, and each has its own characteristics, advantages and disadvantages. Contour line drawing, contour level, gray level, and shading with given light sources are the basic methods used to

visualize images. We will show that the display results greatly depend on the nature of the surface. The choice of method, therefore, depends on the purpose of the display, the nature of the surface, and the complexity of the method itself.

Chapter 4 addresses the topic of digital surface comparison. *Digital Surface Comparison* finds differences between two digital surfaces. It is a very useful technique used to detect and correct errors in images and pictures. The usual way to compare surfaces is to define a correspondence between points on the two surfaces, and to measure the distance between the corresponding points. Several ways of defining *Corresponding Points* between two surfaces are discussed in this chapter. The results of using different correspondence definitions on different types of surfaces will be explained.

Chapter 5 considers transformation techniques for the physical displacement of surfaces and demonstrates how we can combine the comparison methods introduced in Chapter 4 with transformation schemes to perform interactive surface registration.

Chapter 6 describes how the existing system, *Manitoba Adaptor Registration System (MARS)*, combines the use of display method, the SSM, and the transformation techniques on data represented as Range Data, to perform surfaces registration.

Chapter 7 concludes the thesis and suggests future research possibilities for MARS.

Chapter 2

Surface Representation

2.0 Introduction

In this chapter, we will describe the types of surfaces that we will be dealing with and their computerized representations. Surface representation is useful for recognizing and positioning objects because they can be measured by sensors, and the 3D object can be easily inferred by a knowledge of its bounding volume and the corresponding surface representation [LT90]. Every data point on the surface can be uniquely located within a 3D coordinate space. In this thesis, we will restrict ourselves to surfaces that can be written as $z = f(x,y)$. Surface data can be classified into two types:

- a) $(x_i, y_i, f(x_i, y_i))$ where (x_i, y_i) are in a rectangular grid; and
- b) $(x_i, y_i, f(x_i, y_i))$ where (x_i, y_i) are not in a rectangular grid.

The first data type is called *Range Data* and the second type is called *Scattered Data*. Scattered data points are irregular in structure, and they can be converted into range data. Different scattered data interpolation techniques will be introduced and an example of the *Modified Quadratic Shepard's Method* will be discussed.

In our discussion, we will focus on surface data which comes from a surface that can be expressed mathematically as $z = f(x, y)$. This is commonly called the surface's *Explicit Representation*.

An example of surfaces that we will not consider is $f(x, y, z) = x^2 + y^2 + z^2$. The equation represents surface points on a sphere, in which a given (x_i, y_i) can have two corresponding values of z_i , or no corresponding z_i values.

2.1 Representing Surface Data

All surface data points can be embedded into the 3D coordinate space with x , y and z coordinates being restricted to a finite range. For example, Figure 2.1 shows a coordinate space with $1 \leq x \leq 64$, $1 \leq y \leq 64$, $0 \leq z \leq 12000$.

Surface data points can be written as $D = (x, y, z)$ which can be written in the form:

$$D = X_0 + \left(\frac{(x - x_0)}{a} + \frac{(y - y_0)}{b} + \frac{(z - z_0)}{c} \right)$$

where $X_0 = (x_0, y_0, z_0)$ is the *origin* of the coordinate space

which is usually chosen to be $(0,0,0)$,

$x_0 \leq x \leq a+x_0$, $y_0 \leq y \leq b+y_0$, $z_0 \leq z \leq c+z_0$, and

there are no two points with the same (x,y) combination.

E.g. $a = b = 64$, $c = 12,000$

=> 64 * 64 square grid with height values no more than 12,000 from the ground

Figure 2.1: A parallelepiped-shaped lattice representing the domain of data.

Consider the range data sets used in MARS as an example, where $a = b = 256$, and $c \geq 0$. In this case, there are 256×256 data points located at the various pairs of (x, y) values, forming the surface, as seen in Figure 2.2.

Figure 2.2: Digital Surface Data.

2.1.1 Mathematical Modeling Function

A mathematically defined function $z = F(x, y)$ can be used to represent the surface $z = f(x, y)$. The basic criteria is that $F(x, y)$ has to be an approximation of $f(x, y)$. In reality, more restrictions have to be taken into account for better approximation. For example, for smooth approximations, $F(x, y)$ has to be C^1 continuous, and so has to have at least continuous first-order derivatives.

Spline Fitting has been extensively used to produce functions representing a surface. Instead of storing every height value, only a few control points are stored. We

approximate the data by a spline surface and represent the spline by its control points. The approximated height value at each point $P(x,y,z)$ can then be computed when needed.

To consider another approximation, we use the *Bezier Cubic Spline* [FD90, RAD90] as an approximation, where the space is defined as in Section 2.1. If we select a few control points from the plane $w = w_0$, and assign a local coordinate (x,y) by using a mapping applied to the coordinates in the x and y directions, then the height values, z , on the Bezier surface are a weighted sum of the control points. The weights are functions called *Bernstein Polynomials*. Each subdivided element on the surface can be interpolated using the tensor product of the Bernstein Polynomial, and we will have a mapping from R^3 to R^3 . A cubic Bezier surface is defined by 16 control points [FD90, RAD90] and

$$F(u,v) = \sum_{i=0}^3 B_i(u) \sum_{j=0}^3 B_j(v) D_{ij}$$

where $0 \leq u, v \leq 1$

B_i, B_j are degree 3 Bernstein Polynomial with:

$$B_0(v) = (1 - v)^3$$

$$B_1(v) = 3 v (1 - v)^2$$

$$B_2(v) = 3 v^2 (1 - v)$$

$$B_3(v) = v^3$$

D_{ij} are coefficients defined so that they can be used as control points to the spline function specified.

The surface represented by using a spline fitting function represents only an approximation of the actual surface model. How well the surface image can represent the surface itself depends on the number of control points used, their placement, and how well the spline function itself is defined. It is well known that the cubic spline is found to produce a good approximation of the surface with a minimum of 16 control points. But like other spline fitting schemes, its accuracy cannot be compared with the accuracy of surface information preserved by storing the surface in the range data format.

2.1.2 Unit Normal and Shape Descriptors List Array (UNSDLA)

Categories of 3D surface shape characterization by differential geometric descriptors and curvature-based representations were first reviewed by Liang and Todhunter in [LT90]. They proposed the UNSDLA representation, which provides unique representation to any class of surfaces. UNSDLA extracts edges from range images and stores the 3D shape characteristics of a surface. It is a generalization of the *Extended Gaussian Image (EGI)*. A surface is represented in two levels: a relational graph is used to represent the connectivity between segmented surface regions, each of which is represented by a 2D list array of unit normals and surface shape descriptors associated with the unit normals. Based on two corollaries to the fundamental theory of surfaces, Liang and Todhunter showed that UNSDLA is a set of geometric descriptors which uniquely identify all classes of surfaces, namely, hyperbolic, elliptic and developable surfaces. Our focus is to deal with raw data and so this method will not be used in this thesis but is mentioned in the interest of completeness.

2.1.3 Image Segmentation

Image segmentation approximates a surface by simple surface patches such as planar and quadratic surfaces. *Region Based Segmentation* and *Edge Based Segmentation* are described by Rodriguez and Aggarwal [RA90] to segmentate a large image into surface patches. Images are partitioned into regions which are homogeneous with respect to some image property such as gray level, color, texture or surface type. Segmentation is used primarily for extracting “invariant” features useful for the tasks of recognition and interpretation. The details of such a representation are described in [RA90] and omitted here as it is not a primary tool used in this thesis.

2.1.4 Range Data

The *Range Data* representation is an approach in which surfaces are represented as a set of height values in the form $z = f(x, y)$ over a rectangular (or square) grid of (x, y) values. It is also known as *2 1/2D data* [YF86] representation. Data are said to be *Regular* because the x and y grid lines are equally spaced. Height values (or Intensity values) of z in a surface can be written as A_{ij} where $i = 0, 1, \dots, m-1$; $j = 0, 1, \dots, n-1$

2.2 Range Data as a Surface Descriptor

Range Data is one of the data structures employed by image processing techniques nowadays. In this thesis, it is used as the surface data descriptor.

There is a lot of literature describing various representations of surfaces in the range data format [BR92, HO84, BA84, WB93, PM90, FL90, RA90, SA91, LT90, CB92, BU93, ZE86]. Many registration algorithms are developed based on the range data representation [BR92, HO84, PR86, CA90, RA90, LT90, CB92, CG84, BU93], and these will be catalogued presently.

2.2.1 Advantages of Range Data as a Surface Descriptor

In this thesis we opt to choose the range data representation as our surface descriptor because range data has numerous advantages over other representations. These advantages are:

- 1) The range data format greatly facilitates description, recognition, and measurement of objects.
- 2) The conversion of data formats between the range data format and other representations can be simple and very straightforward.
- 3) Intensity based segmentation using range data maintains quality of data by retaining the spatial resolution. Segmentation using other representations is subject to different developing procedures that either enhance features and/or “hide” others [HO84].
- 4) There is a big saving in storage because only z values are stored. The domains for x and y are specified (without extra storage or computation) in a simple rectangular shape.

- 5) If the boundaries of the surfaces represented in the range data format are known, it facilitates the ease of accessing data as compared to using other surface representations.

2.2.2 Disadvantages of Range Data as a Surface Descriptor

There are a few disadvantages in using range data as a surface descriptor. These include:

- 1) **Explicit Representation:** Range can only represent surfaces in the form $z = f(x,y)$, and so one would need to represent surfaces such as spheres by two sets of range data.
- 2) **Limitation on Surface Size:** As a raster graphics data representation, range data has a 1-1 correspondence between screen pixels and data values. For example, a large file storage of 65536 data values is required to create a small picture of 256 x 256 pixels. Thus, the time and space required during manipulation increase dramatically with the image size.
- 3) **Limitation on Surface Information:** It is expensive to store extra surface information other than height values. For example, images with color require at least triple the storage, and thus greatly increase the manipulation time.
- 4) **Inaccuracy in Transformation and Comparison:** If one set of range data is to be transformed and then compared with another set of range data, the transformed points have to be approximated by an interpolation scheme. This produces points that do not fall on the original grid, and this is sometimes a disadvantage during surface comparison.

2.3 Converting Scattered Data to Range Data

Data collected from the real world is often scattered and noisy because the processes used in gathering, measuring and storing data are not perfect. Consider the following cases:

- a) Sometimes it is not reasonable to make measurements at one fixed interval. The regularity of data depends on the sample size and the sampling interval during the data acquisition. It is almost impossible to obtain regular surface data set in a real world situation. In Figure 2.3, the relationship between y and z is shown by keeping x constant. z fluctuates a lot at some intervals and is very flat in some other intervals. We would therefore probably use different sampling intervals in order to reduce the amount of data collected.

Figure 2.3: Different sampling intervals are expected in the y -direction.

- b) Surface data taken from real world are from possibly different coordinate spaces and we have to scale and map them into a 3D range data space (e.g. a 256 x 256 grid). This typically requires interpolation as well as approximation. Observe that this applies to regular data as well.
- c) Sometimes it is not possible to make measurements in a rectangular grid. Consider measuring mineral concentrations known at various depths of scattered bore hole locations. In this case the numbers collected will not be regularly spaced due to the nature of data source.

Following are some examples of scattered data in scientific studies [NI93]:

- Pressure values computed or measured at various points on the surface of an airplane wing.
- Precipitation measurements at various weather stations.
- Electroencephalogram (EEG) measurements from electrodes attached to a scalp.
- Mineral concentrations known at various depths of scattered bore hole locations.
- Economic performance levels known at various times, interest rates, and unemployment levels, one being a fixed factor.

The above scattered data can all be written as a finite set of points (x_i, y_i, z_i) where $z_i = f(x_i, y_i)$. To convert scattered surface data to range data representation, we have to interpolate values on a rectangular grid.

2.3.1 Spatial Resolution and Interpolation

“Mathematical models based on high resolution measurements can provide additional insight into the performance of various data analysis techniques.”

[Digital Processing to Improve Forest Digitification, HOP].

Resolution of an object means the separation of the object into its components or subparts. When digitizing a surface, there is always a question of how good the representation is when compared to the original surface data. Besides having to know the adequate sampling rate for data acquisition, it is also very important to note when to group data points as a pixel. **Spatial Resolution** defines the density of pixels in a resolution unit. It describes how an image is broken into discrete pixels and expresses the fineness of the details. Decreasing the spatial resolution implies loss of details by the *Blocking Effect* that is introduced [FD90, BA84]. It can be shown as in Figure 2.4 that tiny rectangular blocks can be seen when data is mapped from a higher resolution (one unit contains one pixel) to a lower resolution (one unit contains nine pixels). The finer this resolution is, the closer we approach the spatial appearance of the original image [KH90].

Since it is desirable that the eyes detect no difference between the digitized image and the original image, the spatial resolution must take into account the following factors:

- i) Details in the original image that are to be seen in the digital image
- ii) Size of the digital image
- iii) Viewer's distance from the image
- iv) Intensity resolution (number of intensity levels in one unit).

a) A 64 by 64 picture with 1 pixel for a data point.

b) A 128 by 128 picture with 4 pixels to represent a data point, leading to an image with a lower resolution than a) above.

c) A lower resolution than a) and b) above where a 192 by 192 picture is obtained with 9 pixels for each data point.

Figure 2.4: Image of Dice obtained with different resolutions.

In some literature, a function is defined to be embedded into the interpolation process and this function determines the spatial resolution of the surface image. Hopskin mentions the *Point Spread Function* (PSF) in [HO84] of one such function.

2.3.2 Scattered Data and Interpolation Methods

Carlson and Foley provided a mathematical definition for scattered data points in [CF92]. For any set $S = \{(x_i, y_i) : i = 1, 2, \dots, n\}$ of n distinct points in the plane, points in S are called *Scattered Data Points* if there is no assumption that they form a rectangular grid. The set of data can be written as $V = \{f(x_i, y_i) : i = 1, 2, \dots, n\}$. *Scattered Data Interpolation* is a procedure for constructing a function $F(x, y)$ so that

$$F(x, y) = f(x, y) \quad \text{for all } (x_i, y_i) \text{ in } S$$

The z values over a rectangular grid of (x, y) can be estimated by function $z=F(x, y)$. Some scattered data interpolation methods are described and compared in [NI93, CF92, FR82]. The most commonly used methods are:

- Basic Shepard's method
- Modified Quadratic Shepards' (MQS) Method due to Franke and Nielson.
- Minimum Norm Network (MNN) of Nelson.
- Local Volume Spline and Volume Spline method.
- 3D reconstruction using a linear least square fitting of feature points.
- Multiquadric (MQ) Method of Hardy.
- Thin Plate Spline (TPS) of Franke.

The above interpolation methods could be classified into two major groups: one uses a network of triangles in the plane using points as vertices, and the other uses radial basis functions in the calculation. All of the methods above belong to the former class except for MQ and TPS. **Basic Shepard's method** is the simplest method among the first group. In the Basic Shepard's method, an inverse distance weighted approximation is used and the function can be written in the form:

$$F(P) = \frac{\sum_{i=1}^n \frac{f_i}{\|P - P_i\|^2}}{\sum_{i=1}^n \frac{1}{\|P - P_i\|^2}}$$

where $P = (x,y)$ is location of point to be interpolated

$P_i = (x_i, y_i)$ is the set of data points within radius R from P

n is the number of points

This method is easy to describe and implement but it has the shortcomings of having local extrema at the data sites [NI93]. In our implementation, we use the **Modified Quadratic Shepards' (MQS)** method. In the case of bivariate data, it eliminates the deficiencies of the basic Shepard's method by localizing the overall approximation by multiplying f_i by a weight function. To do this we define a radius R , which will be the maximum distance of the weighted point from the interpolating point. A weight of zero is

assigned to points with distance greater than R from the point considered. To find the value at a given point x , we define a weight function $w_i(x)$ for each point x_i within the radius R from x (see Figure 2.5) as:

$$\begin{aligned}
 w_i(P) &= \sqrt{\frac{R - d_i}{R * d_i}} && \text{if } R - d_i > 0 \\
 &= 0 && \text{otherwise,} \\
 \text{where } d_i &= P - P_i \\
 &= \sqrt{(x - y)^2 - (x_i - y_i)^2} && \text{in 2D} \\
 &= |x - x_i| && \text{in 1D.}
 \end{aligned}$$

For easy illustration, we will show example of the equation in 2D, where a point P can be written as $(x_i, f(x_i))$, and $w(P) = w(x)$. If N is the number of points found within the radius with respect to x , then the interpolated value $F(x)$ at x is:

$$\begin{aligned}
 F(x) &= \frac{A(x)}{B(x)} \\
 \text{where } A(x) &= \sum_{i=1}^N w_i(x) f_i \\
 B(x) &= \sum_{i=1}^N w_i(x)
 \end{aligned}$$

Figure 2.5: Modified Quadratic Shepards' (MQS) method in which $f(x)$ is approximated. x_i defines the weight function $w_i(x)$, and point 'a' will not be considered.

An example presented below will help clarify MQS. Consider 4 data points in 2D: A(15,10), B(20,20), C(30,20) and D(40,10). If we want to approximate the y coordinate at point P where $x = 25$ with $R = 15$, then:

$$\text{At A: } d_A = \|25 - 15\| = 10$$

$$\text{B: } d_B = \|25 - 20\| = 5$$

$$\text{C: } d_C = \|25 - 30\| = 5$$

$$\text{D: } d_D = \|25 - 40\| = 15$$

Only A, B and C will contribute in approximating P, and we have:

$$\text{At A: } W_A = \sqrt{\frac{15 - 10}{15 * 10}} = \sqrt{\frac{1}{30}}$$

$$\text{B: } W_B = \sqrt{\frac{15 - 5}{15 * 5}} = \sqrt{\frac{2}{15}}$$

$$\text{C: } W_C = \sqrt{\frac{15 - 5}{15 * 5}} = \sqrt{\frac{2}{15}}$$

$$A(P) = \sqrt{\frac{1}{30}} * 10 + \sqrt{\frac{2}{15}} * 20 + \sqrt{\frac{2}{15}} * 20$$

$$= 1.83 + 7.30 + 7.30 = 16.43$$

$$B(P) = \sqrt{\frac{1}{30}} + \sqrt{\frac{2}{15}} + \sqrt{\frac{2}{15}}$$

$$= 0.18 + 2.37 + 0.37 = 0.91$$

$$F(P) = \frac{A(P)}{B(P)} = 18.00$$

Note that since the function $F(x)$ is locally determined, the influence of any point will not extend further than a distance R from other data points (Figure 2.6).

Figure 2.6: Diagram showing MQS with points A, B, C, D. Note that $d_A = 10$, and that D will not be considered because $d_D = 0$.

2.4 Conclusion

In this chapter, we have described different types of surface representations, and the data type that we use in this thesis, namely, the *Range Data* format. We also described

how irregular data can be converted to range data formats using scattered data interpolation techniques.

We have implemented the interpolation method MQS to procure a regular range data format for the MARS prototype system described later. In all brevity, we have also illustrated by means of a simplified example, the technique of obtaining range data by interpolation, using MQS in a uni-dimensional space. In MARS, the z-coordinate is interpolated as the height value of the surface from the x-y plane.

Chapter 3

Displaying a Range Data Surface

3.0 Introduction

This chapter gives a brief description and comparison of display method, for range data surfaces. Four common display methods are described in detail and their implementation results are presented. The methods are:

Gray Level Display: Displays a surface by converting the z values to gray levels.

Contour Display: It shows lines or regions of constant value. Two types of contour display are the: ***Contour Line Display*** displays a surface by showing the lines of constant z , ***contour lines*** as polygons. The ***Contour Level*** shows the regions for which z is in between two limits, and the ***contour regions***, an alternate between black and white.

Shading: Display a surface with gray levels so that it looks as if it were lighted from a given direction.

Edges: Show lines and points of high curvature.

Gray level display and shading both show the entire picture as a matrix of pixels in full gray scale, which is converted directly from the range data. The choice of display method is influenced by:

- whether the display information is sensitive to noise in which case; a display method can be chosen to magnify or reduce the distortion caused by noise;
- which properties of the images will be visualized (e.g., Display by edge shows better image structure compared with display by gray level, while information on textural properties can be shown better using gray level);
- the computational cost for different displays vs. the results. For example, is the extraction of features more computationally expensive to obtain than gray level display? Which one can present the information in a better manner?

3.1 Gray Level Display

From Section 2.4.1, we know that the quality of an image depends on its spatial resolution, which is determined during digitization. Apart from spatial resolution, we also have to consider the **Brightness Resolution (BR)** (also called **Gray Level Resolution**) when bringing an image on the screen. BR refers to the accuracy of surface representation by gray levels and depends on the number of gray levels. The **Gray Level** is the level of brightness spanning from black to white, which is represented as a numeric value within a certain range. The process of representing continuous values or many different discrete values by a small number of discrete values is called **Quantization**. It is used to convert brightness into the discrete gray levels that are available. The continuity of the images is better with an increased number of gray levels.

3.1.1 Mathematical Definition

We will use **Level** to represent the gray level at a point P with height value H. It must be an integer between 0 and N-1. Level is defined as:

$$\begin{aligned}\text{Level} &= \text{round}\left(\frac{(M - H) * N}{M - m}\right) \\ &= \text{gray level at P}\end{aligned}$$

where M is the maximum height value in A;

m is the minimum height value in A;

N is the number of gray levels to be displayed.

A is the rectangular matrix representing the surface.

If a machine uses a number ranging from MAX_INTENSITY and MIN_INTENSITY to represent the gray component of pixels between white and black, then

$$\Delta\text{Intensity} = \frac{\text{MAX_INTENSITY} - \text{MIN_INTENSITY}}{N}$$

= the net intensity change in one gray level.

Intensity at P is calculated from the gray level by the formula:

$$\text{Intensity} = \text{Level} * \Delta\text{Intensity}.$$

The number of gray levels, N, which we can use depends on the number of levels a machine can provide, and any number greater than the maximum number that a monitor can support will be ignored by the hardware. We can vary the range of gray level

intensities by adjusting the values of MIN_INTENSITY and MAX_INTENSITY, and therefore the color does not necessarily range from black to white.

In our implementation, we used Macintosh monitor which could display 256 gray levels. Pixels intensities range from black to white are represented by internal RGB number ranges from 0 to 65535, and consequently:

$$\text{MIN_INTENSITY} \geq 0$$

$$\text{MAX_INTENSITY} \leq 65535$$

Figure 3.1 shows the result of implementing the method described in the above section. Three sets of data are used in the test: a dice, a tooth and horse head. N is chosen to be 2, 4, 8, 16, 32 and 64, displaying picture of gray levels 2 to 64 respectively. The display with more gray levels shows the details better.

Figure 3.1: Dice, tooth and horse data at different numbers of gray levels.

3.1.2 Enhancement on Gray Level

It is often possible to get a better gray level display of an image if we have some knowledge of the image context, and how our eyes perceive the image. The eye is more sensitive to intensity changes in a dark region than in a bright region because it responds logarithmically to intensity change. Therefore it is often beneficial to quantize brightness on a logarithmic scale rather than on a linear scale. *Histogram Equalization* [PA82, FD90] is a technique used to filter the image. It expands the range of intensities used in the region of interest by rearranging the distribution of intensities in the picture. The dynamic range of the picture is increased, thereby making the intensity distribution somewhat more uniform and balanced.

3.1.3 Advantages and Disadvantages

The advantages of the gray level display scheme are:

- It is the simplest display method;
- It shows details of the image because the intensity level at each pixel is displayed;
- It has no need to handle exceptional cases such as a sudden change in intensity;
- The eye can detect contrast quite easily;
- Erroneous data shows up as white or black spots that do not match its surroundings;
- A simpler picture display can be obtained by having fewer gray levels.

For example, a gray level display with $N=2$ show a bi-level picture in Figure 3.1

The disadvantages of using gray level display scheme are:

- The resulting display depends on the local maxima and minima, and consequently image details are hidden by the unbalanced distribution of images intensities. The result can be very unsatisfactory when histogram equalization is not done;
- The number of gray levels displayed is machine dependent. Thus, a 64 gray level monitor would turn a 256 gray level picture into a 64 gray level picture.

3.2 Contour Drawing

A *Contour Line* is a curve in which all points have the same height or intensity and is mathematically written as the set of points (x,y) such that $f(x,y) = C$ where C is a constant height value. There are numerous techniques for contour extraction, descriptions of these are found in [CA90, RA90, ZE86], and a simple contour extraction method is illustrated in this chapter.

A *Contour Region* is an area of the image in which all points fall between two contour lines. It is written mathematically as the set of points (x,y) in the region such that $C_0 \leq f(x,y) \leq C_1$ where C_0 and C_1 are the intensities bounding the region. An image can be segmented into contour regions bounded by contour lines.

3.2.1 Mathematical Definition of Contour Lines

If N is the number of contour levels, M and m is the maximum and minimum A_{ij} respectively. The levels are equally spaced by an amount Δh where:

$M = \text{maximum } A_{ij}$

and $m = \text{minimum } A_{ij}$

$$\Delta h = \frac{(M - m)}{N}$$

Figure 3.2: Different contour levels shown by contour lines, each separated by Δh .

3.2.2 Finding Contour Lines

A simple method for finding contour lines will be described in this section. We will call the region bounded by 4 data points a *Grid Square* (see Figure 3.3). The grid square can be divided into two triangles by either one of the diagonals.

Figure 3.3: A grid square containing 4 data points.

For each triangle, let the maximum and minimum height value be R and S respectively. We refer to the vertex where R is found as V_1 , and V_3 is the vertex with height value S . The remaining vertex will be named V_2 (see Figure 3.4a).

To draw the contour lines that pass through a given triangle, let:

$$r = \text{round}\left(\frac{R - m}{\Delta h}\right), \text{ and}$$

$$s = \text{round}\left(\frac{S - m}{\Delta h}\right).$$

The number of contours that pass through the triangle is $r - s$, and they are the levels $(s+1)\Delta h$, $(s+2)\Delta h$, ..., $r\Delta h$. All these contour lines pass through the edge V_1V_3 and cut one of the two opposite edges at distinct points (Figure 3.4b). A good procedure for finding the contours is to scan from V_3 to V_2 and from V_2 to V_1 . Any contour lines crossing V_3V_2 and V_2V_1 must also cross V_1V_3 . The algorithm joins the crossing points on V_3V_2 and V_2V_1 with the corresponding crossing points on V_1V_3 by straight lines. By

examining the two edges V_1V_2 and V_2V_3 of every single triangle, a contour line map can be constructed (Figure 3.4b & c).

Figure 3.4: Steps showing how a triangle is chosen and how to get the contour.

The continuity of contours depends mostly on the size of the triangles and the flatness of the surfaces. To reduce rounding error in the calculation of Δh , r , s , all computations are typically performed with floating point numbers. This creates a much more accurate contour line of the object. Figure 3.5 shows the contour line extracted from a 256 by 256 tooth images using the above extraction scheme.

Figure 3.5: Contour line drawing of tooth data.

3.2.3 Finding Contour Regions

We now consider the problem of determining the contour regions in which the various points fall. Figure 3.2 represents a surface with four regions of different contour levels and each point (x,y) falls in one of the regions:

$$\text{level 1:} \quad 0 \leq f(x,y) \leq \Delta h,$$

$$\text{level 2:} \quad \Delta h \leq f(x,y) \leq 2\Delta h,$$

$$\text{level 3:} \quad 2\Delta h \leq f(x,y) \leq 3\Delta h,$$

$$\text{level 4:} \quad 3\Delta h \leq f(x,y) \leq 4\Delta h.$$

Points lying on odd levels will be displayed in black and points in even levels will be displayed in white. Figure 3.6 shows the contour region display of Figure 3.2. Observe that since the z values are explicitly stored for the x - y coordinates, obtaining the contour is straightforward, and requires no sorting.

Figure 3.6: Four contour regions resulted from contour lines in Figure 3.2.

3.2.4 Advantages and Disadvantages of Contour Display Schemes

The advantages of contour display schemes are:

- Displaying surfaces with contour lines or contour levels gives the outline of surface intensities and the structure of the surface;
- In comparison with gray level display, contouring provides accurate positioning of the regions of intensity change, local maxima and minima [KH90];
- Contour lines and contour regions both show the rate of change (steepness) of height values clearly;
- Contour displays are good for comparison at a coarser level; comparisons based on corresponding contour lines or regions reduce the number of pixels to be compared;
- Contour lines which are extracted can be used as features in surface registration;
- The complexity of picture can be altered by changing the number of contour levels.

The disadvantages of contour display schemes are:

- These schemes show less surface detail when compared with gray level display;
- Flat surfaces cannot be visualized by this method very well;
- Exceptional cases such as large changes in intensity levels may cause contours to be clustered, making them difficult to see;
- While contour schemes show the steepness of surface, they give no information on the actual height values and the direction of the slope (Figure 3.7).
- A change in level might change the topology of resulting contours (Figure 3.8)

Figure 3.7: Contour lines with same topology can represent 2 entirely different surfaces.

Figure 3.8: The two contours represent the same picture but have different topologies.

3.3 Lighting from Different Directions

Shading an image shows the effect of a light source on a surface image due to the *Reflection of Light* (Figure 3.9). The *Normal* at a point on the surface is a vector perpendicular to the surface at that point. There are three vectors in Figure 3.9: the normal, light source, and observer direction. Consider the observer's eyes located perpendicular to the tangent plane of the surface at the point.

Figure 3.9: Reflection of light at point P.

For a given surface, the brightness of the reflected light is proportional to the cosine of the angle between the light direction and the normal to the surface and thus:

$$\text{Intensity} = k \cos \beta$$

where k is a number depending on the texture

and for $\cos \beta \leq 0$, intensity will be zero, which is black because the light does not reach these points

Similar to gray level displays, in this case the brightness resolution has to be considered. Gray levels from white to black are used to represent the intensities. In the following section, we will introduce a simple method for getting the normal at a point on the surface.

3.3.1 Approximating Normal at a Point

The normal vector generally varies over the surface. The normal at each point is approximated from the normal at the points closest to it. For every pixel point C , we take the two vectors AB and $A'B'$, or $B-A$ and $B'-A'$ in vector notation (see Figure 3.10). The normal at point C can be approximated by taking the normal of the plane defined by vectors AB and $A'B'$. In Figure 3.11, the normal of the plane defined by vectors EF and $E'F'$ is also an approximation to the normal at point C . For a better approximation, we can include all the 8 pixels surrounding C by taking a weighted average of the plane normal defined by vectors AB , $A'B'$ and plane normal from EF , $E'F'$.

Figure 3.10: Normal at C is approximated by vector AB and $A'B'$.

Figure 3.11: Normal at C is approximated by vector EF and E'F'.

3.3.2 Determining the Shading at a Point

For every point P on the surface S, we can calculate the unit normal N at P. Let L be a unit vector representing the reflected light in any arbitrary direction. Then, the cosine of the angle β between N and L is the vector dot product of N and L:

$$\cos \beta = N \cdot L$$

Figure 3.12: Maximum and minimum intensities of light L at different angles.

The light source is assumed to be consisting of parallel light. The intensity is zero when $\cos \beta \leq 0$ because the reflected light is behind the x-y plane pointing away from the observer, and will therefore not be visible if the observer is in front of the x-y plane (positive z-axis). The reflected light will not be blocked by other points on the surface because there is no visible surface determination in our implementation.

a) $L = (1, 0, 0)$

b) $L = (-1, 0, 0)$

c) $L = (0, 1, 0)$

d) $L = (0, -1, 0)$

Figure 3.13: Pictures showing effect of light source from different directions.

a) $L = (-1, 0, 0)$

b) $L = (-1, 0, 1)$

c) $L = (-1, 0, -1)$

Figure 3.14: Effect of having a light source from same direction but from different angles.

$L = (1, 1, 100)$

Figure 3.15: Effect of a light source with a negative angle. The black region is due to the normal making a negative angle with the light source.

3.3.3 Advantages and Disadvantages of Shading

The use of different light directions can show details that are not visible with any one direction (see Figures 3.13 and 3.14). Figure 3.15 shows the effect of having the light source behind the object. The following points summarize the advantages of displaying by shading:

- Display by shading shows various different features of a surface based on its spatial information, which are not provided by a gray level display;
- By changing the directions of the light source, details of structure and features at regions of interest can be highlighted while suppressing other regions;
- Shading is easier to implement than contour extraction.

The disadvantages of shading are:

- The display result depends very much on the direction of the light source, which has to be determined by trial and error;
- Since we use an approximation of the normal to determine the result, this may lead to an inaccurate display;
- The algorithm does not perform visible surface determination and so the reflected light is not blocked by other points on the surface when it should be.

3.4 Edge Representations

Figure 3.16: A simple step edge.

An *Edge* can be defined as a sharp change in a height value. Edges can be detected by finite difference operators. In this case, threshold values are used because finite differences are not absolute. The *Threshold* is a value defined so that each pixel falls above or below the threshold, resulting in two categories [PA82]. Fleck suggested in [FL90] that in a sequence of scalar values, the edge might be located at the point where the second finite difference approaches zero. The finite difference E is defined as:

$$E = \frac{I(P+w) - I(P-w)}{2w}$$

where P is the image cell,

$I(P)$ is the intensity value at P , and

w is the cell distance that most accurately represents fine details, and is usually 1.

If $\text{absolute}(E) > \text{threshold}$, which is a small non-zero prescribed number, P is said to be on an edge. Otherwise, P is not on an edge.

The *Laplacian operator* (figure 3.17) is another example of an edge detector. The kernel of the operator is defined as a matrix centered at the current location P, and it is invoked by multiplying the elements in the matrix with the corresponding surface values. If the sum of the products is larger than a specified threshold, the location P is said to be on an edge [PA82] of a boundary.

Figure 3.17: The discrete version of the Laplacian operator.

Figure 3.18: Results by applying the Laplacian Operator on the dice, horse head and tooth images.

There are some cases where edge detection using simple finite differences does not work well. Edges may not be ideal step edges, and most typically are a combination of steps, peaks and roof profiles (see Figure 3.19). This implies that there are systematic errors in detection and localization. Also, false edges can be detected at smoothly shaded regions with constant gradient of the image brightness. Perona and Malik in [PM90] proposed an edge finder to detect and localize composite edges. They solved the problem of false responses in smoothly shaded regions with constant gradient by using a nonlinear filter called the *Quadratic Filter*. Further details on edge finders as well as noise reduction can be found in [FL90, ZE86].

Figure 3.19: Some examples of edges [PM90] determined using edge detection schemes.

3.4.1 Advantages and Disadvantages of Edge Display

The advantages of edge display schemes are:

- These schemes show the salient features of a surface;
- Satisfactory results can be obtained by relatively simple operators;
- Like contour line displays, these schemes are good for comparison at a coarser level;

- The extracted curves can be used as a feature for surface registration;
- The number and position of the edges found can be altered by changing the value of the threshold;

The disadvantages of edge display schemes are:

- Exceptional cases like having edges that are a mixture of different types may cause errors when detecting the edges (Figure 3.19);
- A poor choice of threshold may flatten the edges, widen the edges, or make them disappear. In these cases, it could be very difficult to comprehend the picture that the edge finding process produces.

3.5 Conclusion

In this chapter, we have described different schemes of surfaces display in range data surfaces. The schemes include: Gray Level display, Contour Regions and Contour Levels displays, Shading, Edges. Mathematical formula for implementation and results of each scheme are shown and for each scheme, the advantages and disadvantages of using the scheme are described. In MARS, we have implemented all display schemes using the dice, horse and tooth range data, and we can compare the result in different schemes. In MARS, all schemes show the object feature well. In real life application, the use of different display schemes will be dependent on the use of the result and the nature of the range data file.

Chapter 4

Comparing Surfaces

4.0 Introduction

This chapter focuses on the problem of surface comparison. Surface comparison is an essential part of surface registration. During surface registration, we have to compare the surfaces likenesses, which is called the *Surface Similarity Measure (SSM)*, to determine if the registration is optimal. SSM between two surfaces S_1 and S_2 depends on determining corresponding points between the two surfaces. Given a point in surface S_1 , there will be a point in surface S_2 that corresponds to the point in S_1 ; the two points are called the *Corresponding Points* to each other. Three ways of setting up corresponding points are by computing the: the *Vertical correspondence*, the *Perpendicular correspondence* and the *Equal Angle correspondence*. The SSM is defined in terms of the Euclidean distances of all the correspondences. The simplest similarity measure is the sum of the distances between all the corresponding points. Other definitions on similarity measure based on the three correspondences are described later in this chapter.

Various SSMs have been defined in the literature [BR92, WB93]. The similarity metrics chosen for surface comparison depends on the nature of the comparison. The key points to consider are:

- Should the range data itself or the extracted features from the surfaces be used in the comparison?

- How do we define the correspondences to measure the similarity between two surfaces?
- How do we represent the similarity between the surfaces; e.g. by using a numerical representation or visually by showing the similarity at every pixel?

Below are some examples of the applications of surface comparison:

- As a technique for *error detection*. In this case, we would compare two digital surfaces which should be identical. If there is a mismatch between the surfaces, then there are errors in one or both of them.
- As an *error correction* tool. The information stored as a digital surface can be accessed easily. This is an important feature when modifying the data set for error correction or noise reduction. An example can be given using the images of a horse's head. The pictures are not the same because they are viewed from slightly different angles, but we can correct errors in their backgrounds because they should have the same background (Figure 4.1). A comparison can be made and the missing regions found in the background are estimated from the correct data (Figure 4.2).

Figure 4.1: Image of Horse Head at different angles.

Figure 4.2: Background identified by region coherence after identifying identical regions.

4.1 Featureless Vs Feature Comparison

Surface comparison is classified into two types: Featureless and Feature-based comparison. In *Featureless Comparison*, all the pixels on the two surfaces are compared. In *Features-based Comparison*, a set of points characterizing the intrinsic features, such as the maximum and minimum curvatures, or sharp edges are identified. The *features* from the two surfaces then serve as the basis for the comparison [RA90, MW94, CA90].

The choice between the two types depends on the purposes of comparison and the nature of the surface data. Feature-based comparison works well if the two images have a small number of easily identified features. Featureless comparison is used when there are no easily identifiable features.

4.1.1 Advantages of Featureless and Feature Comparisons

The advantages of featureless comparison are:

- Since featureless comparison compares every pixel in the surface, it is easy to implement. In feature comparison, a filtering algorithm is needed to extract the features as sets of critical points which are then compared;
- Resulting differences measured using featureless comparison can be displayed pixel by pixel, and hence they can be visualized and easily understood;
- The result from feature comparison depends on the extracting algorithm used and is more sensitive to erroneous data. The result from featureless comparison is more consistent.

The advantages of feature-based comparison are:

- Feature-based comparison can be used when only feature of the object are given;
- Feature-based comparison is generally faster than featureless comparison because fewer comparisons are made.

Comparison methods described in this thesis are featureless comparison schemes because we are dealing with range data in the raw format, and not with the object features.

4.2 Corresponding Points

A point D_1 on surface S_1 that corresponds to a given point D_2 on surface S_2 is called the *Corresponding Point* of D_2 . In Figure 4.3, D_1 , D_2 correspond. D_1 is chosen

to be the corresponding point of D_2 at P and vice versa. The *SSM* is defined by measuring the closeness of the corresponding points between the surfaces S_1 and S_2 . Setting up the corresponding points determines how well a similarity measure is defined.

Figure 4.3: Corresponding points D_1 and D_2 taken from S_1 and S_2 respectively at $P = (10, 10)$. D_1D_2 is parallel to the z -axis.

The following sections in this chapter describe some approaches of setting up the corresponding point for a given point P (Figure 4.4 and 4.5). We will first consider the simpler problem of finding corresponding points between curves due to the following reasons:

- a) The ideas are easier to illustrate with curves;

- b) The methods developed for curves can be extended so that they apply to surfaces.

Figure 4.4: Corresponding points C, P of two curves taken from two surfaces.

Figure 4.5: Corresponding points C, P from two curve segments.

4.2.1 Vertical Correspondence

Vertical correspondence is the simplest way to define corresponding points between two surfaces. For a given point (u, v, w_1) on surface S_1 , (u, v, w_2) on S_2 is its

corresponding point on S_2 . In Figure 4.6, C and D are vertical corresponding points of each other.

Figure 4.6: Vertical Correspondence.

This approach is simple, but it does have some limitations. The vertical corresponding points do not always satisfy the intuitive idea of correspondence. In Figure 4.7, C does not correspond well to D because there are points on C_2 that are much closer to D than C.

Figure 4.7: Vertical correspondence; C, D correspond.

4.2.2 Perpendicular Correspondence

Perpendicular Correspondence of point D on S_1 is a point C on S_2 such that DC is perpendicular to S_2 (Figure 4.8).

Figure 4.8: Perpendicular Correspondence; In this case C corresponds to point D.

To find perpendicular correspondence, we have to select a *Search Space*, or *Search Range* for curves. *Search space (Search Range)* sets the boundaries for the algorithm when solving for the corresponding points. Points that do not fall in the search space or search range will not be considered as corresponding points (Figures 4.9 and 4.10). Figure 4.9 shows that perpendicular correspondence of P is found by solving for the perpendicular intersect on C_2 from P within the interval $[x-r, x+r]$. This can be done by traversing all pixels within the search space, and finding the normal N of each pixel as described in Section 3.3.1.

Figure 4.9: Search range of P given radius r .

Figure 4.10: Search space of P given radius r .

In Figure 4.10, given P on S_1 , we can find P' on S_2 such that $P_x = P'_x$ and $P_y = P'_y$, use the search space for P is the circular area of radius r with P' as center. The perpendicularly corresponding point to P can be found by computing the point on S_2 whose normal passes through P within the search area.

Perpendicular correspondence is independent of the orientation of the surfaces; this is an advantage over vertical correspondence. Another advantage is that the perpendicular corresponding point to P is the closest point on S_2 to P . In Figure 4.11, C_2 is a very small interval of a curve, so that it can be approximated by a straight line. If PD is perpendicular to C_2 , joining any two points D_1 and D_2 on C_2 to P , forms a triangle PD_1D_2 . PD as the height of $\triangle PD_1D_2$, which is always the shortest among PD_1 and PD_2 . Therefore, D is the closest point from P at C_1 to C_2 .

Figure 4.11: P, D correspond, D is the closest point in C_2 to the point P in C_1 .

A given point D in C_1 may have several or no perpendicular corresponding points in C_2 (see Figure 4.12 to Figure 4.14).

Figure 4.12: A simple case of more than one corresponding point.

Figure 4.13: Another case of obtaining more than one corresponding point. P has more than one perpendicular correspondence from S_2 .

In Figure 4.13, PC_1 and PC_2 are both perpendicular to surface S_2 . The following rules can be used to select a perpendicular correspondence between C_1 and C_2 :

- Eliminate all points outside the search space;
- If both C_1 and C_2 are in the search space, choose the point closer to P .

**Figure 4.14: Perpendicular intersection from P to C_2 does not fall on the line segment.
There is thus no perpendicularly corresponding point to P .**

A perpendicularly corresponding point, if any is found for a given point P , is the closest point to P on the other surface. If the perpendicular intersect does not exist (Figure 4.14), a good approximation of perpendicular correspondence is the closest point from P on C_2 . The *Closest Point Correspondence* can be used when there is no perpendicular correspondence because there must be at least one closest point.

Perpendicular correspondence is not reflexive or symmetric in nature. A pair of perpendicular corresponding points does not always correspond to each other. In Figure 4.15, C corresponds to D ; and $D' \neq D$ corresponds to C . It is desirable to have $C \leftrightarrow D$, where C corresponds D given D on S_1 and D corresponds C given C on S_2 .

Figure 4.15: Different results by choosing points $C_1 \rightarrow C_2$ and $C_2 \rightarrow C_1$.

4.2.3 Equal Angles Correspondence

Equal Angles Correspondence solves the non-symmetric property of the perpendicular correspondence definition. The corresponding point of D on C_1 is a point C chosen from C_2 such that DC makes an equal angle with C_1 and C_2 (Figure 4.16).

Figure 4.16: Equal Angles Correspondence.

Given D and curves C_1 and C_2 , we can solve for the C where CD makes the same angle with C_1 and C_2 . As in the case of perpendicular correspondence, a search radius is chosen to bind the curve when searching for point C. There may be none or more than one

point that corresponds to D . In Figure 4.17, D has two corresponding points, C' and C'' . When more than one corresponding point to D is found, the point closest to D is chosen.

Figure 4.17: A point with two equal angle correspondences.

Equal angle correspondence has the same disadvantage as perpendicular correspondence, a pair of corresponding points do not always correspond to each other. In Figure 4.18, we get C' and C'' as corresponding points for D' ; starting from the opposite surface, we get D' and D'' from C' . We identify the relevant cases as follows:

- If $C' D''$ is the shortest among the three lines $C' D'$, $C'' D'$, $C' D''$, then $C' D'$ will form an ideal correspondence;
- $C' D'$ is the longest among the three lines, then C' will correspond to D'' and C'' will correspond to D' ; there may or may not be any ideal correspondence among these points.
- $C' D'$ is in-between $C' D''$ and $C'' D'$. There will be one ideal correspondence pair on either C'' or D'' , depending on which one of $C' D''$ or $C'' D'$ is longer;

- If only two consecutive points (C' and D') have more than multiple correspondences, then $C' D'$ will still be the ideal correspondence if $C' D'$ is the shortest within $C' D'$, $C' D''$ and $C'' D'$

Figure 4.18: Multiple corresponding points from two curves.

It can be seen that the probability of having different equal angle correspondence pairs with measurements initiated from the two different directions is very small. The solution to the problem of equal angle correspondence is usually not robust, and is time consuming. Another problem is there may be no corresponding point to the points C' , C'' , D' or D'' using equal angle correspondences.

4.3 Similarity Measure

The *Surface Similarity Measure (SSM)* which we propose quantifies the likeness between two surfaces. For each point in surface S_1 , we will find a corresponding point on

surface S_2 . An SSM is defined by summarizing the distances of all corresponding points between the surfaces S_1 and S_2 .

Figure 4.19: The distance between corresponding pairs D_1 and D_2 is a measure of closeness between S_1 and S_2 at $P = (10,10)$.

The similarity measure between two surfaces can be represented in two ways:

- 1) *Pixels based:* In this case the similarity is represented as a matrix of numbers, consisting of the height difference of every pixel and its corresponding point;
- 2) *Surface based:* In this case the SSM consists of a single number showing the similarity of the two surfaces in their entity.

A pixel based similarity measure is useful for range data, because comparison is made at every pixel/point, and the result can be displayed by methods described in Chapter 3. The most commonly used units for the similarity measurement are the pixel

gray values (intensity), the height values, the gradient (the angles with the reference plane or the gradient with respect to the surrounding 4 or 8 points) or the curvatures at the respective points. The unit used depends on the nature of the comparison involved, and thus, for example, we may want to use the intensity as unit for the purpose of displaying the final result.

Instead of having to describe similarity at every single pair of surface points, we can collapse the information into a single value – to yield a Surface Based similarity measure.

Figure 4.20: Surface Based Similarity Measure.

In Figure 4.20, the goal is to maximize the similarity measure. If any two corresponding points are identical, the similarity measure is infinity. Sometimes it is not necessary to collapse all the points. An example of this is the similarity measure on separate contour lines extracted from the surfaces. Independent of how similarity measure is represented, the distance between two surfaces constitutes two major components that are based on:

- 1) The definition of the unit to be compared, and
- 2) The definition of the allocation of corresponding point on the second surface given the original point on the first.

Pixel-based similarity representations need no further processing since the points are gathered on a point-by-point basis. The single number obtained in the case of surface-based similarity measures is obtained by extracting a single quantity from the matrix of similarities (pixels based) either by summing the numbers in the matrix or taking the average of these numbers. Both representations serve to provide a relative measurement on how well the two surfaces match.

4.4 Conclusion

In this chapter, we discussed the need for a similarity index that compares surfaces, and explained the details of one particular measure that we called the Surface Similarity Measure (SSM). We also discussed the various schemes by which corresponding points can be defined within the context of the SSM, and how they can be used in surface registration.

In particular, we described three methods by which we could define the corresponding points for the SSM: the Vertical Correspondence, the Perpendicular Correspondence, and the Equal Angle Correspondence. The advantages and

disadvantages of each scheme were listed, and possible enhancements for each scheme were suggested in this chapter.

In the system that we have developed, MARS, we have implemented the SSM using the Vertical Correspondence method and Perpendicular Correspondence method. Equal Angle Correspondence was not implemented in MARS, and is a possible future extension.

Chapter 5

Interactive Registration of Surfaces

5.0 Introduction

Surface registration is the process of aligning two surfaces which involves two steps: *comparison* and *transformation*. As mentioned earlier, the *Surface Similarity Measure (SSM)* is used as a tool to measure how well the two surfaces compare. The *Transformation* operation modifies one of the surfaces in order to maximize the similarity between the other surface and the transformed surface. Registration of surfaces allows one to relate pieces of information available in several images of a given object. It is an important tool for object localization and identification from sets of surface data. While choice of transformation and comparison techniques used depends on the surface content [CB92], the goal is to facilitate the registration process.

In chapter 4, strategies for general surface comparison were described which indicated how we could handle different surface types in interactive registration. Other registration methods that use similarity measures can be found in [PMR91, PR86, WB93, JA93, CA90, RA90, SA91, LT90, CB92, BU93].

In this chapter, we will explain how to transform range data and how comparison and transformation can be put together to achieve surface registration. Below are some applications of registration:

- i. **Data Merging:** Registration is critical when we are merging two related data sets. This is especially useful when we have a connected surface represented by two data sets having some *overlapping regions*. On one hand, the overlapping regions might be known beforehand to get the best registration. On the other hand, the regions can be found during registration. By merging the overlapping regions, a union of the two data sets can be done. A *perfect merging* is obtained when the overlapping regions match exactly.

An example where registration is required is in analyzing dental data. Two data sets representing the left and right side of the same jaw are obtained. The overlapping region is the middle part of the jaw, which can be found in both data sets. Because of imperfect alignment of the apparatus, the complete jaw cannot be obtained by simply combining the two images. At least one of the images has to be translated or rotated to the correct location before the two images can be merged. The registration on the overlapping regions of two images results in an image of the whole jaw.

- ii. **Instrument calibration:** For determining or calibrating the position of the measuring instrument which captures the images. By knowing the transformations performed on the images in the registration, the position of instrument for the next measurement can be adjusted.

5.1 Registration Algorithm

Figure 5.1 shows how registration can be performed by transformations until the SSM is minimized.

Figure 5.1: Utilizing the surface similarity measure for interactive registration.

The transformation used in surface registration is determined by the similarity measure used. The similarity measure can in turn be improved by the corresponding transformation. Whereas Figure 5.1 shows the basic transformation steps, we describe below an example of transformation using feature comparison. This example clarifies a number of issues that need to be considered when designing a registration algorithm.

Carson's registration by features in [CA90] points out that registration used on features in the presence of geometric uncertainty is difficult in the domain of object localization for three reasons:

- i. The object occlusion in a scene implies failures in feature extraction, and consequently some model features may have no corresponding image feature;
- ii. If there are other objects in the image, some image features will not correspond to any model features and these features are spurious; and
- iii. Due to inaccuracies in sensing and feature extraction, there can be uncertainty in the geometry of the image, and hence the resulting image features may be distorted.

Note that while the term "model" refers to the geometric model of the object to be registered, the image is the snapshot of the object's geometric model, which is what we have referred to as the "surface".

The following points highlight properties that a registration algorithm should have in order to effectively perform registration:

- i. The algorithm should be able to handle the presence of geometric uncertainties. For example, the algorithm should be capable of achieving registration of surface shapes with occlusion, and performing matching a part of a surface shape to the whole surface shape.
- ii. The transformation of one of the surface should be determined from the second surface.

- iii. The algorithm should be able to determine the orientation and scaling factor between the two surfaces images. This is because the model features may be similar in shape but different in size.
- iv. The algorithm must have a simple implementation that will permit parallel processing for acceleration of the registration process [LT90].

A general registration algorithm involves two major steps:

- i) *Choice of a geometric transformation and estimation of its parameter*

Choice of the transformation parameter is a very important step. Proper registration cannot be obtained if the transformation parameters do not correctly represent the actual geometric modifications. Estimation of the transformation parameters could be done in two ways:

- a) *Interactive transformation*: In this case, we simply estimate the transformation parameters interactively using the displayed differences; and
- b) *Assign Correspondence*: In this case the system would establish feature correspondences to compute the transformation parameters. This task deals with four central issues [SA91]:

- selecting the feature to be used for the task;
- representing the transformation motion;
- representing the feature; and
- using the representations to compute the transformation.

ii) *Implementation of the transformation made*

Evaluating the registration quality requires the definition of a measure of similarity between the two images after registration. This is a difficult task because even if images are correctly registered, residual differences may exist due to inaccuracies in measurements and in the scene variations. Quality evaluation must be able to differentiate residual differences from misregistration, and to determine the influence of the former on registration robustness. Step i) will be repeated until we get a satisfactory evaluation.

In Chapter 4, we presented methods of measuring the similarities between surfaces under different conditions. The following section will focus on describing the transformations that can be done on range data.

5.2 Affine Transformation on Range Data

Affine transformations are a special transformations that include rigid body motion and scaling. It preserves the parallelism of lines and the object being transformed is not distorted. Therefore, it can only match surfaces with positional differences, but not those with differences involving object characteristics.

From Figure 5.1, we see that a transformation can also be used to enhance the result of surface comparison. Sometimes surfaces can be preprocessed by transformation to reduce complexity of the comparison. For example, steep surfaces can be rotated to

the horizon as close as possible in order to reduce the steepness, and then they can be compared by a vertical correspondence.

The matrix operations for affine transformations provide the spatial reorientation of data points within the surface. Data points from an input surface may be transformed into new spatial locations, as defined by the geometric algorithm, resulting in changes of its surface characteristics. But the improvement from transforming is limited by range data. This is because a transformation cannot be carried out analytically for all possible orientations since the transformation space is discretized. This has the effect of decreasing the accuracy of transformation [LT90]. The goal of a registration algorithm is to find the transformation that maximizes the similarity measure. The three affine transformations that will be used in our implementation are *Scaling*, *Rotation* and *Translation*.

The three transformations described below are performed on pixel position (x,y) , which is transformed to new location (x',y') . The position changes, but not the height value. Therefore, if $F(x',y')$ is the pixel value of transformed surface at (x', y') and $f(x,y)$ is the original surface value at (x,y) :

$$F(x',y') = f(x,y)$$

5.2.1 Scaling

Scaling deals with the *enlarging* and *shrinking* of a surface or portion of a surface. It allows the two input surfaces to be size-adjusted before they are compared and transformed by other operations.

The equations for the scaling operation can be written as:

$$x' = Sx \quad \text{and} \quad y' = Sy$$

where x and y are the coordinates of the input data points;

x' and y' are the new coordinate of data points; and

S is the scaling factor.

The scale factor S indicates the amount of magnification (enlarging) or demagnification (shrinking) that will occur. For example, if $S = 2$ is applied to a 256 x 256 picture, data from (x, y) of the original picture will be stored in $(2x, 2y)$ of output picture. Thus the pixels in the values from $(0,0)$ to $(127,127)$ are mapped to form a 256 x 256 output frame resulting in a new picture.

a) Original Dice Image;

b) Output Dice Image at $S = 2$ c) Output Dice Image at $S = 3$

Figure 5.2: Magnification of Dice with Scaling.

Figure 5.2 shows the result of magnification using the Dice image with the scaling factors $S = 2$ and $S = 3$. In Figure 5.2b, pixel locations in the output image where x' or y' are odd numbers have no valid pixel information to be mapped to and they are filled by replication. The pixel to the immediate left of a odd column position is usually replicated into that position. Likewise, the line of pixels above an odd line (where y' is odd) is replicated into that line. In Figure 5.2c, pixels with x' or y' congruent to 1 and 2 mod 3 are filled by replication of previous x' and y' which are congruent to 0 mod 3.

a) Original Dice Image; b) Output Dice Image at $S = 0.5$

Figure 5.3: Scaling of image by an S value less than unity.

Figure 5.3 shows image shrinkage by a factor of 2 with $S = 0.5$. The image of size 256×256 is mapped into an output image of size 128×128 . An input location of $(67, 67)$ yields an output location of $(33.5, 33.5)$ which will be rounded off to $(33, 33)$. Similarly, values in output location with $x \geq 128$ or $y \geq 128$ will be undefined. Usually, a background value will be assigned to it. In our example, the number 1 or 65535

(Maximum intensity or Minimum intensity) is assigned to these locations, which produced a white background.

The formal pseudo-code is given in Algorithm Scaling below:

Global Input: Original pixel (x, y), Scaling factor S, column replication counter PRC, line replication counter LRC

Global Output: Return pixel (x',y')

Algorithm Scaling
Get S from user

If S > 1 { /* Enlargement */
Reset x, y, x', y'
PRC = LRC = S

While True {
ReplicatePixel (x, y, x', y')
If x = max pixel index {
If y = max line index { return }
LRC = LRC-1
y' = y' + 1
If LRC = 0 { newtLine }
LineReplicate
} **Else** {
PRC = PRC -1
x' = x' + 1
If PRC = 0 { nextPixel }
} /* If x=max... */
} /* While True */

} **Else** { /* Reduction */
Reset x, y, x', y'
S' = 1 / S

While True {
ReplicatePixel (x, y, x', y')
If x = max pixel index {
If y = max line index { return }
y = y + S'
y' = y' + 1
Reset x, x'
} **Else** {

```

        x = x + S'
        x' = x' + 1
    } /* If x=max.. */
} /* While True */
} /* S>1 */
End
End Algorithm Scaling

```

```

Algorithm Reset
    Assign 0 to each parameters passed
End Algorithm Reset

```

```

Algorithm ReplicatePixel (x, y, x', y')
    Retrieve input image pixel (x,y) and store it to output image pixel (x',y')
End Algorithm ReplicatePixel

```

```

Algorithm newLine
    y = y + 1
    LRC = S
End Algorithm newLine

```

```

Algorithm LineReplicate
    Reset x, x'
    PRC = S
End Algorithm LineReplicate

```

```

Algorithm nextPixel
    x = x + 1
    PRC = S
End Algorithm nextPixel

```

5.2.2 Translation

Translation allows the horizontal and vertical displacement of the pixels of an image. The equation for the translation operation can be written as:

$$x' = x + T_x \text{ and } y' = y + T_y$$

where x and y are the coordinates of the input image;

x' and y' are the coordinates of output image; and

T_x and T_y are the amount of translation in the x and y directions.

For example, with $T_x = 10$ and $T_y = -50$, the pixel value at $(100,100)$ will be assigned to pixel $(110,50)$ in the output image. The image will be moved to the left by 10 pixels and down by 50 pixels. As in 5.2.1, locations not assigned any value will be filled with the background intensity of either 1 or 65535. Figure 5.5 shows a typical image translation.

Figure 5.4: Image translation with $T_x = 10$, $T_y = -50$.

When the input image is of full size or taking up large portion of the frame, the side is often translated off the image frame that will result in a loss of image data which cannot be recovered from reverse translation operation. Wraparound can be used to avoid this loss (Figure 5.6), in which case, the pixels that go off on one side come up on the opposite side of the frame.

Figure 5.5: Wraparound effect on a large format image.

The formal pseudo-code is given in Algorithm Translation below:

Global Input: Original pixel (x, y), Translation Scalar Tx, Ty

Output: Return pixel (x',y')

Algorithm Translation

Get Tx Ty from user

Reset x, y

While True {

 x' = x + Tx

 y' = y + Ty

 ReplicatePixel (x, y, x', y')

If x = max pixel index {

If y = max line index { return }

 y = y + 1

 Reset x

} Else {

 x = x + 1

} /* If x = max.. */

} /* While True */

End

End Algorithm Translation

Refer to 5.2.1 for other modules used.

5.2.3 Rotation

The equations for rotation about the origin are :

$$x' = x \cos \phi + y \sin \phi \quad \text{and} \quad y' = -x \sin \phi + y \cos \phi$$

where ϕ represents the angle of *clockwise* rotation about the origin;

and $0^\circ \leq \phi \leq 360^\circ$.

Before a rotation operation can be performed, we have to translate the image so that the point we wish to rotate the image about is at the origin. Then we can perform the rotation, followed by a reverse translation. The center of image frame is usually chosen as the center of rotation, which has to be translated to the origin (0,0) for the operation.

For example, with $\phi = 90^\circ$, giving $\sin \phi = 1$ and $\cos \phi = 0$, we have $x' = y$ and $y' = -x$. Pixel (255,61) is translated to (127,-67) before the rotation. Rotating to pixel (127,-67) yields the new location (-67,-127), which will be translated backward to yield the output coordinate (61,1). Applying the transformation to all pixels within the input image, we will get an output image rotated by 90° . When ϕ is not a multiple of 90° , the output pixel coordinates are non-integer value because $\sin \phi$ and $\cos \phi$ are irrational. To handle this problem, we can store the data value to the nearest pixel in output frame. But this technique tends to transform straight lines into jagged lines, producing a resultant transformed image that is very inaccurate and of bad quality.

Techniques like *Interpolation* and *Reversed Transformation* algorithms are often employed to improve the quality of output images.

5.3 Enhancing the Transformation

In 5.2.1, we explained that undefined pixel locations are filled by replication of their neighbouring image pixels. Shrinking produces inaccuracy from rounding the fractional part during the scaling process. Further undefined pixels in the output image are also caused by rotation, wherever the output location is due to a rotation angle that is not a multiple of 90°. In this section, two transformation enhancements are introduced which can improve the results considerably.

5.3.1 Reversed Pixel Mapping

Typically, a given transformation takes input pixel coordinates and calculates their corresponding position in the output image. Sometimes, not all output pixels get assigned values from the input pixels. In transformations such as rotation, which is a one-to-one mapping in reals but not in integers, we can use *Reversed Pixel Mapping* to correct this phenomenon. In reversed pixel mapping, we traverse each output pixel, and calculate the corresponding input pixel coordinates. This scheme ensures that all output pixels will acquire a new transformed value.

For example, to perform a magnification on a 256 x 256 pixel frame, we can get the data value at output pixel (127,127), from value at the input pixel:

$$x = \text{ROUND} \left(\frac{127}{S} \right) \quad \text{and} \quad y = \text{ROUND} \left(\frac{127}{S} \right)$$

where S is the scaling factor

We use the ROUND function instead of the INT function because INT takes the nearest left pixel as the estimated input location for data value, while ROUND gets the nearest input location instead, and so we have a better estimation of the undefined locations. S has to be a positive number and if $S < 1$, the operation results in a shrinking operation.

The same operation can be applied to image rotation. Mathematically, all transformation equations can be expressed as multiplication of a matrix and a vector where the matrix M characterizes the type of operation to be performed, and the vector $[x \ y]$ represents a pixel from the input frame. The vector product $[x' \ y']$ below represents the output pixel location [FD90, RAD90]:

$$[x' \ y'] = M [x \ y], \text{ where,}$$

x and y are the coordinates from input frame;

x' and y' are the coordinates from output frame;

and M is the matrix representing the operation.

To perform a reverse pixel mapping, we want to compute $[x \ y]$ from $[x' \ y']$ and the multiplication would be:

$$[x \ y] = M^{-1} [x' \ y'],$$

where M^{-1} is the inverse of matrix M;

The formal pseudo-code is given in Algorithm Rotation below:

Global Input: Original pixel (x, y), Angle of Rotation β

Output: Return pixel (x',y')

Algorithm Rotation

Get β from user

$x' = -127$

$y' = -127$

While True {

$x = -y' \sin \beta + x' \cos \beta$

$y = x' \sin \beta + y' \cos \beta$

 ReplicatePixel (127+x, 127-y, 127+x', 127-y')

If $x' = 127$ {

If $y' = 127$ { return }

$y' = y' + 1$

$x' = -127$

} Else {

$x' = x' + 1$

} /* If $x' = 127$ */

} /* While True */

End

End Algorithm Rotation

Refer to 5.2.1 for other modules used.

5.3.2 Bilinear Interpolation

Bilinear interpolation is a technique used to get better estimation of data values for the output pixel when non-integer coordinates are involved. For example, in reverse mapping transformation, we do not usually get integer [x,y] for vector [x',y'] even when x' , y' are both integers. In rotation, the resulting output coordinates are integers only if the rotation angle is a multiple of 90° provided that the input coordinates are both integers. Instead of rounding the non-integral coordinates to the nearest integer, we can perform

interpolation to get an accurate estimation of z-value at the point. Pixel in the input image corresponding to a pixel in the output image will always have a value in the neighbourhood of four valid input pixel locations. **Bilinear Interpolation**, produces a continuous patch so we can evaluate $f(x,y)$ when (x,y) are real. Figure 5.9 shows a bilinear surface patch and one square of the grid.

Figure 5.6: Patch used in Bilinear Interpolation.

$$\text{Let } A = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} \quad B = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} \quad C = \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} \quad D = \begin{pmatrix} D_x \\ D_y \\ D_z \end{pmatrix}.$$

Then, if $Q(s,t)$ is the bilinear surface, where s, t are parameters with $0 \leq s, t \leq 1$,

$$A = Q(0,0), B = Q(1,0), C = Q(0,1), D = Q(1,1).$$

This yields:

$$Q(s,t) = (1-t)(1-s) \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} + (1-t)s \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} + t(1-s) \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} + st \begin{pmatrix} D_x \\ D_y \\ D_z \end{pmatrix}.$$

If s is fixed ($0 \leq s \leq 1$), $Q(s,t)$ is a straight line:

$$Q(s,t) = (1-t)((1-s)A + sB) + t((1-s)C + sD) \quad 0 \leq t \leq 1$$

Also if t is fixed, $Q(s,t)$ is a straight line as s varies $0 \leq s \leq 1$.

Figure 5.10 shows this interpolation scheme and figure 5.11 shows the result using the dice as test data. Note that in figure 5.11, rotating 45° twice is different from rotating 90° once because some information is lost during rotation.

Figure 5.7: Rotation of Dice Image with Pixel location Interpolation [BA84].

Figure 5.8: Rotation of Dice Image.

5.4 Conclusion

In this chapter, we explained how comparison and transformation can be coordinated so as to achieve surface registration, and how the *Surface Similarity Measure (SSM)* can be used as a tool to measure how well the two surfaces compare. We explained how the three *Transformation* operations, Translation, Rotation and Scaling, can be used to modify surfaces in order to maximize the similarity between the transformed surface and the surface being compared, thus facilitating the registration process.

Chapter 6

Manitoba Adaptor Registration System (MARS)

6.0 Introduction

In the previous chapters we discussed how various aspects of a system could be built for surface registration using data in range data format. This chapter describes the implementation of one such system. The Manitoba Adaptor Registration System (MARS) was developed in the University of Manitoba. MARS was developed on a Macintosh system using Synmatic C++. It has a window-based GUI with a pull-down menu. In MARS, the user can perform interactive registration by selecting appropriate items from the menu. MARS is a very comprehensive system in which we have implemented different functions described in Chapters 2, 3, 4 and 5. In this chapter, we briefly list all the functions, the subsystem that each function belongs to, and present examples and applications of each function. Where applicable, we also describe how these functions work together in MARS.

6.1 Functions Implemented

There are groups of functions implemented and used in MARS. A brief description of each group and how groups are link together will be described in this section. In every case the result of the function is described. If the data is incorrect or if the function does not terminate properly, a value of '0' is returned.

MARS-User interface

GetFile: This function creates the file handle when it is given the name of the range data file.

FilePhaseToRangeData: This function gets the Range data from the data file given the file handle.

DisplayImage: This function translates elements in a range data array to be displayed on screen.

MARS-Display Method

The following functions calculate the display value (based on the display method) for each element of the range data, and stores them in output array. These functions also affect the translation of each value to be displayed:

GetGrayLevel: This function gets the gray level value for each element.

GetContour: This function computes the contour line from the given range data.

GetBiLevelContourRegion: This function determines the contour region from the given range data.

GetShading: Given the direction of the light source, this function calculates the reflection of light for each element in the range data.

GetEdge: Given the threshold, this function detects the edge from range data.

MARS-Transformation

The following functions that perform transformation have been implemented in MARS:

SurfaceTranslation: which performs translation on a bi-directional array,

SurfaceScaling: which achieves scaling on a bi-directional array, and

SurfaceRotation: which achieves rotation on a bi-directional array.

The input to these functions is the quantities that specify how much the array must be translated, scaled or rotated respectively.

MARS-Comparison

The primary function that performs comparison is **CompareRangeData**, which takes two bi-directional arrays of the same size and find the difference on pixels taken from same position of the two arrays.

Figure 6.1 shows MARS's main menu and the submenus available. Each of these submenu functions is briefly explained in subsequence sections.

Figure 6.1: A Schematic of MARS' main menu.

6.2 Display Menu

The transformation menu is as below:

Figure 6.2: A Schematic of the Display sub-menus in MARS.

6.2.1 GetGrayLevel

Description

This procedure calculates the gray level value of each unit based on the desired number of levels, and details of a given range data files, and translates the unit to become a bi-directional array which is returned. It will call *DisplayImage* to display the result on the screen.

This function returns a bi-directional array containing the gray level values for corresponding elements in the given range data array and gray level displayed on screen if the argument passed in is valid.

Interface Notes

The syntax for calling the function is:

GetGrayLevel(width, length, aRangeData, GLevel, aGreyLevel)

The arguments of the function and their meanings are:

width (IN, PASS BY VALUE): This parameter is the width of the range data, and also the total number of columns of pixels to be displayed. This has to be an integer.

length (IN, PASS BY VALUE): This is the length of the range data, and also the total number rows of pixels to be displayed. This also has to be an integer.

aRangeData (IN, PASS BY REFERENCE): This is the array containing the range data of size *<width>* by *<length>*. This has to be an array of integer or real numbers.

GLevel (IN, PASS BY VALUE): This is number of gray levels to be extracted from the range data. This has to be an integer.

aGreyLevel (OUT, PASS BY REFERENCE): This is the array containing the gray level values of range data of size *<width>* by *<length>*. This will be an array of real numbers.

The semantics for the arguments *width*, *length*, *aRangeData* and *Glevel* will be the same throughout this chapter and will not be repeated.

Implementation

This procedure is to be called through the GUI in MARS. If the range data array contains no data, the user will be asked to provide the name of the range data file, and have range data read into the array. The user can also access range data from a file. This

is valid for all procedures in the MARS-Display and MARS-Transformation subsystems.

Thus, if the system uses the procedure `GetFile()` as below:

If (`GetFile(fileRange)` &&

`FileParseToRangeData(fileRange, width, length, aRangeData)`) {

`GetGrayLevel(width, length, aRangeData, 256, aGreyLevel)`

},

it will yield *aGreyLevel*, which will be an array containing a maximum of *<GLevel>* different sets of numbers.

6.2.2 GetContour

Description

This procedure will get the contour lines given the number of contour levels and a given range data file as a bi-directional array to be returned. It also displays the result on the screen.

The procedure returns a bi-directional array containing the *<Clevel>* number of contour lines of the given range data array as a bitmap, and the contour lines displayed on the screen if the argument passed in is valid.

Interface Notes

The syntax for invoking the procedure is:

`GetContour(width, length, aRangeData, CLevel, aContour)`

The arguments of the procedure and their semantics are:

CLevel (IN, PASS BY VALUE): The number of contour levels to be extracted from the range data. This has to be an integer.

aContour (OUT, PASS BY REFERENCE): This is the array containing the contour lines of range data of size *<width>* by *<length>*. This will be an array containing values of {0, 1}.

The semantics for the argument *Clevel* will be the same throughout this chapter.

Implementation

With data read into *fileRange*, if the procedure is invoked as below:

```
If (GetFile(fileRange) &&  
    FileParseToRangeData(fileRange, width, length, aRangeData)) {  
    GetContour(width, length, aRangeData, 3, aContour)  
},
```

the procedure will yield *aContour* which contains elements range {0, 1} where 1 indicates position of a contour line, and will have a 0 otherwise.

6.2.3 GetBiLevelContourRegion

Description

This procedure will find the contour line and fill the alternate regions. The number of levels is required and the bilevel contours of a given range data will be returned as a bi-directional array, the procedure also takes care of displaying the result on the screen.

The procedure returns a bi-directional array containing the *<Clevel>* number of bi-level contour regions of the given range data array as a bitmap, and the contour regions displayed on the screen if the argument passed in is valid.

Interface Notes

The syntax for invoking the procedure is:

```
GetBiLevelContourRegion(width, length, aRangeData, CLevel,  
                        aContourRegion)
```

The arguments of the procedure are:

CLevel (IN, PASS BY VALUE): The number of contour levels to be extracted from the range data. This has to be an integer.

aContourRegions (OUT, PASS BY REFERENCE): This is the array containing the contour regions of range data of size *<width>* by *<length>*. This will be an array containing values of {0 1}

Implementation

With data read into *fileRange*, if the procedure is invoked as below:

```
If (GetFile(fileRange) &&  
    FileParseToRangeData(fileRange, width, length, aRangeData)) {  
    GetBiLevelContourRegion(width, length, aRangeData, 4, aContourRegion)  
},
```

the procedure will yield *aContourRegion* which will contain elements range {0 1} where 1 indicates element inside a contour region, and 0 otherwise.

6.2.4 GetShading

Description

Given the direction of a light source, this procedure calculates the reflection of light for each element in the range data and returns this as a bi-directional array. It will call *DisplayImage* to display result on the screen.

The procedure returns a bi-directional array containing the shaded elements of the given range data array as bitmap and the shading displayed on screen if argument passed in is valid.

Interface Notes

The syntax for invoking the procedure is:

GetShading(width, length, aRangeData, LightDir, aShade)

The arguments of the procedure are:

LightDir (IN, PASS BY VALUE): is an array in format (x, y, z) indicating direction and altitude of light source on the range data image. x, y, z are integers.

aShade (OUT, PASS BY REFERENCE): This is the array containing the shaded range data of size <width> by <length>. This will be an array of real numbers.

Implementation

With data read into *fileRange*, if the procedure is invoked as below:

If (GetFile(fileRange) &&

FileParseToRangeData(fileRange, width, length, aRangeData)) {

aLight[0] = 1

aLight[1] = -2

aLight[2] = 0

GetShading(width, length, aRangeData, aLigth, aShade)

},

array *aGreyLevel* returned from the procedure will be an array of real numbers.

6.2.5 GetEdge

Description

This procedure will extract the edge from a range data file given the threshold, and returns the result as a bi-directional array, as well as displaying the result on the screen.

The procedure returns a bi-directional array containing the edges of the given range data array as bitmap and the edges displayed on the screen if argument passed in is valid.

Interface Notes

The syntax for invoking the procedure are:

GetEdges(width, length, aRangeData, Threshold, aEdges)

The arguments of the procedure are:

Threshold (IN, PASS BY VALUE): This parameter is the threshold value used in the calculation of the edge. This value determines how precise the detection of the edge will be, and consequently what the result is. This has to be a positive real number.

aEdges (OUT, PASS BY REFERENCE): This is the array containing the edges of range data of size *<width>* by *<length>*. This will be an array containing values of {0, 1}.

Implementation

With data read into *fileRange*, if the procedure is invoked as below:

If (GetFile(*fileRange*) &&

 FileParseToRangeData(*fileRange*, width, length, *aRangeData*)) {

 threshold = 0.5

 GetContour(width, length, *aRangeData*, threshold, *aEdges*)

 },

the procedure will yield *aEdges* containing elements range {0 1} where 1 indicates the position where an edge is detected, and 0 otherwise.

6.3 Transformation Menu

The transformation menu is as below, and each of these functions is described in this section.

Figure 6.3: The schematic of the transformation sub-menu of MARS.

6.3.1 SurfaceTranslation

Description

This procedure will translate a range data file given the translation array and returns the result as a bi-directional array. It will invoke *DisplayImage* to display the result on the screen.

The procedure returns a bi-directional array containing the translated image of the given range data array as a bitmap, and the result displays on the screen if argument passed in is valid.

Interface Notes

The syntax for invoking the procedure is:

SurfaceTranslation(width, length, aRangeData, aTran, aTranslated)

The arguments of the procedure are:

aTran (IN, PASS BY VALUE): This is the translation vector (Tx, Ty) used in the transformation. Tx and Ty must be integers.

aTranslated(OUT, PASS BY REFERENCE): This is the array containing the translated range data of size *<width>* by *<length>*.

Implementation

With data read into *fileRange*, if the procedure is invoked as below:

If (GetFile(*fileRange*) &&

FileParseToRangeData(*fileRange*, width, length, *aRangeData*)) {

aTran[0] = 10

aTran[1] = -3

SurfaceTranslation(width, length, *aRangeData*, *aTran*, *aTranslated*)

},

aTranslated will contain same elements as *aRangeData* shifted up 10 rows and 3 column to the left

6.3.2 SurfaceScaling

Description

This procedure will shrink or enlarge a range data file given the Scaling Factor and returns the result as a bi-directional array. It will call *DisplayImage* to display result on the screen.

The procedure returns a bi-directional array containing the scaled image of the given range data array as a bitmap and the result is displayed on the screen if argument passed in is valid.

Interface Notes

The syntax for invoking the procedure is:

SurfaceScaling(width, length, aRangeData, SFactor, aScaled)

The arguments of the procedure are:

SFactor (IN, PASS BY VALUE): This is the Scaling Factor used in the transformation. This number determines whether the range data is to be shrunk or enlarged. It has to be a positive integer or real number.

aScaled (OUT, PASS BY REFERENCE): This is the array containing the scaled range data of size <width> by <length>.

Implementation

If the data file is read into *fileRange*, and the procedure is invoked as below:

```
If (GetFile(fileRange) &&  
    FileParseToRangeData(fileRange, width, length, aRangeData)) {  
    SurfaceScaling(width, length, aRangeData, 4, aEnlarged)  
    SurfaceScaling(width, length, aEnlarged, .25, aShrunked)  
},
```

aEnlarge should contain an enlarged upper left quarter of range data *aRangeData* in the *<width>* by *<length>* frame, and

aShrunked should contain only the upper left quarter of *aRangeData*. All the elements outside the upper left corner of *aShrunked* are 0.

6.3.3 SurfaceRotation

Description

This procedure will rotate a range data given the angle or rotation and returns the result as a bi-directional array, it will call *DisplayImage* to display result on the screen.

The procedure returns a bi-directional array containing the rotated image of the given range data array as a bitmap, and the result will be displayed on the screen if the argument passed in is valid.

Interface Notes

The syntax for invoking the procedure is:

SurfaceRotation(width, length, aRangeData, Angle, aRotated)

The arguments of the procedure are:

Angle (IN, PASS BY VALUE): This is the degree of rotation (out of 360 degrees) used in the transformation. This number can be any real number in this range.

aRotated (OUT, PASS BY REFERENCE): This is the array containing the translated range data of size *<width>* by *<length>*.

Implementation

If the data file is read into *fileRange*, and the procedure is invoked as below:

```
If (GetFile(fileRange) &&
```

```
    FileParseToRangeData(fileRange, width, length, aRangeData)) {
```

```
    SurfaceRotation(width, length, aRangeData, 30, aRotated1)
```

```
    SurfaceRotation(width, length, aRangeData, 390, aRotated2)
```

```
    SurfaceRotation(width, length, aRangeData, -330, aRotated3)
```

```
},
```

aRotated1, aRotated2 and aRotated3 will contain same elements, rotated clockwise by 30 degrees.

6.4 Similarity Measure Menu

The similarity menu for MARS is shown below. The functions are described briefly in this section:

Figure 6.4: The similarity sub-menu for MARS.

6.4.1 CompareRangeData

Description

Given two bi-directional arrays with same dimensions, this procedure calculates the difference of each element taken from same position of the two arrays, and returns the difference as a bi-directional array or a single number.

The procedure returns a bi-directional array containing the difference of elements between two given range data arrays as bitmap, or a single number as the similarity measure.

Interface Notes

The syntax for invoking the procedure is as follows:

```
CompareRangeData(width, length, aRangeData1, aRangeData2,  
                  SSM_Ptr, bSM)
```

The arguments of the procedure are:

aRangeData1 (IN, PASS BY REFERENCE): This is the first array containing the range data of size *<width>* by *<length>*. This has to be array of integer or real numbers.

aRangeData2 (IN, PASS BY REFERENCE): This is the second array containing the range data of size same size, *<width>* by *<length>* as *<aRangeData1>*. This has to be array of integers or real numbers.

bSM (IN, PASS BY VALUE): This is a boolean used to indicate the returned comparison result, and the values of *bSM* which can be one of the following:

0: Returns a *<width>* by *<length>* array of difference of individual elements

l: Returns a single real number as Similarity Measure of two range data sets

SM_Ptr (OUT, PASS BY REFERENCE): This is an address of a bi-directional array of size *<width>* by *<length>* or real variable location if *bSM* is 0, else it should be address of a real number.

Implementation

If the procedure is invoked with the input data file in *fileRange* as follows:

```
If (GetFile(fileRange) &&  
    FileParseToRangeData(fileRange, width, length, aRangeData1) &&  
    FileParseToRangeData(fileRange, width, length, aRangeData2)) {  
    CompareRangeData(width, length, aRangeData1, aRangeData2, &aDiff, 0)  
    CompareRangeData(width, length, aRangeData1, aRangeData2, &iSM, 1)  
},
```

the procedure with the given parameters yields *aDiff* which is an array containing only 0s, and *iSM*, which will be 0.

Consider the invocation:

```
If (GetFile(fileRange) &&  
    FileParseToRangeData(fileRange, width, length, aRangeData)) {  
    SurfaceScaling(width, length, aRangeData, 4, aEnlarged)  
    SurfaceScaling(width, length, aEnlarged, .25, aShrunked)  
    CompareRangeData(width, length, aRangeData, aShrunked, &aDiff, 0)  
    DisplayImage(width, length, aRangeData)
```



```
DisplayImage(width, length, aDiff)
}.
```

With these parameters, the Upper left quarter of *aDiff* will be 0s and all elements will have same value as corresponding element in *aRangeData*.

Display of *aDiff* will be same as *aRangeData* except for the blackout upper left.

6.5 MARS Interface Functions

The functions in this section provide the information required for other procedures. Most of the required input data are collected using a dialog box (dialogBox). The data file specified by the user contains range data that has to be read into the MARS system before any other operation is done. We will explain each of these functions briefly.

Figure 6.5: The Main "File"- menu for MARS.

6.5.1 GetFile

Description

This procedure gets the filename from dialogBox and opens the file as a READ-only file. It returns a Boolean value of 1 and the file handle of the given filename if the file exists. It returns 0 when the file does not exist or if it fails to get or open the specified file. It also returns 0 if the operation is aborted using the provided "CANCEL" option.

The procedure is invoked as:

GetFile(sFileHandle)

The argument of the procedure is:

sFileHandle(OUT, PASS BY REFERENCE): which is the file handle pointing to the file opened for READ-ONLY purposes.

A typical invocation as below:

If (GetFile(fileRange) &&

FileParseToRangeData(fileRange, width, length, aRangeData)) {

DisplayImage(width, length, aRangeData)

GetGrayLevel(width, length, aRangeData, 256, aGrayLevel)

}, where,

DisplayImage will show the unprocessed range data *aRangeData*.

GetGrayLevel converts the data in *aRangeData* to the gray level format, and stores it in *aGrayLevel*. It will also display the converted data in the gray level format, showing better feature information than the unprocessed *aRangeData*.

6.5.2 FileParseToRangeData

Description

This procedure will retrieve range data from a given data file. It will ensure that the file format is valid, and that the content of the file is read into *<width>*, *<length>* and *<aRangeData>*. It returns a Boolean value of 1 if dimension of range data and the bi-directional range data array contains real numbers or integers only, and 0 if the file specified does not exist or it cannot be opened. The operation can be aborted with the provide "CANCEL" option, a 0 will be returned.

Interface Notes

The syntax for invoking the procedure is:

FileParseToRangeData(sFileHandle, width, length, aRangeData)

The arguments of the procedure are:

sFileHandle(IN, PASS BY REFERENCE): This is the file handle pointing to the file opened for READ-ONLY purposes.

aRangeData (OUT, PASS BY REFERENCE): This is the array containing the range data of size *<width>* by *<length>*. This will be array of integer or real numbers.

If the procedure is invoked as below:

```

If (FileParseToRangeData(fileRange, width, length, aRangeData) {
    DisplayImage(width, length, aRangeData)
    GetGrayLevel(width, length, aRangeData, 256, aGrayLevel)
},

```

DisplayImage will show the unprocessed range data *aRangeData* and *GetGrayLevel* shows the gray level formatted data, *aGrayLevel*.

6.5.3 DisplayImage

Description

This procedure will translate elements from the given range data option to values that can be displayed on screen. This procedure returns 0 if an invalid array is passed into it.

The syntax for invoking the procedure is:

```

    DisplayImage(sFileHandle, width, length, aRangeData)

```

The arguments of the procedure are:

sFileHandle(IN, PASS BY REFERENCE): This is the file handle pointing to the file opened for READ-ONLY purposes.

aRangeData (IN, PASS BY REFERENCE): This is the array containing the range data of size *<width>* by *<length>*. This has to be an array of integers or real numbers.

Implementation

This procedure is to be called after invoking *GetGrayLevel*, *GetContour*, *GetBiLevelContourRegion*, *GetShading* and *GetEdges*. It can be called after *GetFile* and *FileParseToRangeData* through a GUI in MARS. Examples of how this procedure can be invoked can be found in *GetFile* and *FileParseToRangeData*.

6.6 Conclusion

In this Chapter, we have briefly described how MARS has been implemented. MARS can be divided into three sub-systems: Display, Transformation and Comparison. In the Display sub-system, MARS can show the image features (information) given in the raw range data format by showing the corresponding gray level image, the underlying contours and edges, and by shading the image. In the Transformation sub-system, the data can be translated, scaled or rotated for the purposes of comparison. The transformed data can be displayed using the Display sub-system. To complete the registration process, the Similarity Measure sub-system allows the user to compare two processed or unprocessed data sets, and returns the SSM as an array or as a single number. The User can then perform surface registration interactively through a series of transformations and image comparisons.

Chapter 7

Summary and Future Research

7.0 Summary and Conclusions

In this thesis, we introduced some strategies for computing similarity measure on surfaces represented as range data. The range data representation is used because it is simple, easy to understand, and easy to manipulate. One major drawback of using range data is that it cannot be used to represent shapes with hidden parts.

Surface data may be scattered during data capturing stage. This thesis shows how to regularize them by the *Modified Quadratic Shepards' Method* into range data format. Methods of displaying 3D range images are discussed and compared. The methods we demonstrated included those involving gray level display, shading, edge extraction, contour lines and contour level drawing. As examples of the techniques, we used the images of a tooth, dice and horse head in their range data formats for demonstration purposes.

The Surface Similarity measure (SSM) between two digital surfaces is defined in terms of the difference between their corresponding points. We defined the SSM in terms of three basic definitions for correspondence namely, the: *Vertical Correspondence* which takes the difference in the z-direction, *Perpendicular Correspondence*, which measures the difference of perpendicular point taken from the opposite surface, and the *Equal*

Angle Correspondence which is the difference between two points making equal angles with the two surfaces. The averaging of distances made it possible for us to measure most surfaces including steep and fluctuating surfaces. Problems and possible solutions using different methods are discussed and distance defined based on the *Equal Angle Correspondence* is found to be the best correspondence for similarity measure between surfaces.

Finally, we summarized criteria for a registration algorithm suggested by Liang, Chiron and Sabata in [LT90, CB92, SA91]. We also discussed 3D transformation operations used in surface matching, namely: translation, scaling and rotation. Again, the dice, tooth and horse head data were used for testing and demonstration. We showed how registration is achieved by computing the transformation motion parameters. We showed that it could be done automatically by considering it as an optimization problem. Alternatively, it could be achieved by estimating the transformation parameters interactively by trial and error, and then examining the similarity measure between the surface and the transformed surface. We have implemented the second method in Manitoba Adaptor Registration System (MARS). MARS can be enhanced in the future to perform registration automation, and the transformation parameters can be calculated by optimizing the similarity measure.

References

Abbreviations

CVGIP: Computer Vision, Graphics and Image Processing

CMIP: Geometric Modeling and Image Processing

CG&A: Computer Graphics and Application

- [BAR84] A. H. Barr, "Global and Local Deformations of Solid Primitives", *Computer Graphics*, Vol. 18, No.3, July 1984, pp. 21-30.
- [BU93] R. Basri and S. Ullman, "The Alignment of Objects with Smooth Surface", *CVGIP:Image Understanding*, Vol. 57, No 3, May 1993, pp. 331-345.
- [BA84] G. A. Baxes, *Digital Image Processing - A Practical Primer*, Englewood Cliffs, N.J: Prentice Hall, 1984.
- [BB92] R. V. D. Boomgaard and R. V. Balen, "Methods for Fast Morphological Image Transforms Using Bitmapped Binary Images", *CVGIP:GMIP*, Vol. 54, No 3, May 1992, pp. 252-258.
- [BR92] L. G. Brown, "A Survey of Image Registration Techniques", Technical Report, Dept. of Computer Science, Columbia University, Jan. 1992.
- [CA90] T. A. Cass, "Feature Matching for Object Localization in the Presence of Uncertainty", *IEEE 3rd International Conference on ComputerVision*, 1990, pp. 360-364.

- [CB92] P. Gerlot-Chiron and T. Bizais, "Registration of Multimodality Medical Images Using a Region Overlap Criterion", *CVGIP*, Vol. 54, No 5, 1992, pp. 396-406.
- [CF92] R. E. Carlson and T. A. Foley, "Interpolation of Track Data with Radial Basis Methods", *Computers Mathematics Applications*, Vol. 24, No.12, 1992, pp. 27-34.
- [CG84] S. Coquillart and M. Gangnet, "Shaded Display of Digital Maps", *IEEE CG&A*, July 1984, pp. 35-42.
- [FL90] M. M. Fleck, "Multiple Widths Yield Reliable Finite Differences", *IEEE 3rd International Conference on Computer Vision*, 1990, pp. 58-61.
- [FD90] J. D. Foley and A. V. Dam and S. K. Feiner and J. F. Hughes, *Computer Graphics, Principles and Practice*, 2nd Ed, Addison Wesley, 1990.
- [FR82] R. Franke, "Scattered Data Interpolation: Tests of Some Methods", *Mathematics of Computation*, Vol. 38, No.157, Jan. 1982, pp. 181-200.
- [HO84] P. F. Hopkins, "Extraction of Information from Remotely Sensed Images", Falls Church, VA American Society of Photogrammetry, Conference Proceeding, 1984, pp. 153-161.
- [JA93] D. W. Jacobs, "2D Images of 3D Oriented Points", *IEEE:Comp. Vision & Pattern Recognition*, 1993, pp. 226-232.
- [KH90] M. J. Korsten and Z. Houkes, "The Estimation of Geometry and Motion of a Surface from Image Sequence by Means of Linearization of a Parametric Model", *CVGIP*, Vol. 50, No 6, 1990, pp. 1- 28.

- [LT90] P. Liang and J. S. Todhunter, "Representation and Recognition of Surface Shapes in Range Images: A Differential Geometry Approach", *CVGIP*, Vol. 52, No 3, 1990, pp. 78-109.
- [MW94] D. S. Meek and D. J. Walton, "Alignment of Planar Curves", *Image and Vision Computing*, Vol. 12, June 1994, pp. 305-311.
- [NI93] G. M. Nielson, "Scattered Data Modeling", *IEEE CG&A*, Vol. 13, No.1, Jan. 1993, pp. 60-70.
- [PMR91] B. Kamgar-Parsi and A. Margalit and A. Rosenfeld, "Matching General Polygon Arcs", *CVGIP:Image Understanding*, Vol. 53, No.2, Mar 1991, pp. 227-234.
- [PA82] T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, 1982.
- [PM90] P. Perona and J. Malik, "Detecting and Localizing Edges Composed of Steps, Peaks and Roofs", *IEEE 3rd International Conference on Computer Vision*, 1990, pp. 52-57.
- [PR86] K. E. Price, "Hierarchical Matching Using Relaxation", *CVGIP*, Vol. 34, 1986, pp. 66-75.
- [RA90] J. J. Rodriguez and J. K. Aggarwal, "Terrain Matching by Analysis of Aerial Images", *IEEE 3rd International Conference on Computer Vision*, 1990, pp. 677-681.
- [RAD90] D. F. Rogers and J. A. Adams, *Mathematics Elements for Computer Graphics*, 2nd Ed, McGraw-Hill, 1990.

- [SA91] B. Sabata and J. K. Aggarwal, "Estimation of Motion from a Pair of Range Images: A Review", *CVGIP:Image Understanding*, Vol. 54, No 3, Nov. 1991, pp. 309-324.
- [SP86] T. W. Sederberg and S. R. Parry, "Free-Form Deformation of Solid Geometric Models", *ACM SIGGRAPH' 86*, Vol. 20, No.4, Aug. 1986, pp.151-160.
- [WB93] D. Weinshall and R. Basri, "Distance matrix between 3D Models and 2D Images for Recognition and Classification", *IEEE:Comp. Vision & Pattern Recognition*, 1993, pp. 220-225.
- [YF86] T. Y. Young and K. S. Fu, *Handbook of Pattern Recognition and Image Processing*, Orlando: Academic Press, 1986.
- [ZE86] S. D. Zenzo, "A Note on the Gradient of a Multi-Image", *CVGIP*. Vol. 33, No 5, 1986, pp.116-125.