Virtual 3D Sculpturing

Wong Pui Yee, Janis

M. PHIL.

THE HONG KONG POLYTECHNIC UNIVERSITY

1999

Acknowledgements

The first person that I would like to thank is my supervisor, Dr. Rynson Lau. In the course of my MPhil study, he has provided me with much helpful guidance, comments and criticisms, which drive me to the success of my research and attainment. Also, I am grateful for the kindness of Dr. Qing Li in supporting my research and for his help in dealing with all the administrative procedures towards the end of my study. Further, I would like to thank the examiners for spending their precious time on examining my thesis.

In the two and a half year of the research study, I am very grateful to have the groupmates of the Computer Graphics and Media Laboratory who worked and played with me during my research life. Bor, being competent in hardware and software, provided me with helpful advice on all kinds of computer related matters. Hung is a real research person. With his extensive knowledge, he is capable of giving everyone, including myself, valuable criticisms and suggestions. I thank George for his sharing of the attitude towards research work. His humour brought happiness and drove away the boring times. I also thank Danny for his participation in my early project and Jimmy brings energetic atmosphere to the research group. Apart from them, I would like to give special thanks to Dr Lizhuang Ma of the State Key Lab of CAD&CG of the Zhejiang University. During his visit to our laboratory, he was ready to help me with the necessary background of the research topic. Also, his valuable advice provided me with a good foundation in my research direction. All in all, I feel thankful for Dr Lau's offering of the peaceful atmosphere and open environment for the research group. I am delighted to be one of the members of the group and I think we really learnt much from each other.

Besides these pleasant groupmates, I have to thank all research colleagues in rooms PQ713 and PQ720. Boris encouraged me and gave me valuable ideas for making important decisions. Ho conducted me the tricks in programming, helped me get familiar with our special programming environment in the university. Sam helped me solve mathematics problems. Also, the basketball team helped me keep myself in good condition to tackle the harsh times of research life. I really had good times and enjoyed every game with all of you.

Although there had been a lot of laughter, the MPhil research is indeed a very demanding work and there had been a lot of pressure during the period of study. I must be blessed that I have a lot of good friends keeping me in company to get through the difficult times. First of all, I would like to express my greatest gratitude to my intimate friend, Ying, who cares for me, shares her life experience and shows her never-ending support to me. Also, I really appreciate Silva for sharing and giving me very special and valuable opinions; Kitty being ready to listen to and take care of me; Miranda, my pure and sincere aunt, for giving me a good sample to support myself in keeping up with my way. With all these companions, I found myself developed a lot during the years. Also, I really enjoy the good times with the mcFamily and mcPets. And thanks for all my bear friends Sailor Liz, Furry Bear Bear, Pooh Poohs, and etc… for accompanying me every single night while I am lonely at work.

Special thanks have to be given to my sister PuiFan, brothers WaiMan and WaiLeung, who accept my bad temper and being considerate during the period of the research life and all these years we are together. Also, thanks for Bella for engaging in our family activities and enjoying every gathering with us.

Finally, to my mum WoSiu and dad KatHung. I must first apologize for making you worried every single night I came back lately from the office. I would like to express my greatest gratitude for your patience and never letting me forget that I am loved and supported. Also, I really appreciate my mum for looking after every aspect of my family and myself. She lets me enjoy her excellent care on my every sickness, with a big regret that I seldom have time to accompany her. I sincerely devote my research work to both of you and hope that you are proud to have me attaining the success.

Janis Wong
June 1999

Abstract

3D surface control and deformation have been well developed in CAD/CAM for years. However, the application of this technique in virtual reality environment is still an under-explored area. Although artistic free-form models are often used in many present manufacturing products, traditional artists and designers always find it difficult to do design work with computers. This is because most of the conventional 3D CAD systems require users to have knowledge of the basis and sometimes in-depth knowledge of graphical object representations in order for them to master the systems. And in fact, most of the artists and designers are not familiar with these systems and the underneath theory. Besides, they are unwilling to learn to use such systems because they argue that the anomalous design of the interface always restrains the inspiration of the creative design work. Virtual reality system is a solution to the problem. A virtual reality system is expected to provide natural interaction which is an important benefit to non-procedural design work.

In this dissertation, we present a method for virtual sculpturing in 3D space based on the use of the glove device for direct object/surface modelling or deformation. We labelled the method ***Virtual 3D Sculpturing***. We explore the possibility of having all the degrees of freedom that the glove device provides to attain realistic 3D sculpturing experience. The main idea of the algorithm is to make use of the data collected from the glove device to create a parametric *control hand surface*, which is basically an open-uniform bicubic B-Spline tensor product surface. An object to be deformed is mapped to the *control hand surface* by a novel *ray-projection* method. The mapping method is further enhanced with a *two-pass projection process* to increase its efficiency. By maintaining the mapping relationship between the *control hand surface* and the object, the change of hand gesture can be effectively passed to deform the object. The dissertation describes the *sculpturing transformation* in detail and provides methods to attain *sculpturing region control* over the object being deformed.

Virtual 3D Sculpturing provides a more natural algorithm and interface for surface/object deformation. In addition, this method produces well-defined data model outputs. Therefore, we believe that our method is able to improve the usefulness of the existing CAD systems.

Table of Content

List of Figures

List of Tables

List of Plates

Chapter 1

Introduction

1.1 Definitions and Requirements

Conventional sculpturing is a work of art which is produced by carving or shaping stone, wood, clay or other materials. In the computer sense, it is something that falls into the spectrum of solid/surface deformation. In a broader view, deformation falls into graphics modelling. Modelling refers to the operation of construction of a graphics database. We will use object to replace the term graphics database throughout the document. Deformation is similar to modelling and it is an operation of changing the shapes of an object by transforming its geometric attributes. Notice that as distinct from rendering, modelling and deformation are not anything related to the process of producing images from the object but to define or modify the object.

A virtual reality system is an electronic system which suggests the sense of involvement or interaction associated with the virtual reality as practised using different virtual input and output tools. Being a virtual reality system, its interface is necessary to be natural and intuitive to use. Natural implies being "free from artificiality, affectation, or constraint". A natural skill is something that born with and does not require one to learn. Intuitive can be seen as allowing one to perform the tasks without conscious attention. The system is expected to attain the level of naturalness and transparency which the interface almost becomes invisible and users can interact with the virtual environment as if he/she is really there. As a result, the user can focus more on the task, becomes completely engaged with the virtual world, and feels as if he/she is interacting with the objects directly with no intermediary. Also, for a virtual reality system, we expect the algorithm to be efficient enough to support real time interaction.

Virtual 3D Sculpturing is thus the methodology used to change the shape of an object in the virtual environment with natural operations and real time interaction similar to the way we do sculpturing in the real world.

1.2 Motivation

Deformation systems developed so far make use of traditional inputs, which rely greatly on the skilfulness of users to manipulate the system, and researchers pay little attention on the interface for interaction with the objects. Most of the methods developed concentrate on deforming free-form surfaces by modifying either the control points or the sample points one by one. These algorithms are tedious when they are applied to complicated surfaces that are composed of many control points. Besides, these deformation systems work with interface constrained by clumsy intermediary devices such as keyboards, mice, and joysticks. These devices with only one, two or three degrees of freedom are often ill-suited for the complicated modelling and deformation tasks like artistic sculpturing.

In the real world situation, we usually make changes to real objects by the hands. As we realize since birth, our primary physical connection to the world is through our hands and we perform everyday tasks mostly with them. Among the variety of the virtual input tools, the most prevalent one for *Virtual 3D Sculpturing* is the electronic glove which tracks finger joint angles in order to determine the hand gesture. This kind of devices takes advantage of all the degrees of freedom of the hand. Using a virtual hand controlled by a glove device provides a very direct, expressive and natural input for virtual reality systems. Therefore, it is generally believed to be able to support truly direct manipulation operations. Previous and ongoing researches with the glove device have focused on specific areas such as tele-robotic control, and sign-language interpretation. For virtual interaction, there has been little work to examine the full potential of the hand as an input device. Interaction with 3D computer simulations developed with gloves limited to "point, reach, and grab" interactions or similar functions. Little of the naturalness that characterises our hands has been transferred to the natural tasks such as sculpturing, or more precisely object deformation.

1.3 Objectives/Significance

The main objective of our research is to develop a virtual reality system that would provide an intuitive and natural interface for the control of object deformation in the virtual environment similar to the way that we do sculpturing in the real world. Below is the task list we set out to achieve.

➢ To investigate the possibility of full and direct use of the glove's capabilities for a more realistic control of object deformation. Notice that the system described in this dissertation is one of the few systems that uses essentially all the degrees of joint-angle freedom available in glove devices to control something other than an animated hand (or a hand-like character).

➢ To provide the ability for simultaneously controlling a set of feature points of the object in order to deform the shape of it. The technique described in this dissertation supports the modification of multiple vertices of the object at the same time with one or more fingers moving simultaneously.

➢ To develop a fast and flexible deformation technique which is able to adapt to virtual environment sculpturing needs and able to apply to different representations of objects.

➢ To produce well defined free-form objects as the sculpturing output which is useful for further manufacturing processes in automatic CAD/CAM systems.

1.4 Research Methodology

1.4.1 Literature Study

In order to attain our objectives, we have to fully understand a number of fields.

First, we studied the variety of surface representations and methods for surface interpolation, which contributed an important part for the choice of the representation and construction of the virtual *control hand surface*. Then, we studied a number of existing object deformation techniques and some direct manipulation methods that might provide ideas on sculpturing techniques. In addition, some glove-input applications were investigated to facilitate the development of intuitive interfaces.

To incorporate the *Cyber*Glove^{TM} as the input device in our project, we studied the structure of the human hand. Some of the concepts in robotic articulated rigid models and transformations techniques are also employed for interpretation of the data from the device.

1.4.2 Prototype Development

Prototype is built to implement the idea of *Virtual 3D Sculpturing* and study the performance of the algorithm. In our prototype, we have adopted the *Cyber*Glove^{TM} from Virtual Technologies for hand gesture input. By attaching a Polhemus sensor to the *Cyber*Glove^{TM}, the 3D position and orientation of the user's hand can be found. The prototype is built with C++ language along with the *Virtual*Hand^{TM} library. For rendering and display, we have employed OpenInventor and OpenGL libraries.

1.5 Outline of Our Method

To modify the shape of an object, the users may want to do it simply by flexing the hand and the fingers. Therefore, the glove device which captures the hand gesture by tracking finger bend angles is believed to be the most natural and the most expressive input device to facilitate the task of *Virtual 3D Sculpturing*. In this dissertation, we present a method for virtual sculpturing in 3D space based on the use of the *Cyber*Glove™ for direct object/surface modelling or deformation. We explore the possibility of having all the degrees of freedom that the *Cyber*Glove™ provides to attain realistic 3D sculpturing experience. The main idea of the algorithm is to make use of the data collected from the *Cyber*Glove™ to create a parametric ***control hand surface***, which is basically a bicubic B-Spline tensor product surface. An object to be deformed is mapped to the *control hand surface* by a novel ***ray-projection*** method. The mapping method is further enhanced with a ***two-pass projection process*** to increase its efficiency. By maintaining the mapping relationship between the *control hand surface* and the object, the change of hand gesture can be effectively passed to control the deformation of the object. The dissertation describes the ***sculpturing transformation*** in detail and provides methods to attain ***sculpturing region control*** over the object being deformed.

1.6 Reader's Guide

Following this introduction, we will discuss previous work that constitutes the background and context for this dissertation. Chapter 2 provides a basis understanding of the general topics and the properties of the representation of the surface chosen for the modelling of the *control hand surface*. Chapter 3 deals with the different research directions of deformation/modelling systems and looks into the diversity of glove-based applications.

Chapter 4 presents the construction of the B-Spline tensor product *control hand surface*. It begins with an exploration of the mystery of hand input. It describes how the *Cyber*Glove^{TM} captures the feature of the hand gesture and how these data are used to construct the *control hand surface*. Towards the end of the chapter, it introduces efficient numerical methods to resolve the implementation problem, which results in an improvement in the performance.

Chapter 5 describes the details of setting up the mapping between the *control hand surface* and the object to be deformed. A concept of *ray-projection* is presented in this chapter for the purpose. Together with a novel *two-pass projection* process, object vertices could be properly mapped to the *control hand surface*.

Chapter 6 describes the deformation algorithm, which we refer as the *sculpturing transformation*, in detail. It discusses methods for flexible control of the *sculpturing region*. These methods can be applied to the object being deformed.

Chapter 7 illustrates some output of our prototype and discusses the performance of the algorithm.

Chapter 8 provides suggestions some extension to further improve the work and proposes potential applications of our algorithms.

Finally, the conclusion of the thesis will be included in chapter 9.

Chapter 2

Surface Representations

This chapter presents and compares different surface representation techniques and discusses the reasons of choosing the ***open-uniform B-Spline tensor product surface*** to model the *control hand surface.*

2.1 Parametric Functions

The two most common representations of curves and surfaces in geometric modelling are implicit equations and parametric functions. Parametric curves possess a natural direction of traversal while implicit curves do not [Piegl95]. Hence, it is easy to generate ordered sequences of points along a parametric curve. Similar statement holds for generating sample points on surfaces. The parametric form is more natural for designing and shape representation in a computer and the coefficients of many parametric functions, such as, Bézier splines and B-Splines, possess considerable geometric significance. This translates into intuitive design methods and numerically stable algorithm with a distinctly geometric flavor.

In parametric form, each of the coordinates (x, y, z) of a point on the curve is represented separately as an explicit function of an independent parameter u.

$$C(u) = (x(u), y(u), z(u))$$

Thus, $C(u)$ is a vector-valued function of the independent variable u where $a \le u \le b$. The boundary of the parametric representation is explicitly defined by their parametric intervals. And this type of curves possesses geometric properties that make it a basis tool for graphics modelling. Notice that although the interval $[a,b]$ is arbitrary, it is usually normalized to $[0,1]$.

Two common parametric representations are the Bézier spline and B-Spline. They can be used to represent both analytic shapes and free-form surfaces.

2.2 Bézier Representations

This method was developed by the French engineer Pierre Bézier and had used for the design of Renault automobile bodies. Bézier splines possess properties that make them highly useful and convenient for curve and surface design. They are also easy to implement. For these reasons, Bézier splines are widely available in various CAD systems, in general graphics packages (such as OpenGL), and in assorted drawing and painting packages.

In general, a pth-degree Bézier curve is defined by

$$C(u) = \sum_{i=0}^{p} B_{i,p}(u) P_i \qquad \text{for } 0 \le u \le 1 \tag{2-1}$$

The geometric coefficients of this form, P_i with $i = 0,\dots,p$, are called control points. The basis (blending) functions, $B_{i,p}(u)$, are the classical nth-degree Bernstein polynomials given by

$$B_{i,p}(u) = \frac{p!}{i!(p-i)!} u^i (1-u)^{n-i} \tag{2-2}$$

As mentioned above, Bézier curves possess some geometric properties that make it easy for design. Other properties can be found in [Farin97,Piegl95].

1. **Endpoints Interpolation**: their endpoints intersect with the first and last control points; $C(0) = P_0$ and $C(1) = P_p$.

2. **Convex Hull property**: the curves are contained in the convex hulls of their defining control points.

3. **Transformation Invariance**: rotations, translations, and scaling can be applied to the curve by applying them to the control points.

4. **Variation Diminishing property**: no straight line intersects a curve more times than it intersects the curve's control polygon. Thus, the control polygons approximate the shapes of the curve.

5. The *k*th derivative at u=0 (u=1) depends on the first (last) $k+1$ control points. In particular, their endpoint tangent directions, $C'(0)$ and $C'(1)$, are parallel to the line connecting the first and second control points, $P_1 - P_0$, and the line connecting the second last control point and the last control point, $P_p - P_{p-1}$, respectively.

6. **Global Control**: In moving one of the control points, the shape of the whole curve is affected.

7. The **number of control point** equals to the order of a Bézier curve (i.e. p+1) where p is the degree of the curve.

We can see from the above that the Bézier curve can provide a powerful tools in curve design, but still, it has some limitations. If the curve to be modelled has a complex shape, its Bézier representation will have a prohibitively high degree. Such complex curves can, however, be modelled using composite Bézier curves. These piecewise polynomial curves are also called B-Spline curves.

2.3 B-Spline Representations

B-Spline representations are the most widely used class of approximating splines. Precisely, a B-Spline curve is a combination of a number of Bézier curves which the connecting points are automatically maintained at C^2 continuity. It can model complex shapes easily.

A B-Spline curve is a vector-valued piecewise polynomial function and a *p*th-degree B-Spline curve is defined as

$$C(u) = \sum_{k=0}^{n} N_{k,p}(u) P_k \tag{2-3}$$

where P_k, $k = 0,1,\ldots,n$ are Euclidean control points. $N_{k,p}(u)$ are the normalized B-Spline basis functions of order $p+1$, or degree p. The B-Spline curve is defined over the knot vector $U = \{u_0, u, \ldots, u_m\}$, a sequence of nondecreasing real numbers. That is, we can choose any values for the subinterval endpoints (knots) if they satisfying the relation $u_j \le u_{j+1}$. A Bézier curve is in fact a special case of B-Spline curves such that the knot vector of a Bézier curve is defined as:

$$U = \{\underbrace{0,\ldots,0}_{p+1},\underbrace{1,\ldots,1}_{p+1}\} \tag{2-4}$$

The B-Spline basis functions $N_{k,p}$ are polynomials of degree *p*, and different from Bézier representation, they also depend on the knot values in the knot vector (Figure 2-1). Basis functions for B-Spline curves are defined by the Cox-deBoor recursion formulas:

$$N_{k,p}(u) = \begin{cases} 1 & \text{for } (u_k \le u < u_{k+1}) \\ 0 & \text{otherwise} \end{cases} \quad \text{for } p = 1 \tag{2-5}$$

$$N_{k,p}(u) = \frac{u - u_k}{u_{k+p-1} - u_k} N_{k,p-1}(u) + \frac{u_{k+p} - u}{u_{k+p} - u_{k+1}} N_{k+1,p-1}(u)$$

$$\text{for } p > 1 \qquad (2\text{-}6)$$

Note: Since it is possible to choose the knot values of the knot vector so that the denominators in the previous calculations can have a value of 0, this formulation assumes that any terms evaluated as $0/0$ are to be assigned the value 0.

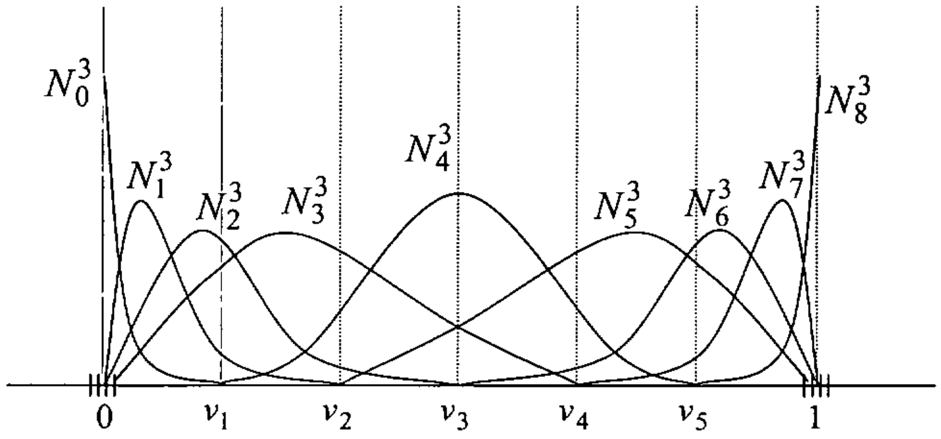


Figure 2-1 B-Spline Basis (blending) Functions.

Generally, B-Splines have similar properties as Bézier splines. In addition, they possess the following two main advantages over Bézier splines.

1. **Local Support Property**

B-Splines allow local control over the shape of the spline function. This can be derived from the property of the B-Spline basis functions that $N_{k,p}(u) = 0$ if the parameter u is outside the interval $[u_k, u_{k+p+1})$. Local control for B-Splines is achieved by defining the basis functions over subintervals of the total range of u. Precisely, each of them is defined over $p+1$ subintervals of the total range of u. This implies a local control of each of the control points, P_k. In other words, moving

P_k changes $C(u)$ only within the interval $[u_k, u_{k+p+1})$.

2. **Relaxation of the Dependency of Degree on the Number of Control Points**

   The values of the basis functions depend on the number of control points we select, the degree of the curve, and how we set up the subintervals (knots) of the knot vector. The degree, p, number of control points, $n+1$, and the number of knots, $m+1$, are related by

   $$m = n + p + 1 \tag{2-7}$$

   Therefore, B-Splines allow us to vary the number of control points used to design a curve without changing the degree of the polynomial. Also, any number of control points can be added to modify or manipulate the shape of the curve. Similarly, we can increase the number of knots in the knot vector to aid curve design. Notice that increasing the number of knots in the knot vector will require the addition of control points, $n+1$, since the size of the knot vector, $m+1$, depends on n.

The difference between B-Splines and Bézier splines is that the range of parameter u now depends on how we choose the knot vector, U. If the knot vector of an order k B-Spline curve has k multiple end knots, it is called an *open-uniform B-Spline.* This is a combination of *Bézier spline* and *B-Spline.* A knot vector of an *open-uniform B-Spline* has the form

$$U = \{\underbrace{0,\ldots,0}_{p+1}, \underbrace{u_{p+1},\ldots,u_{r-p-1}}_{r+1}, \underbrace{1,\ldots,1}_{p+1}\} \quad \text{where } r = n + p + 1 \tag{2-8}$$

In addition to the "local support properties" and "the relaxation of the dependency of degree on the number of control points", *open-uniform B-Spline* characterize in interpolating the end control points with the data points. Thus, it is able to give models with explicit boundaries. Moreover, these splines are tangential to the control polygon at the end points. This property further eases the task of interpolation. Therefore, *open-uniform B-Spline* is chosen for the approximation of the *control hand surface* in our implementations.

2.4 Tensor Product Surfaces

A tensor product surface is a vector-valued function of two parameters, u and v. It represents a mapping of a parametric region, $\Re$, of the uv plane into Euclidean 3D space. It has the form

$$S(u,v) = (x(u,v), y(u,v), z(u,v)) \qquad (u,v) \in \Re$$

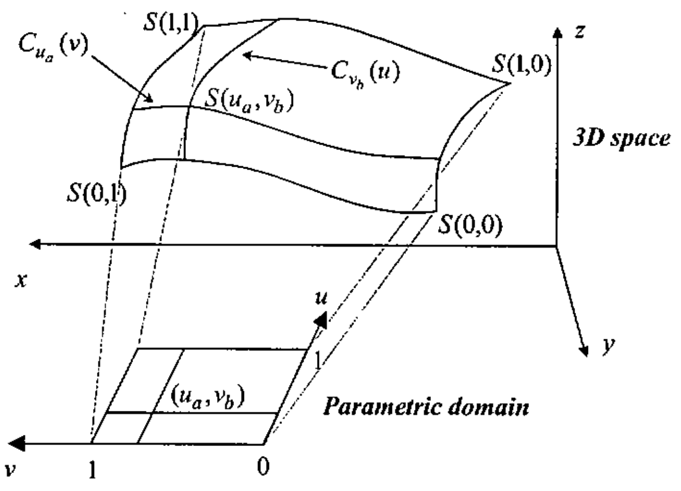


Figure 2-2 Tensor Product Surface.

The simplest and the most widely used surface representation in geometric modelling applications are the tensor product surface (Figure 2-2). Tensor product surfaces are developed based on the concept of bilinear interpolation. While linear interpolation fits the "simplest" curve between two points, bilinear interpolation fits the "simplest" surface between four points. It is basically a bi-directional curve scheme making use of basis functions and geometric coefficients of control points.

Instead of depending on a univariate in parametric curve system, the basis functions of a tensor product surface are bivariate functions of u and v. A tensor product surface is constructed as the product of univariate basis functions.

An open-uniform B-Spline tensor product surface can be obtained by taking a bi-directional $(n+1)\times(m+1)$ net of control points, $P_{i,j}$, and the products of the univariate B-Spline basis functions, $N^u_{i,p}(u)N^v_{j,q}(v)$.

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}N^u_{i,p}(u)N^v_{j,q}(v)P_{i,j} \tag{2-9}$$

where $0\le u,v\le 1$ and the parametric domain (u,v) of this mapping is a square. There are $r+1$ and $s+1$ internal knots in the knot vectors U and V in 2D respectively. The knot vectors are given by

$$U=\{\underbrace{0,\ldots,0}_{p+1},\underbrace{u_{p+1},\ldots,u_{r-p-1}}_{r+1},\underbrace{1,\ldots,1}_{p+1}\} \qquad \text{where } r=n+p+1$$

$$V=\{\underbrace{0,\ldots,0}_{q+1},\underbrace{u_{q+1},\ldots,u_{s-q-1}}_{s+1},\underbrace{1,\ldots,1}_{q+1}\} \qquad \text{where } s=m+q+1$$

A tensor product surface possesses a useful geometric property. At fixed $u=u_a$,

$$C_{u_a}(v)=S(u_a,v)=\sum_{j=0}^{m}\left(\sum_{i=0}^{n}N^u_{i,p}(u_a)P_{i,j}\right)N^v_{j,q}(v) \tag{2-10}$$

$C_{u_a}(v)$ is a power basis curve lying on the surface, $S(u,v)$ with basis functions $\sum_{i=0}^{n}N^u_{i,p}(u_a)P_{i,j}$. Similarly, at fixed $v=v_b$, $C_{v_b}(u)$ is a power basis curve lying on $S(u,v)$ with basis functions $\sum_{j=0}^{m} N_{j,q}^{v}(v_b)P_{i,j}$. The curves $C_{u_a}(v)$ and $C_{v_b}(u)$ intersect at the surface point, $S(u_a, v_b)$. These curves are called ***isoparametric curves*** (or ***isocurves***) (Figure 2-2). We will discuss how this property facilitates the *control hand surface* interpolation process in chapter 4.

Chapter 3

Object/Surface Deformation Systems

As already discussed in chapter 1, many modelling/deformation systems have been developed. In this chapter, we will take a look at some of the existing solid modelling systems and solid/surface deformation algorithms. Then, we explore the current research directions in the application of the glove devices.

3.1 Solid Modelling Systems

3.1.1 Constructive Solid Geometry (CSG)

JDCAD [Liang93] is a system built on the MR Toolkit [Shaw93]. It introduces interaction techniques based on the use of the keyboard and a hand-held bat which is a six-degree-of-freedom 3D magnetic tracking device together with the keyboard. Their work aimed at providing a better interface - a set of 3D manipulation mechanisms like 3D menu pop-up and spotlight selection, and some navigation techniques. Unlike the work described in this paper, the system is for constructing mechanical components by creating and assembling primitive volumes instead of for free-form geometric modelling.

3.1.2 Spatial Approach (Positive/Negative Volume)

Kurmann in [Kurmann96] proposed a system for interactive spatial modelling to support architectural design in a VR environment. By using the mouse or other 3D input devices, the user may interact with the system in a 3D environment. This enables the user to formulate design ideas in a 3D space. Similar to JDCAD, this system contains additional predefined building elements or even furniture for assembling in the architectural design. They introduce the concept of positive (solid) and negative (space) volumes to spatial modelling to meet the special design concept of architecture.

This method is useful for outlining vague ideas in the conceptual abstract design phase of architectural design where conceptual decisions are made and major constraints are established. However, it is not suitable for artistic free-form deformation and in fact, little of the potential of the 3D input devices was explored in this system.

3.2 Deformation Algorithms

The systems discussed so far concern mainly scene assembling. We now concentrate on the systems developed for free-form geometric shape modification. The term "free-form" refers to the situation that objects can be reshaped flexibly, and geometric shape modification is about the techniques for changing the shape of an object under some restriction related to the properties of an object (e.g. continuity). These systems produce free-form object reshape instead of scene assembling.

3.2.1 Free Form Deformations

The most well known method of object deformation is the **Free-Form Deformation (FFD).** It was first developed by Sederberg and Parry [Sederberg86]. Although this technique has been developed for quite some time, object deformation still continues to be a major research area in CAD/CAM and computer graphics. Free-form deformation is a powerful modelling technique that enables the deformation of objects by deforming the space around them. This is a method for deforming solid geometric models in a free-form manner. It can deform surface primitives of any type or degree.

This method consists of embedding the geometric object or the region of the object to be deformed into a regularly subdivided parallelepipedical 3D parametric solid lattice (Figure 3-1). Each of the dimensions of the lattice is defined by cubic Bernstein polynomials and the volume of the lattice is a tensor product piecewise tricubic Bernstein polynomials. Each sample point of the object in 3D space can be mapped to a parametric coordinate set. The deformation of the FFD lattice is induced by modifying its control points and the deformation is automatically passed to the embedded model.

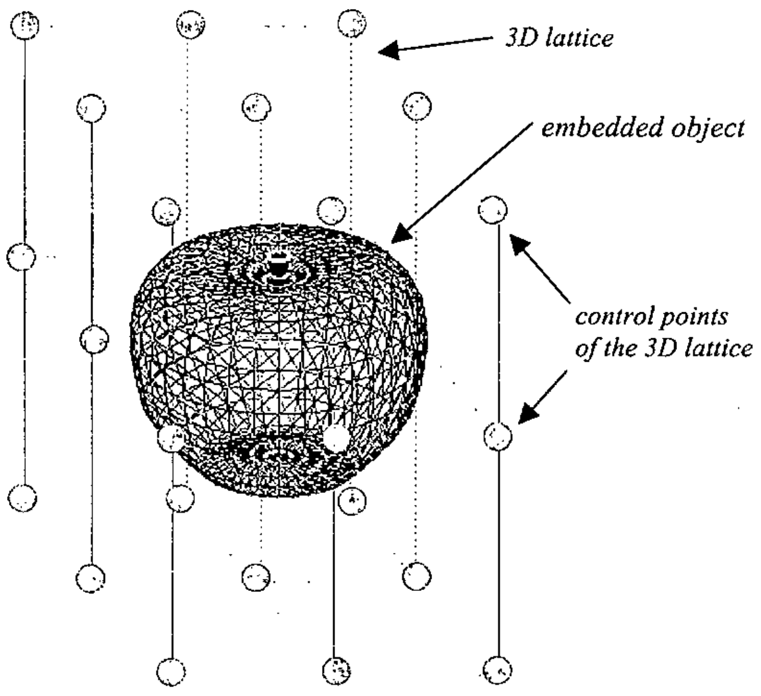


Figure 3-1 Free-Form Deformations (FFDs).

The original FFD formulation proposed by Sederberg and Parry restricted the parametric solid to a regular parallelepiped with uniform divisions and a Bernstein polynomial basis. The degree of parameterization of the solid in each parametric coordinate was set directly by the number of uniform divisions within the control lattice - two for a quadratic parameterization, three for a cubic, and so on. Bernstein-based formulations yield an unfortunate relation between lattice divisions and the degree of the enclosing parametric solid. Lattices requiring more than two or three divisions for flexibility are undesirable, since evaluating the basis becomes computationally more expensive as the degrees of the polynomials increase.

After the work of Sederberg and Parry, there are a number of enhancements of the method proposed. **Extended Free-Form Deformation (EFFD)** [Coquillart90] does not restrict the enclosing solid to a parallelepiped. An EFFD lattice is defined either from a predefined 3D lattice or from 2D lattices. Predefined EFFD lattices are obtained by the transformation of the parallelepiped lattice described in original FFDs

2D lattices can be formed by making loft of a surface intersecting on the object surface. Also, 2D lattices can be welded together in order to form a composite 2D lattice. In general, arbitrarily shaped lattices could be made by composing elementary lattices. However, extra effort is needed to maintain the continuity between these lattices.

With EFFDs, the user configures the initial lattice of control points to the approximate shape of the intended deformation instead of starting with the FFD's regular parallelepiped of control points. The user must know the general shape of the deformed object before starting to model, and the interface of the system is only a direct representation of the underlying mathematics.

Nurbs-based Free-Form Deformations (NFFDs) [Lamousin94] replace the Bézier representation with Non-uniform Rational B-Splines (NURBS). Therefore, NFFDs can provide 3D lattices which have non-uniform subdivisions. The lattices can be divided coarser in areas requiring little or no deformation and finer in areas requiring substantial deformation. The resultant lattice is non-uniform subdivided and thus different weightings can be given to different regions of the model to create different deformation effects. As a result, additional control may be achieved in finer divided regions.

All FFDs discussed so far concern about manipulating control points on the 3D lattices are not intuitive to use. They are only suitable for expert users who have knowledge in parametric object modelling. **Direct Manipulation of FFDs** [Hsu92], on the other hand, allows the user to move the sample points on the object model and compute the necessary alteration to the control points of the FFDs. However, it involves the computation of least square which is very computationally expensive. Also, the problem could be very complex when more than one vertex point is to be moved at the same time, and the solution may not be unique. Similar to other FFDs, the technique only tackles the modification of object points one by one.

The FFDs technique has been implemented with an elastic input device that has a tactile feedback [Murakami94]. The elastic input device has the same shape as the control volume (a cube). By deforming the input device, the 3D model embedded will be deformed accordingly. Although the input device provides a better interface to the FFD methods, the nature of the underneath algorithm still hinders the naturalness of the deformation interaction.

3.2.2 Grid Surface Modelling

Similar to Murakami, Kameyama designed a rubber pad with tactile sensor sheet [Kameyama97] which can detect user's shape deforming actions. The system creates grid surface data and provides a method for converting the input shape data into solid model data. The method emphasizes how the output solid model can be directly utilized in other engineering analysis or manufacturing processes and also the collaborations with other design system.

A major limitation of the method is that it requires that the initial shape of the surface to be deformed to be the same as that of the tactile sensor sheet, i.e. flat. Moreover, due to the nature of the input device, only vertical deformation of the surface gird is allowed. These limitations imply that the system is not suitable for highly complex modelling tasks like 3D sculpturing.

3.2.3 Volumetric Approach

Galyean and Hughes developed a technique for virtual sculpturing with a 3D tracker [Galyean91]. It is basically a 3D painting system. The material is represented by voxel data and the method works by modifying the values in the voxel array, similar to the paint brush function of a "paint" program that modifies bitmap values. The fundamental notion is to describe the shape of a piece of clay by its characteristic function, whose value is 1.0 at any point in space where there is clay and 0.0 elsewhere. The system modifies the values of volumetric data by moving a 3D tool through space. And modifying the shape of the clay is equivalent to modifying the characteristic function of the voxel array. Direct manipulation of a volumetric representation is achieved by chiselling away parts of the volume with a Polhemus-based tool. The resultant voxel data is converted to a polygonal surface using a "marching-cubes" algorithm. The models created are said to be free-form and may have complex topology. The method is was useful for controlling the shape by modifying a point or a small part of an object. However, they are not useful for global deformations and also it is not precise.

3.2.4 Nurbs Model Creation System

Li [Li97] developed a prototype for creating and editing of Nurbs surfaces. Other than developing a deformation system, their main theme is to provide a fast rendering technique for parametric surfaces.

3.3 Glove-based Applications

3.3.1 Gesture Recognition

Conventional computer interaction depends on the use of keyboards, mice, and joysticks. These devices with only one, two or three degrees of freedom are often ill-suited for complicated tasks. These systems are clumsy and require special training for users. With the glove device, event triggering may require the recognition of the user's hand gesture. Different approaches had been implemented for the purpose. The Grimes Digital Data Entry Glove [Grimes83] recognizes hand signs from simple finger spelling postures. It relies on custom hardware circuits. More sophisticated applications of the interpretation of signed languages are for computer input and for human communication. James Kramer developed the *Cyber*Glove™ as part of his work to translate American Sign Language (ASL) into spoken English [Kramer89]. It makes use of a Bayesian decision rule-based pattern recognition scheme to map finger positions into predefined letters or symbols.

Glove devices are also used to control acoustic parameters in live musical performance [Morita91]. The attitude and posture of the hand are interpreted to determine the commands of musical expression.

3.3.2 Teleoperation and Robotic Control

Glove interfaces in teleoperation and robotic control are important for facile, dexterous control of the remote end. Research projects have used the glove device to control a dexterous robot hand. To adapt glove devices to robotic control, Pao constructed algebraic transformation matrices to map human hand poses to robot hand poses [Pao89]. [Hong89] resolved the kinematic differences between the human hand and the robotic hand by determining the positions of the user's fingertips and driving the robot hand fingertip positions to match.

Most of the teleoperation systems make use of gesture recognition to interpret commands. Brooks used a neural net to interpret DataGlove motion for robot control [Brooks89]. Sturman implemented a gesture recognition system to simulate the operation of a construction crane with hand signal conventionally used on construction sites. Also, they use the same system for the simulation of a six-legged robot. The robot's entire interface, including locomotion, point of view, manipulator control, and mode selection, was glove-based [Sturman92].

3.3.3 Natural Interface – "Point, reach, and grab"

Since we manipulate physical objects most naturally with our hands, there is a great desire to apply the skills, dexterity, and naturalness of the hand directly to the human-computer interface.

The DataGloves from VPL uses the hand as the user's manipulative extension into the virtual environment. By pantomiming reaches and grabs, the user causes the hand to reach and grab objects in the virtual environment. The user can move through the virtual space by pointing in the desired direction [Hand97].

The virtual hand is displayed in an interactive computer environment and used as a tool for "point, reach, and grab" interaction. Also, the systems provide event triggers similar to buttons in mice by the finger posture and movement recognition.

The advantage of this model of interaction is naturalness – users' actions correlate closely with those that might be performed on physical objects. However, we see that the glove is somehow functioned as little more than a 3D joystick with several buttons. Little of the dexterity and naturalness that characterise our hands have been transferred to the computer tasks.

3.3.4 Computer-based Puppetry

This is a method of tracking the live motion of the human hand or even the human body in order to inject life into computer animation. This method overcomes the limitation of the conventional computer animation of characters. Conventional computer animation uses key-frame technique with interpolation which gives the result a subtly unnatural quality, not quite mechanical, but not quite living. Computer-based puppetry is done by putting a performer in direct interactive control of a character, as in puppetry, or capturing body motion for later application to animation, translates the nuances of natural motion to computer characters, making them seem very much alive. This is also one of the very few applications which uses all the degree of freedom of a sensor glove for the purpose.

Chapter 4 The Control Hand Surface

In the following subsections, we will describe the construction of the *control hand surface* in detail. We begin by providing a brief introduction on the feature and the properties of the human hand. Then we discuss the virtual input devices that we have employed in the prototype. We will also give a detailed description of the *Cyber*Glove^{TM}, which is the glove device we have adopted in the prototype. Finally, we explain the definitions, method for setting up, and the techniques for enhancing the efficiency of constructing the *control hand surface.*

4.1 Human Hand Input

Human hand input is the most direct use of the hand capabilities for the control of computer mediated tasks such as 3D sculpturing. The human hand shows an extraordinary degree of primitiveness. Its specialized movements, acute sensitivity, precision, subtlety and expressiveness are not easy to model. Hand gesture forms a notable part of our day-to-day lives. According to Sir Richard Paget (1869-1955), human can make 700,000 different hand gestures. Figure 4-1 shows the muscle and tendons of the back of the human hand. The many interconnections and interactions of the muscles and tendons of the hand give rise to the complexity of hand motion.

Most of the muscles lie in the forearm with long tendons transmitting poser to the fingers. This arrangement allows the hand to be light and flexible without sacrificing strength.

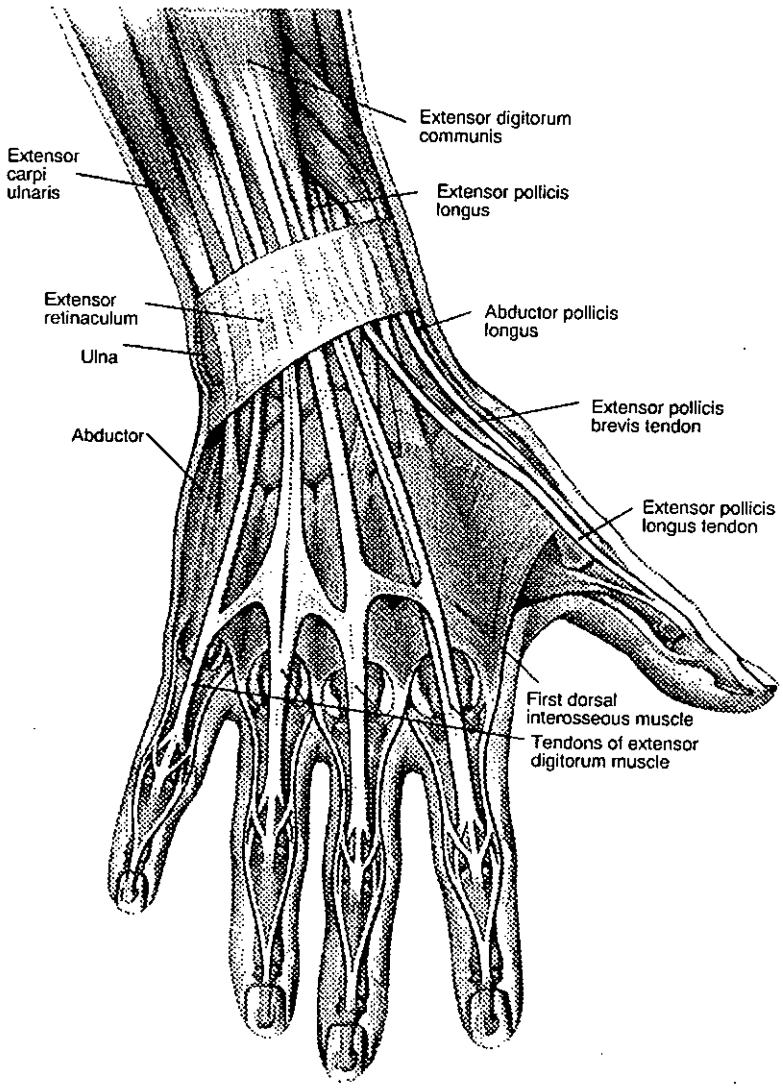

Extensor digitorum communis
Extensor carpi ulnaris
Extensor pollicis longus
Extensor retinaculum
Abductor pollicis longus
Ulna
Extensor pollicis brevis tendon
Abductor
Extensor pollicis longus tendon
First dorsal interosseous muscle
Tendons of extensor digitorum muscle

Figure 4-1 Muscles and Tendons of the Back of a Human Hand. (Diagram adapted from *Hand and wrist*, a wall chart by the Anatomical Chart Co., Stokoe, IL, 1988)

At a functional level, human hand input is the information that a computer derives from the monitoring of the individual degrees of freedom of the hand. In the fullest sense of the term, this input involves 29 degrees of freedom of the hand: 23 degrees of freedom in the hand joints above the wrist, and 6 degrees of freedom in the free motion of the palm (derived from the wrist, forearm, elbow, and shoulder joint motions). Figure 4-2 shows the skeleton and joints of a human hand. There are 17 active joints in the hand, together providing 23 degrees of freedom. The third degree of freedom of the trapeziometacarpal joint (base of thumb) allows the thumb to rotate longitudinally as it is brought into opposition with the fingers. This rotation is dependent on the degrees of freedom of the thumb. Thus, it might be said that it is not a true degree of freedom and that the hand joints only embody 22 degrees of freedom. By using a glove device, the information in the feature joint positions could be recorded and we are thus capable of capturing or reproducing the hand gesture as digital input.

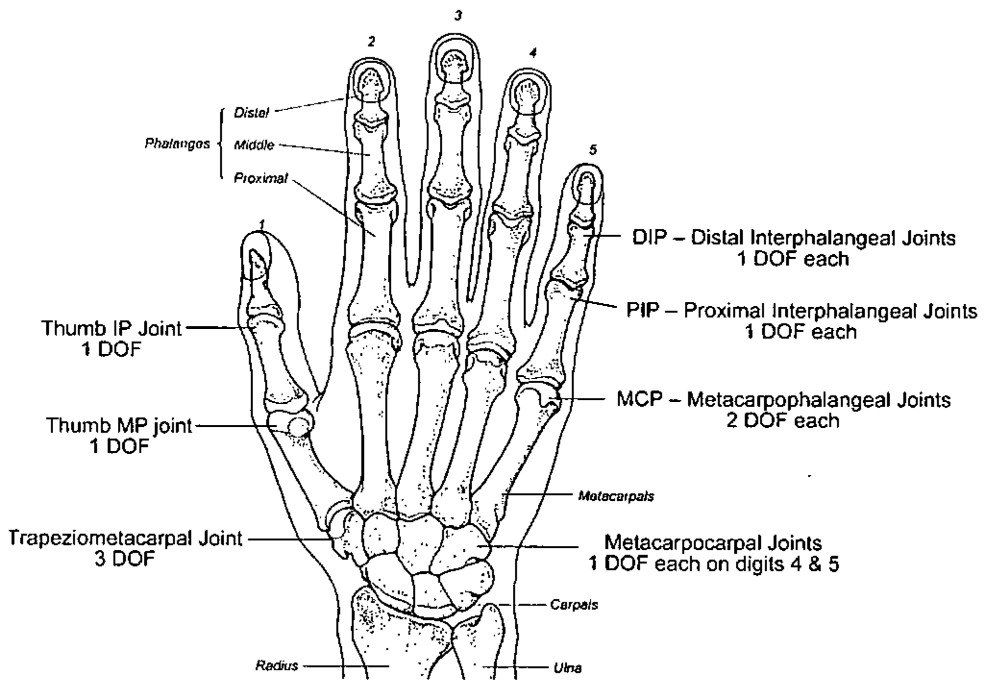

3
2
4
5
1
Phalanges
Distal
Middle
Proximal
DIP – Distal Interphalangeal Joints 1 DOF each
PIP – Proximal Interphalangeal Joints 1 DOF each
Thumb IP Joint 1 DOF
MCP – Metacarpophalangeal Joints 2 DOF each
Thumb MP joint 1 DOF
Metacarpals
Trapeziometacarpal Joint 3 DOF
Metacarpocarpal Joints 1 DOF each on digits 4 & 5
Carpals
Radius
Ulna

Figure 4-2 Anatomical Definitions of the Human Hand.

(Diagram adapted from *Hands*, Pantheon Books, New York, 1980.)

At the other end of the spectrum, human hand input may be as simple as monitoring the 3D position and orientation of the palm or the bends of three or four fingers. The distinguishing characteristic of the human hand input is that the user does not think in terms of manipulating an input device, as is the case with other haptic forms of input (e.g., mouse, joystick, and trackball), but moving his/her hand to directly affect the task. A functional way to describe the distinction is that human hand input is derived from direct measurement of hand motion rather than measurement of the motion of a device manipulated by the hand.

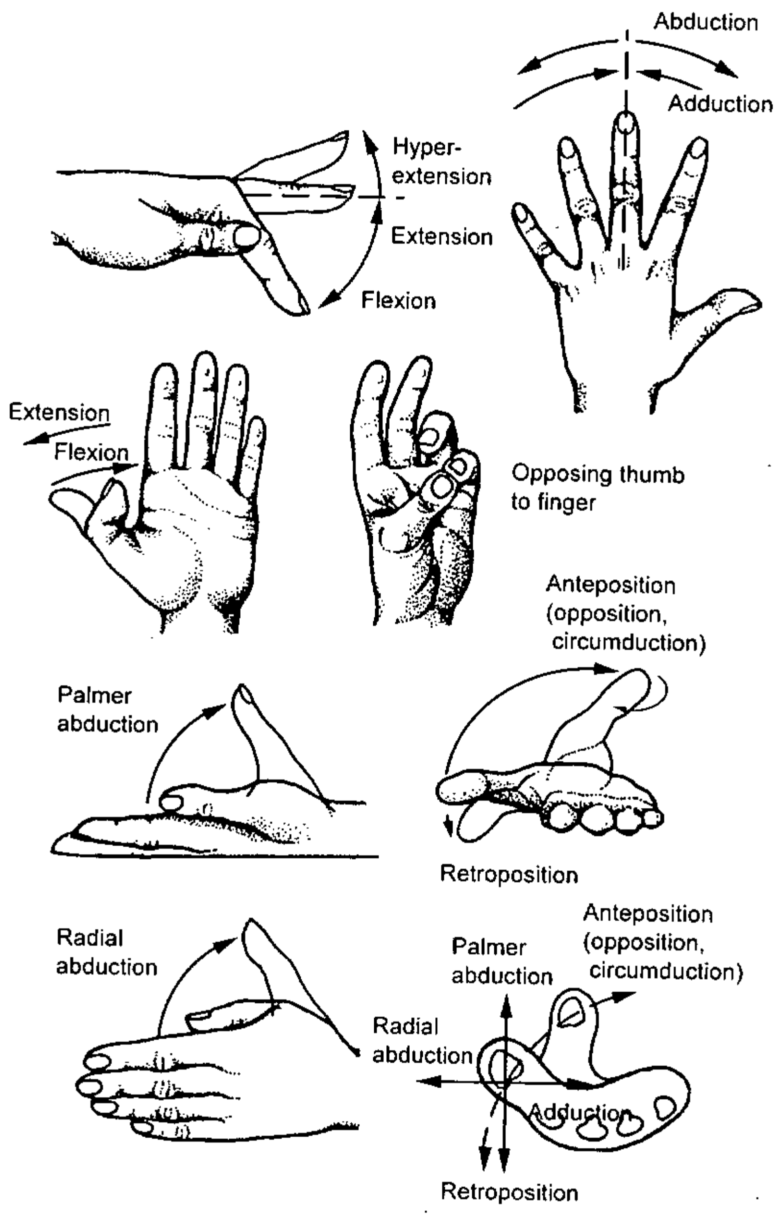


Figure 4-3 Motion of the Hand Joints.
(Diagram adapted from American Society for Surgery of the Hand, 1978.)

Terms used to describe the motions of the fingers are well established. The human fingers have flexion-extension, lateral abduction-adduction Additionally, the thumb has an antesposition-retroposition motion which brings it in opposition to the palm. Figure 4-3 illustrates each of them and shows some of the possible hand gestures produced.

4.2 *Virtual Input Tools*

Most input tools fall into one of the two categories: **interaction devices** and **tracking devices**. Interaction devices provide access to the virtual world. With these devices, the user may move and manipulate objects at will. Tracking devices, on the other hand, passively monitor various body parts to create a sense of presence or affordance, and consequently provoke the feeling of being physically present in the virtual world. In most of the virtual reality systems, both types of devices are required and this is the same for ours.

4.2.1 *Interaction devices – the Glove Device*

As we have seen from section 4.1, the human hand is very expressive and is highly complex. However, with conventional input devices such as force balls, 3D mice and 3D probes, it is always not easy to capture the features of the human hand although these devices possess the advantages of simplicity and easy to use. This is mainly because they have limited degrees of freedom. Therefore, if we employ these devices in a virtual reality system, the natural motion of the user's hand is sacrificed. As a result, interaction with the virtual world needs anomalous algorithms to map intuitive tasks to the limited interactions provided by these devices. This kind of input tools is thus not suitable for 3D sculpturing which requires the effect of the highly expressive hand gesture to pass to the virtual object to be deformed. In order to facilitate such gesture-based interactions with the virtual world, it is desirable to allow additional degrees of freedom by sensing individual finger motions using the glove device. Glove devices are believed to be the most natural interactive input tools for which we can use for object modification/deformation.

The glove device is basically an instrument which could be worn on the user's hand and it consists of sensors for measuring the movements of fingers and hand. Commercial models include VPL's DataGlove, Virtex's *Cyber*Glove™, Mattel's PowerGlove and Exos Dextrous Hand Master [Struman94]. They all have sensors that measure some or all the finger joint angles. Some of them have 3D sensors as well in order to track the user's wrist motion. In our implementation, we have adopted the *Cyber*Glove™ of Virtual Technologies as our interaction device, although extension of our method to other glove devices is expected to be straightforward. The *Cyber*Glove™ is a powerful tools for one to interact within the virtual world and input data. By employing the *Cyber*Glove™, the feature gesture of the hand can be captured effectively.

*4.2.1.1 The Cyber*Glove™ *Device*

The *Cyber*Glove™ was invented by Jim Kramer. It makes use of bend sensor at each joint of the hand. Unlike VPL's DataGlove, which uses optical fibre, it uses linear sensors; this reduces the nonlinear sensing problem of the DataGlove (Figure 4-4).

The *Cyber*Glove™ uses bending sensing technology to detect the "posture" of the hand. It consists of thin electrical strain gauges placed on elastic nylon blend material. The electrical strain gauges are placed in pairs at each of the joints to be measured as shown in Figure 4-5a. The palm and the fingertip area are removed for better ventilation and to allow normal activities such as typing and writing. As a result, the glove is very light and easy to wear.

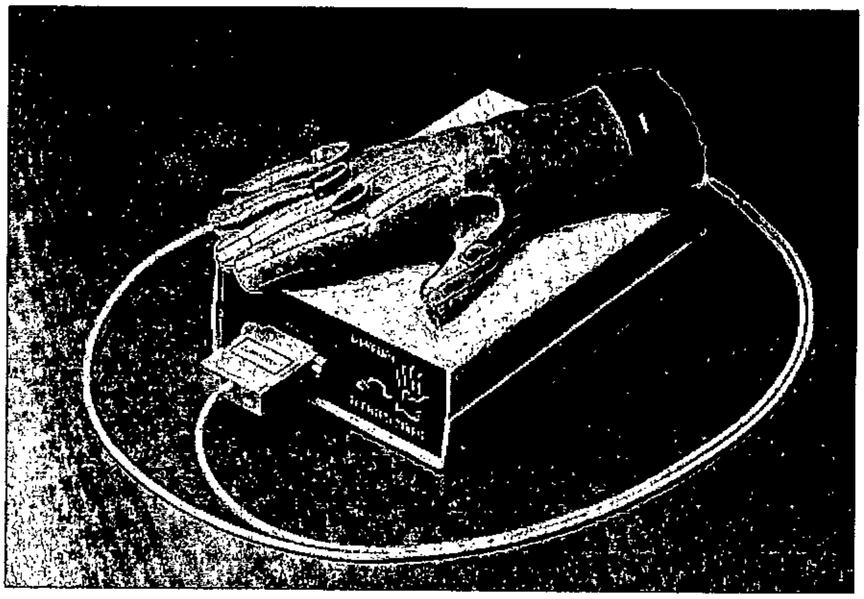
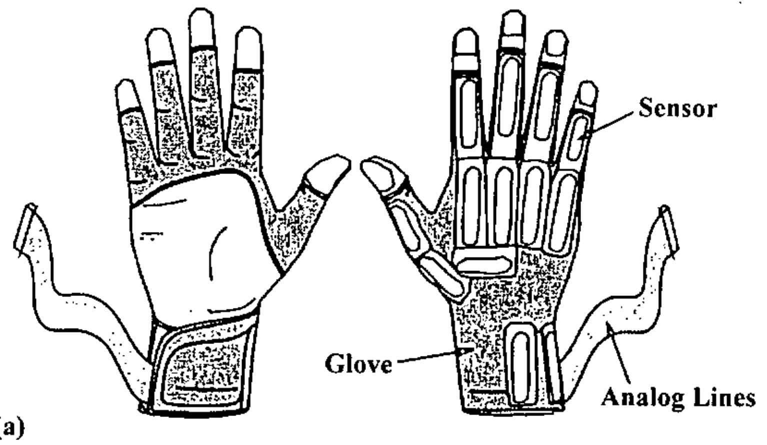


Figure 4-4 The *Cyber*Glove^{TM} device.



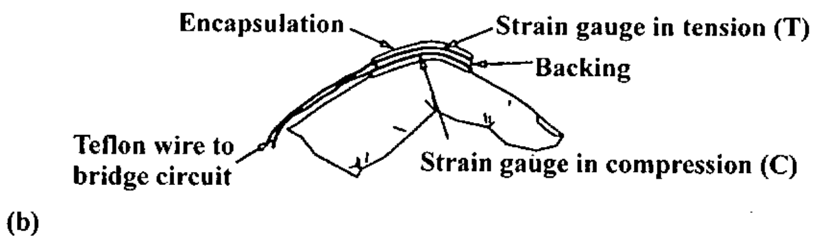
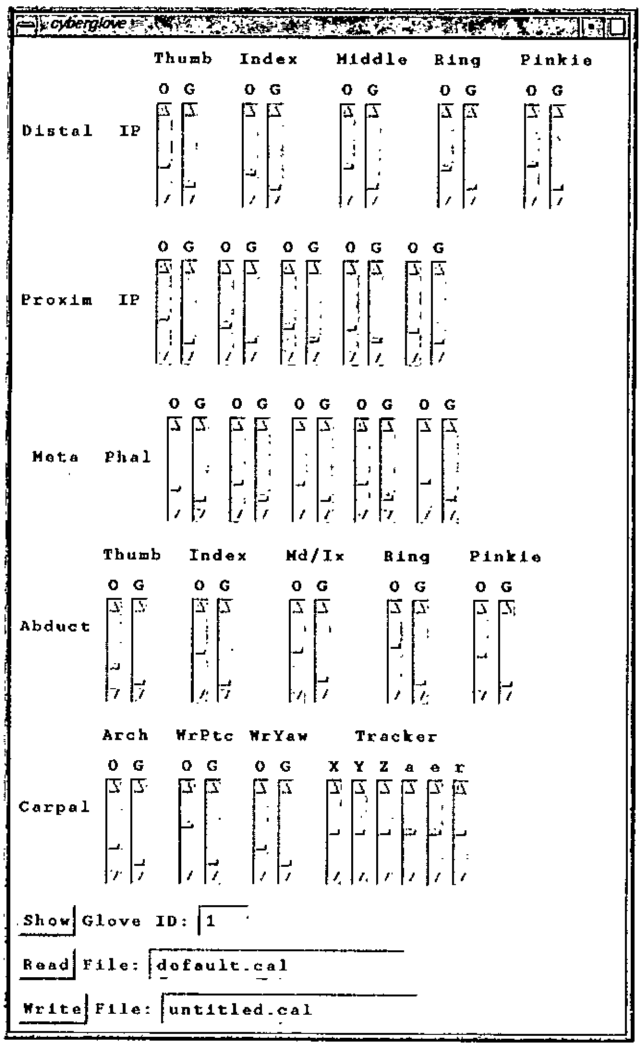

Encapsulation
Strain gauge in tension (T)
Backing
Teflon wire to bridge circuit
Strain gauge in compression (C)
(b)

Figure 4-5 The *Cyber*Glove^{TM} (a) Instrumented Glove; (b) Sensor Detail.

Each joint flexing angle is measured indirectly by a change of resistance in the pair of strain gauges. During finger motion, one of the strain gauges is under compression (C), while the other is under tension (T) (Figure 4-5b). Their change in resistance produces a change in voltage. These differential voltages are then multiplexed by an analog multiplexer, and amplified and digitized by an A/D converter. The digitized voltages are sampled by a host computer which uses a calibration program to obtain the joint angles. The joint "0" position can be changed in hardware (by using offset potentiometers) or in software with a user-friendly graphical interface (Figure 4-6).



Glove ID: 1

File: default.cal

File: untitled.cal

Figure 4-6 The Calibration Interface for *VirtualHand*^{TM} Software.

The glove sensors are either rectangular (for the flexion angles), or U-shaped (for adduction-abduction angles). The *Cyber*Glove^{TM} that we use in the prototype contains 18 sensors. Two sensors are used to measure the flexing of each of the five fingers. On the thumb, two sensors are used to measure the metacarpophalangeal (MP) and interphalangeal (IP) joints (Figure 4-2). On the remaining four fingers, two bend sensors are used to measure the MP and proximal IP joints. Abduction sensors are provided for the thumb, middle-index, ring-middle and pinkie-ring fingers. The abduction sensor measures the amount that the corresponding finger moves laterally in the plane of the palm. The thumb has an additional sensor which measures how much the thumb rotates across the palm toward the pinkie finger. Similarly, the pinkie also has a sensor which measures the amount that the pinkie rotates across the palm toward the thumb, i.e., the arch of the palm near the pinkie finger when the hand is cupped. Finally, there are two sensors, one to measure wrist pitch and one to measure wrist yaw. According to the manufacturer, sensor resolution is 0.5 degree and remains constant over the entire range of joint motion [*Cyber*Glove94]. The manufacturer has also claimed that the glove has decoupled sensors so that outputs are independent of each other.

*4.2.1.2 The Cyber*Glove^{TM} *Hand Model*

The hand model of the *Cyber*Glove^{TM} could be considered as an articulated rigid body system consisting of rigid bone segments connected by joints [Spong89]. The joints are considered as to be rotary and is up to a number of degrees of rotational motion only but with no translational movement. Geometrically, two types of coordinate systems are considered in the hand model. One of them is the ***inertial coordinate system*** which is fixed in space. This is similar to the world coordinates that are used in computer graphics. Another coordinate system is ***moving frame coordinate system*** or local coordinate systems (Figure 4-7). Each segment in the system has its own *moving frame coordinate system*, which refers to the coordinate system attached to each segment of the hand model. This coordinate system moves and rotates with the segment. The origin of the *moving frame coordinate system* is chosen to be at the joint nearest to the wrist of each of the finger segments. For each of the segment, the *moving frame coordinate* can be transformed into the *inertial coordinate* by a rotation and a translation operation. Note that the transformation will change over time as the hand moves through space. Therefore, the transformation matrix for each articulating rigid segment will be generated at each frame, and consequently, the position of each of the joints could be measured.

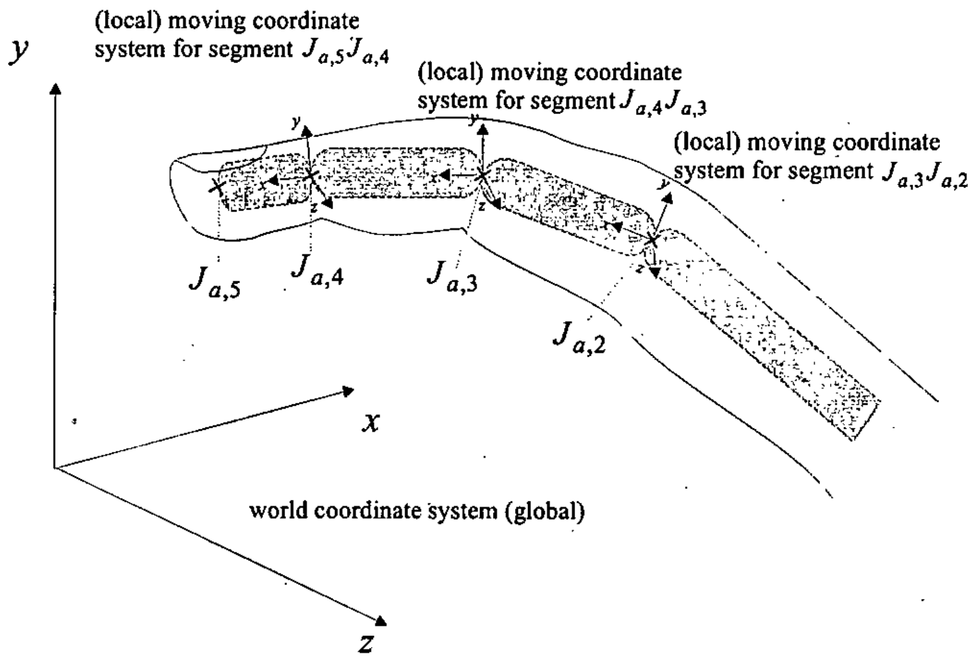

(local) moving coordinate system for segment $J_{a,5}J_{a,4}$
(local) moving coordinate system for segment $J_{a,4}J_{a,3}$
(local) moving coordinate system for segment $J_{a,3}J_{a,2}$
y
$J_{a,5}$
$J_{a,4}$
$J_{a,3}$
$J_{a,2}$
x
world coordinate system (global)
z

Figure 4-7 Coordinate Systems of the Hand Model.

4.2.2 3D Tracking Input Sensors

Tracking sensors have evolved for detecting an object's 3D position and orientation. This kind of tracking requires a physical sensor attached to the object and is known as active tracking. Active 6-DOF trackers use either electromagnetic, mechanical, optical, or ultrasonic techniques for making 6-DOF measurements. Among them, the electromagnetic tracker is the most popular one because of the sensor's small size and freedom of movement.

For electromagnetic trackers, a low-frequency signal is generated by a control box (transmitter). This signal sequentially excites three small coils of wire in the source and thus creates three magnetic fields. When a similar set of three coiled wires (receiver/sensor) is positioned in range of the source, a small voltage is indeced in each of the coils as the three fields are created. This yields a total of nine measurements that are then processed by the control box to yield six values for position and orientation. This technique was first developed over ten years ago by a company called Polhemus Inc., for military applications. The Polhemus FASTRAK system (Figure 4-8) can simultaneously support up to four sensors and two sources. It has since evolved into a reliable and accurate method of tracking. Because the sensor is about the size of a dice cube and does not rely on line of sight, it can be buried inside other devices, like the *Cyber*Glove.

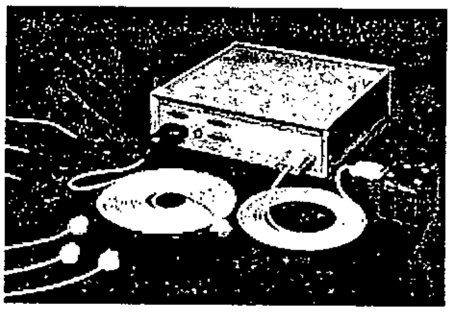


Figure 4-8 FASTRAK with Polhemus sensors.

The major limitation of this kind of tracking system is the lengthy lag time due to the signal processing and filtering of the import measurements. However, Polhemus has dramatically reduced this lag to less than 5 ms (unfiltered), greatly improving its usefulness. Another limitation of these systems is the sensitivity against large metal objects or to magnetic fields. However, this problem is usually easy to avoid.

In our implementation, a Polhemus receiver is attached to the wrist position on the *Cyber*Glove. It provides dynamic, real time six degree-of-freedom (3D position and orientation) tracking of the glove.

4.3 Construction of the control surface

4.3.1 Data Points Collection and Deduction

The *Virtual*Hand™ software library provides functions for returning the transformation matrices of the 5×3 articulated rigid segments comprising the virtual hand model [*Virtual*Hand94]. These matrices hold the transformations to successive joint origins. The first dimension refers to the five fingers (i.e. thumb, index, middle, ring and pinkie). The second dimension refers to the first, second and third joints nearest to the fingertip (Figure 4-2: the IP, MP and Trapeziometacarpal joint for the thumb and the DIP, PIP and MCP for the others). Each of the segments is in its *moving frame coordinate*. Applying specific transformation matrix to one joint, the new matrix holds the coordinate system for the next joint. From these transformation matrices, we can determine 5×3 joint positions (Figure 4-8).

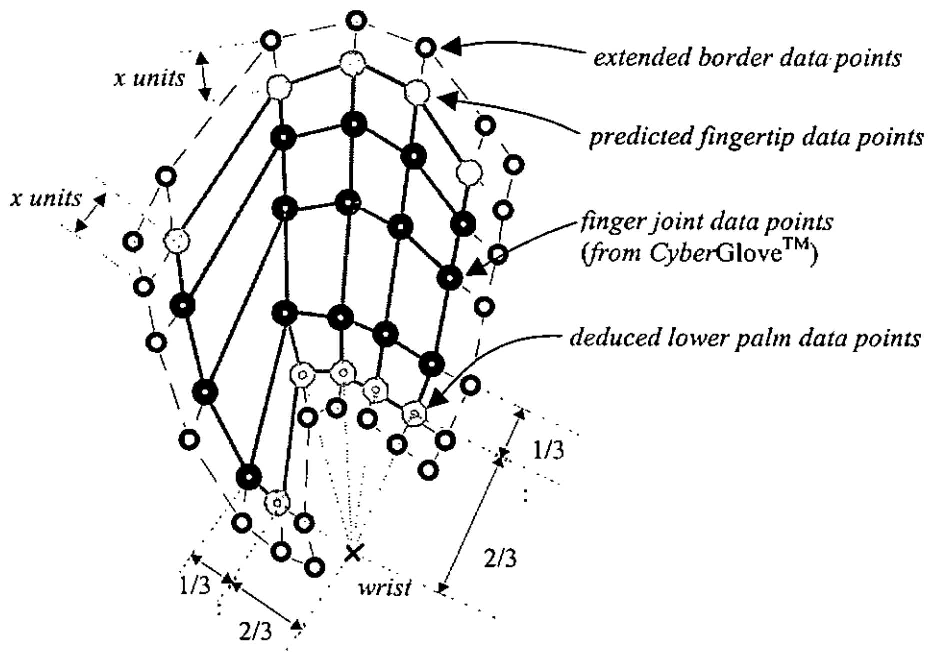

extended border data points
predicted fingertip data points
finger joint data points (from CyberGlove™)
deduced lower palm data points
x units
x units
1/3
2/3
1/3
2/3
wrist

Figure 4-8 Data Point Deduction.

Since the 18-sensor *Cyber*Glove does not explicitly measure the positions of the joints nearest to the fingertips, these data must be determined by some other means. As for most normal human hand motion, the joints nearest to the fingertips are correlated to the neighboring joint. Therefore, the fingertip positions can be estimated by taking into account the coupling that exists between θ_5 and θ_4 (Figure 4-9) over the range of grasping motions. Experimental measurements show that the general coupling formula is of the form [Burdea92]

$$\theta_5 = a_0 - a_1\theta_4 + a_2\theta_4^2$$

where θ_5 is the flexion angle of the distal joints of any of the fingers, except for the thumb and the parameters $a_0, a_1,$ and a_2 depend on the hand characteristics of individual users.

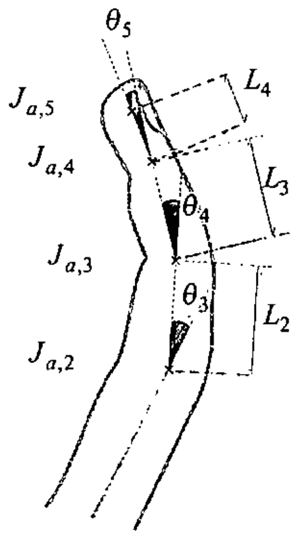


Figure 4-9 Parameters of a Finger.

As a result, 5×1 fingertip positions can be predicted. We then set the data points defining the lower palm of the hand from the other data points (wrist position and the joints nearest to the wrist). By using a Polhemus tracker introduced in section 4.2.2, the wrist position and the hand orientation can be obtained and the joints nearest to the wrist can be calculated from the corresponding transformation matrices produced from the *Cyber*Glove^{TM}. The first joint position of each finger is set to the position 2/3 from the wrist to the joints nearest to it (Figure 4-8). Now, we have 5×5 data points representing the feature gesture of the hand. We then extend the data points to 7×7 simply by extending the last segments in each of the dimensions on the edge by x unit(s). The border parts are used for the prevention of unmatched gap and preserving the continuity where local modification is applied to an object. This can model the effect of elastic property of the object. The more elastic the object is, the smaller the parameter x and hence the narrower the border should be. Detail of the modification will be discussed in chapter 6. We index the data points obtained as $J_{a,b}$ where $a = 0,...,6$ and $b = 0,...,6$ for convenience (Figure 4-10).

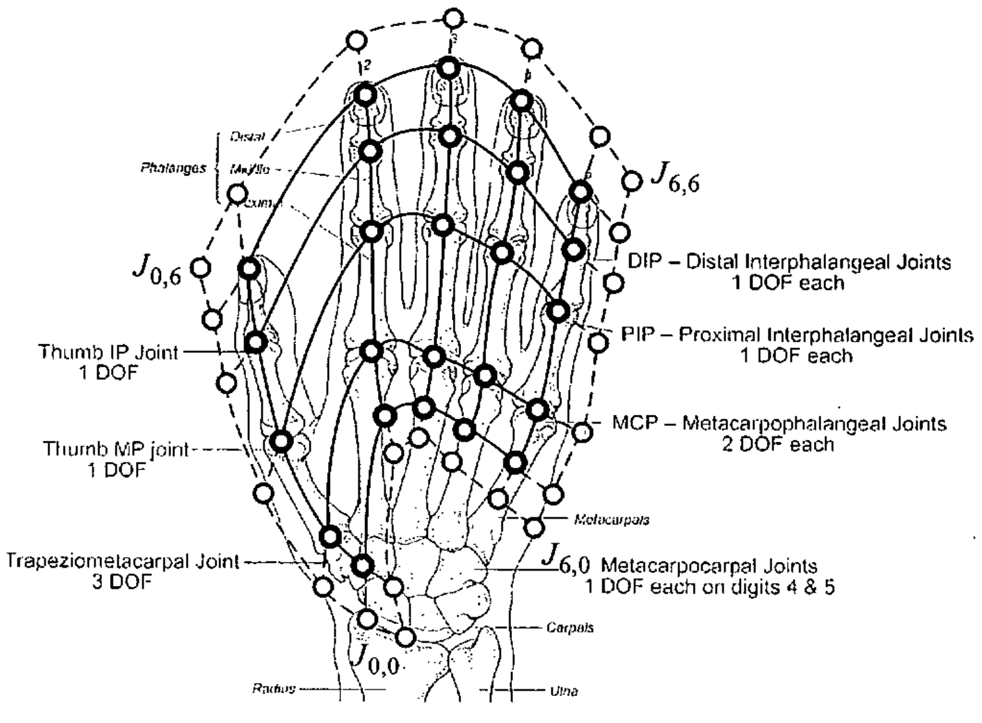

Phalanges
Distal
Middle
Proximal
$J_{6,6}$
$J_{0,6}$
DIP – Distal Interphalangeal Joints 1 DOF each
PIP – Proximal Interphalangeal Joints 1 DOF each
MCP – Metacarpophalangeal Joints 2 DOF each
Thumb IP Joint 1 DOF
Thumb MP joint 1 DOF
Metacarpals
Trapeziometacarpal Joint 3 DOF
$J_{6,0}$ Metacarpocarpal Joints 1 DOF each on digits 4 & 5
Carpals
$J_{0,0}$
Radius
Ulna

Figure 4-10 The Interpolation of the Parametric *Control Hand Surface* to the Finger Joint Data from the *CyberGlove*^{TM}.

4.3.2 *Definition of the Parametric Control Hand Surface*

4.3.2.1 *Cubic B-Spline Tensor Product Surface*

As already discussed in chapter 2, the cubic B-Spline tensor product surface is believed to be sufficient to model the *control hand surface.* The construction of the parametric *control hand surface* is done by employing the technique called interpolating method [Farin97].

The cubic B-Spline tensor product *control hand surface* is given by

$$H(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} Nu_i^3(u) Nv_j^3(v) P_{i,j} \tag{4-1}$$

According to the interpolating method, the problem can be seen as:

Given 7×7 data points $J_{a,b}$ (where $a = 0,\ldots,6$ and $b = 0,\ldots,6$), our goal is to fit a B-Spline tensor product surface with knot vectors U and V in the two dimensions respectively with $m \times n$ unknown control points $P_{i,j}$ (where $i = 0,\ldots,m$ and $j = 0,\ldots,n$). The surface is required to interpolate all 7×7 data points, $J_{a,b}$, at some parametric coordinate $(u_{a,b}, v_{a,b})$ where $0 \le u_{a,b}, v_{a,b} \le 1$. That is,

$$H(u_{a,b}, v_{a,b}) = J_{a,b} = \sum_{i=0}^{m} \sum_{j=0}^{n} Nu_i^3(u_{a,b}) Nv_j^3(v_{a,b}) P_{i,j} \tag{4-2}$$

where $Nu_i^3(u)$ and $Nv_j^3(v)$ are the cubic basis functions at particular parameter values u in U-dimension and v in V-dimension, respectively.

4.3.2.2 *Knot Parameterization*

Now, we compute appropriate values for $(u_{a,b}, v_{a,b})$. These values are chosen according to the chord length ratios between successive data points and then by averaging across the ratios row by row and column by column in the two dimensions respectively. With this method, they are able to reflect the distribution of the data points. The steps are shown below:

Let Lu^b (where $b = 0,\dots,6$) be the total chord length of the bth column of the data in *U*-dimension and Lv^a (where $a = 0,\dots,6$) be the total chord length of the ath row of the data in *V*-dimension, such that

$$Lu^b = \sum_{a=0}^{5} \left| J_{a+1,b} - J_{a,b} \right| \qquad \text{for } b = 0,\dots,6 \qquad (4\text{-}3)$$

$$Lv^a = \sum_{b=0}^{5} \left| J_{a,b+1} - J_{a,b} \right| \qquad \text{for } a = 0,\dots,6 \qquad (4\text{-}4)$$

For each data points $J_{a,b}$, we calculate the normalized chord length ratio $u_{i,b}$ and $v_{a,j}$ in *U*- and *V*- dimension respectively. Obviously at the end points,

$$u_{0,b} = 0 \text{ and } u_{6,b} = 1 \qquad \text{for } b = 0,\dots,6 \qquad (4\text{-}5)$$

$$v_{a,0} = 0 \text{ and } v_{a,6} = 1 \qquad \text{for } a = 0,\dots,6 \qquad (4\text{-}6)$$

For the intermediate data points, the normalized chord length ratios $u_{i,b}$ and $v_{a,j}$ in *U*- and *V*- dimension respectively are given by

$$u_{i,b} = u_{i-1,b} + \frac{\left| J_{i,b} - J_{i-1,b} \right|}{Lu^b} \qquad \text{for } i = 1,\dots,5 \text{ and } b = 0,\dots,6 \qquad (4\text{-}7)$$

$$v_{a,j} = v_{a,j-1} + \frac{\left| J_{a,b} - J_{a,b-1} \right|}{Lv^a} \qquad \text{for } j = 1,\dots,5 \text{ and } a = 0,\dots,6 \qquad (4\text{-}8)$$

Recall the properties of the tensor product surface from section 2.4. We may consider that the connections of the finger joints form isoparametric curves (Figure 4-11). Therefore, we may set the parameters as follows

$u_{a,0} = u_{a,1} = \ldots = u_{a,5} = u_{a,6}$ where $a = 0,\ldots,6$

and let the values be $\bar{u}_a$.

$v_{0,b} = v_{1,b} = \ldots = v_{5,b} = v_{6,b}$ where $b = 0,\ldots,6$

and let the values be $\bar{v}_b$.

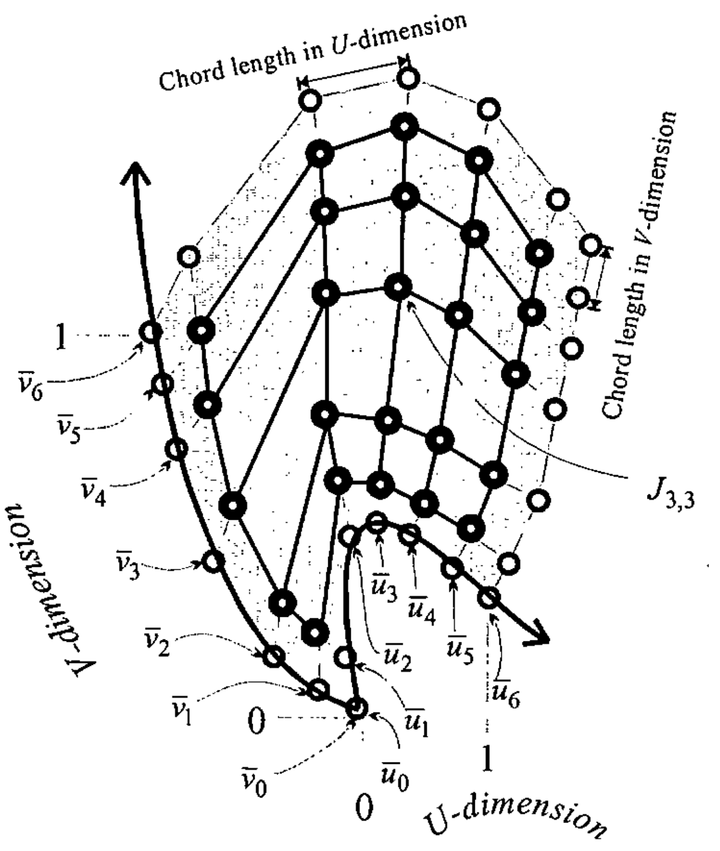


Figure 4-11 Definition of the Parametric Hand Surface.

By averaging across all ratios $u_{i,b}$ and $v_{a,j}$ in the *U*- and *V*- dimension respectively, $\bar{u}_a$ and $\bar{v}_b$ can be evaluated. Follow directly from Equation

(4-5) and (4-6),

$$\bar{u}_0 = 0 \text{ and } \bar{u}_6 = 1 \tag{4-9}$$

$$\bar{v}_0 = 0 \text{ and } \bar{v}_6 = 1 \tag{4-10}$$

And by averaging,

$$\bar{u}_a = \sum_{b=0}^{6} u_{a,b} / 7 \qquad \text{where } a = 1,\ldots,5 \tag{4-11}$$

$$\bar{v}_b = \sum_{a=0}^{6} v_{a,b} / 7 \qquad \text{where } b = 1,..,5 \tag{4-12}$$

Now, we can determine the knot vectors *U* and *V*. The knot vectors are taken in the form of Bézier type parametric representation (Section 2.3). For the interior knots, we set them to coincide with the data points. That is,

$$U = \{0,0,0,0,\bar{u}_1,\bar{u}_2,\bar{u}_3,\bar{u}_4,\bar{u}_5,1,1,1,1\} \tag{4-13}$$

$$V = \{0,0,0,0,\bar{v}_1,\bar{v}_2,\bar{v}_3,\bar{v}_4,\bar{v}_5,1,1,1,1\} \tag{4-14}$$

Recall that the number of knots of the knot vector defining the surface determines the number of control points included in it. Now, we have 13 knots defining each of the dimensions of the surface. Therefore, there should be 9×9 control points to define the surface. That is *m*=9 and *n*=9 in Equation (4-2).

4.3.3 *Efficient Calculation of the Control Hand Surface*

4.3.3.1 *Tensor Product Nature*

The matrix form of Equation (4-2) can be rewritten as

$$\begin{bmatrix} J_{0,0} & \cdots & J_{0,6} \\ \vdots & \ddots & \vdots \\ J_{6,0} & \cdots & J_{6,6} \end{bmatrix} = \begin{bmatrix} Nu_0^3(\bar{u}_0) & \cdots & Nu_8^3(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ Nu_0^3(\bar{u}_6) & \cdots & Nu_8^3(\bar{u}_6) \end{bmatrix} \cdot \begin{bmatrix} P_{0,0} & \cdots & P_{0,8} \\ \vdots & \ddots & \vdots \\ P_{8,0} & \cdots & P_{8,8} \end{bmatrix} \cdot \begin{bmatrix} Nv_0^3(\bar{v}_0) & \cdots & Nv_0^3(\bar{v}_6) \\ \vdots & \ddots & \vdots \\ Nv_8^3(\bar{v}_0) & \cdots & Nv_8^3(\bar{v}_6) \end{bmatrix} \quad (4\text{-}15)$$

Recall that $J_{a,b}$ are the position data obtained from the *Cyber*Glove™. $Nu_i^3(u_a)$ and $Nv_j^3(v_b)$ are the cubic basis functions at knots u_a in *U*-dimension and v_b in *V*-dimension, respectively. $P_{i,j}$ are the control points of the hand surface that fulfill the condition of interpolating $J_{a,b}$.

To improve efficiency of modelling the *control hand surface* with the bicubic B-Spline tensor product surface, we do not determine the inverse of the matrices implicitly. Instead, we simplify the problem by considering the the tensor product nature of the *control hand surface* [Farin97].

First of all, we introduce a set of intermediate control points, $E_{a,b}$ where $a = 0,\ldots,6$ and $b = 0,\ldots,8$, such that

$$\begin{bmatrix} E_{0,0} & \cdots & E_{0,8} \\ \vdots & \ddots & \vdots \\ E_{6,0} & \cdots & E_{6,8} \end{bmatrix} = \begin{bmatrix} Nu_0^3(\bar{u}_0) & \cdots & Nu_8^3(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ Nu_0^3(\bar{u}_6) & \cdots & Nu_8^3(\bar{u}_6) \end{bmatrix} \cdot \begin{bmatrix} P_{0,0} & \cdots & P_{0,8} \\ \vdots & \ddots & \vdots \\ P_{8,0} & \cdots & P_{8,8} \end{bmatrix} \quad (4\text{-}16)$$

Equation (4-15) then becomes

$$\begin{bmatrix} J_{0,0} & \cdots & J_{0,6} \\ \vdots & \ddots & \vdots \\ J_{6,0} & \cdots & J_{6,6} \end{bmatrix} = \begin{bmatrix} E_{0,0} & \cdots & E_{0,8} \\ \vdots & \ddots & \vdots \\ E_{6,0} & \cdots & E_{6,8} \end{bmatrix} \cdot \begin{bmatrix} Nv_0^3(\bar{v}_0) & \cdots & Nv_0^3(\bar{v}_6) \\ \vdots & \ddots & \vdots \\ Nv_8^3(\bar{v}_0) & \cdots & Nv_8^3(\bar{v}_6) \end{bmatrix} \quad (4\text{-}17)$$

The 2D interpolation problem is now reduced to two sets of 1D interpolation problems. Now, we examine each of the 1D interpolation problems.

By considering each *r*th row, where $r = 0,\ldots,6$, in Equation (4-17), the problem is as follows.

> Given 7 data points $J_{r,j}$ (where $j = 0,\ldots,6$). Our goal is to find the 9 unknown intermediate control points $E_{r,b}$ (where $b = 0,\ldots,8$) of the *open-uniform cubic B-Spline* isoparametric curve $C_r^v(v)$ with knot vector $\{0,0,0,0,\bar{v}_1,\bar{v}_2,\bar{v}_3,\bar{v}_4,\bar{v}_5,1,1,1,1\}$ such that it interpolates the data points as $C_r^v(\bar{v}_i) = J_{r,i}$. Also, the curve satisfies the end point interpolation condition. That is, $C_r^v(0.0) = J_{r,0}$ and $C_r^v(1.0) = J_{r,6}$.

By resolving the above problem for each of the 7 rows, 7×9 intermediate control points $E_{a,b}$ will be resulted. The intermediate control points can then be used as the data points in Equation (4-16), and similarly by considering each *s*th column, where $s = 0,\ldots,8$, the problem is

> Given 7 data points $E_{i,s}$ (where $i = 0,\ldots,6$). Our goal is to find the 9 unknown control points $P_{a,s}$ (where $b = 0,\ldots,8$) of the *open-uniform cubic B-Spline* isoparametric curve $C_s^u(u)$ with knot vector $\{0,0,0,0,\bar{u}_1,\bar{u}_2,\bar{u}_3,\bar{u}_4,\bar{u}_5,1,1,1,1\}$ such that it interpolates the data points as $C_s^u(\bar{u}_i) = E_{i,s}$. Also, the curve satisfies the end point interpolation condition. That is, $C_s^u(0.0) = E_{0,s}$ and $C_s^u(1.0) = E_{6,s}$.

After solving the $7+7$ 1D interpolating problems, the 9×9 control points defining the tensor product surface are determined.

Notice that because of the tensor product nature of the *control hand surface*, the 2D interpolation problem of Equation (4-15), initially represented by 7×7 linear equations, is broken down into $7+7$ 1D curve interpolation problems.

4.3.3.2 End Points Interpolations

Recall that for all *open-uniform B-Splines,* the end control points always interpolate the end data points. Therefore, the intermediate end control points $E_{a,0}$ and $E_{a,8}$ in Equation (4-17) can be evaluated as

$$E_{a,0} = J_{a,0} \text{ and } E_{a,8} = J_{a,6} \qquad \text{for } a = 0,\dots,6 \tag{4-18}$$

And the system of equations is reduced to

$$\begin{bmatrix} J_{0,1} & \cdots & J_{0,5} \\ \vdots & \ddots & \vdots \\ J_{6,1} & \cdots & J_{6,5} \end{bmatrix} = \begin{bmatrix} E_{0,1} & \cdots & E_{0,7} \\ \vdots & \ddots & \vdots \\ E_{6,1} & \cdots & E_{6,7} \end{bmatrix} \cdot \begin{bmatrix} Nv_0^3(\bar{v}_1) & \cdots & Nv_0^3(\bar{v}_5) \\ \vdots & \ddots & \vdots \\ Nv_8^3(\bar{v}_1) & \cdots & Nv_8^3(\bar{v}_5) \end{bmatrix} \tag{4-19}$$

Similarly, $P_{0,b}$ and $P_{8,b}$ can be evaluated as

$$P_{0,b} = E_{0,b} \text{ and } P_{8,b} = E_{6,b} \qquad \text{for } b = 0,\dots,8$$

And the system of equations in Equation (4-16) becomes

$$\begin{bmatrix} E_{1,0} & \cdots & E_{1,8} \\ \vdots & \ddots & \vdots \\ E_{5,0} & \cdots & E_{5,8} \end{bmatrix} = \begin{bmatrix} Nu_1^3(\bar{u}_1) & \cdots & Nu_7^3(\bar{u}_1) \\ \vdots & \ddots & \vdots \\ Nu_1^3(\bar{u}_5) & \cdots & Nu_7^3(\bar{u}_5) \end{bmatrix} \cdot \begin{bmatrix} P_{1,0} & \cdots & P_{1,8} \\ \vdots & \ddots & \vdots \\ P_{7,0} & \cdots & P_{7,8} \end{bmatrix} \tag{4-20}$$

4.3.3.3 Designation of knot values at Data points

According to the local control property (section 2.3) of B-Spline surfaces, each of the control point, P_i, only exerts control over the region $u \in [u_i, u_{i+p+1})$, the choice of the knot vectors can effectively simplify the interpolation problems. Recall that the interpolation of data points $J_{i,j}$ are chosen to occur at the interior knots, which define the *control hand surface.* And from the local control property of B-Spline, at each of these data points, there are only three nonzero cubic basis functions (Figure 4-12).

$$S(\overline{u}_i) = N_i^3(\overline{u}_i)E_i + N_{i+1}^3(\overline{u}_i)E_{i+1} + N_{i+2}^3(\overline{u}_i)E_{i+2} \tag{4-21}$$

By Setting $\alpha_i = Nu_i^3(\overline{v}_i)$, $\beta_i = N_{i+1}^3(\overline{v}_i)$ and $\gamma_i = N_{i+2}^3(\overline{v}_i)$, the matrix of the B-Spline basis functions is reduced to a tridiagonal matrix and Equation (4-19) becomes

$$\begin{bmatrix} J_{0,1} & \cdots & J_{0,5} \\ \vdots & \ddots & \vdots \\ J_{6,1} & \cdots & J_{6,5} \end{bmatrix} = \begin{bmatrix} E_{0,1} & \cdots & E_{0,7} \\ \vdots & \ddots & \vdots \\ E_{6,1} & \cdots & E_{6,7} \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 & 0 & 0 & 0 & 0 \\ \beta_1 & \alpha_2 & 0 & 0 & 0 \\ \gamma_1 & \beta_2 & \alpha_3 & 0 & 0 \\ 0 & \gamma_2 & \beta_3 & \alpha_4 & 0 \\ 0 & 0 & \gamma_3 & \beta_4 & \alpha_5 \\ 0 & 0 & 0 & \gamma_4 & \beta_5 \\ 0 & 0 & 0 & 0 & \gamma_5 \end{bmatrix} \tag{4-22}$$

Similarly, by Setting $\lambda_i = N_i^3(\overline{u}_i)$, $\phi_i = N_{i+1}^3(\overline{u}_i)$ and $\varphi_i = N_{i+2}^3(\overline{u}_i)$, Equation (4-20) is simplified to the following tridiagonal system.

$$\begin{bmatrix} E_{1,0} & \cdots & E_{1,8} \\ \vdots & \ddots & \vdots \\ E_{5,0} & \cdots & E_{5,8} \end{bmatrix} = \begin{bmatrix} \lambda_1 & \phi_1 & \varphi_1 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & \phi_2 & \varphi_2 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & \phi_3 & \varphi_3 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & \phi_4 & \varphi_4 & 0 \\ 0 & 0 & 0 & 0 & \lambda_5 & \phi_5 & \varphi_5 \end{bmatrix} \cdot \begin{bmatrix} P_{1,0} & \cdots & P_{1,8} \\ \vdots & \ddots & \vdots \\ P_{7,0} & \cdots & P_{7,8} \end{bmatrix} \tag{4-23}$$

These tridiagonal systems can be solved by numerical methods [Engeln96] without explicitly evaluating the inverse of the matrices. As a result, the control points are evaluated and the *control hand surface* can then be well defined.

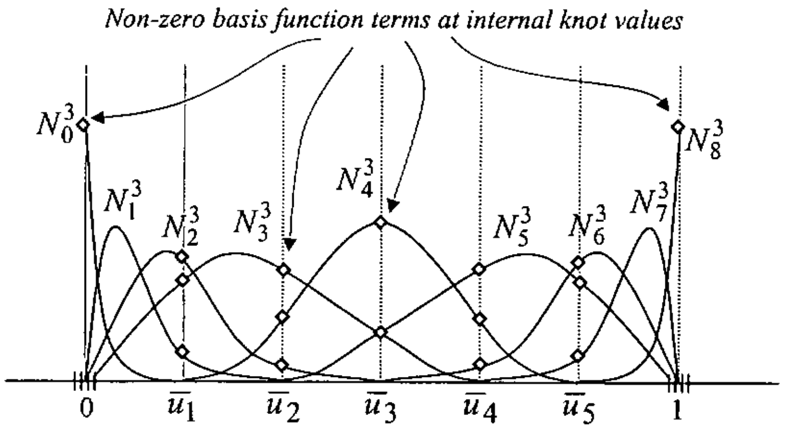


Figure 4-12 Basis Functions of cubic B-Spline.

Chapter 5

Mapping of Object to the Control Hand Surface

In FFDs, object is embedded in a 3D bounding volume. The deformation of the object is done by deforming the space around them (Section 3.2.1). This method is used for deforming solid geometric model in a free-form manner. In our method, surface deformation is induced by passing the change of geometric attributes of the *control hand surface* to the object. In order to facilitate the deformation effect, a well-defined mapping of the object vertices from the Euclidean 3D space to the parametric domain of the *control hand surface* is necessary. In our method, we consider the object to be deformed as embedded in the extended 2D parametric space of the *control hand surface.*

5.1 Setting up the Corresponding Mapping

Generally, each of the geometric attributes, say object vertices, of the virtual object to be deformed is projected to the parametric domain of the hand surface (Figure 5-1). That is, a set of parametric coordinates (u_a, v_a) will be assigned to each of these object vertices V_a. This mapping remains constant as long as the deformation proceeds. This provides an effect of sculpturing with one's hand in contact with the object. (Each of the object vertices is kept in a constant relationship with a specific position on the hand, which is determined by the mapped parametric coordinate.) Change of hand gesture would induce change to the 3D position of a specific parametric coordinate. Consequently, this change is passed to the object through particular modification/deformation algorithm which will be discussed in Chapter 6. As a result, the shape of the object is changed according to the change of hand gesture.

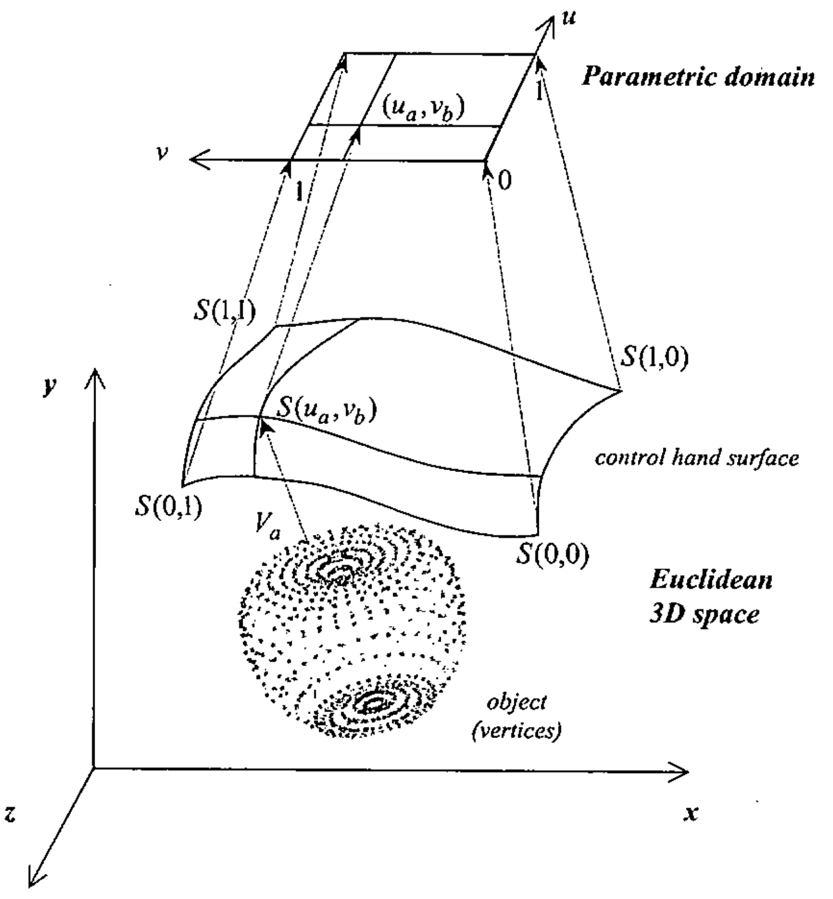


Figure 5-1 Mapping of Spatial domain of the Object Surface to the Parametric domain of the Hand Surface.

5.2 The Mapping Process

The method we developed for mapping an object to be deformed to the *control hand surface* is referred to as *Ray-Projection*. More precisely, it is used to correlate each of the object vertices to a position on the *control hand surface.* The mapping is determined by the parametric coordinate assigned to each of the object vertices. In this section, we describe the concept of *Ray-Projection* in general. Then, we describe how we adapt the method in the prototype.

5.2.1 Ray-projection

This method is originated from the concept of ray-casting. It makes use of converging rays to map an object vertex V_a on the Euclidean 3D space to the 2D parametric space of the *control hand surface.* Initially, a point called the centre of projection, P_c, is determined. Ray-projection is referred to projecting a ray directed from P_c through V_a onto the other surface, that is, the *control hand surface.* (Figure 5-2).

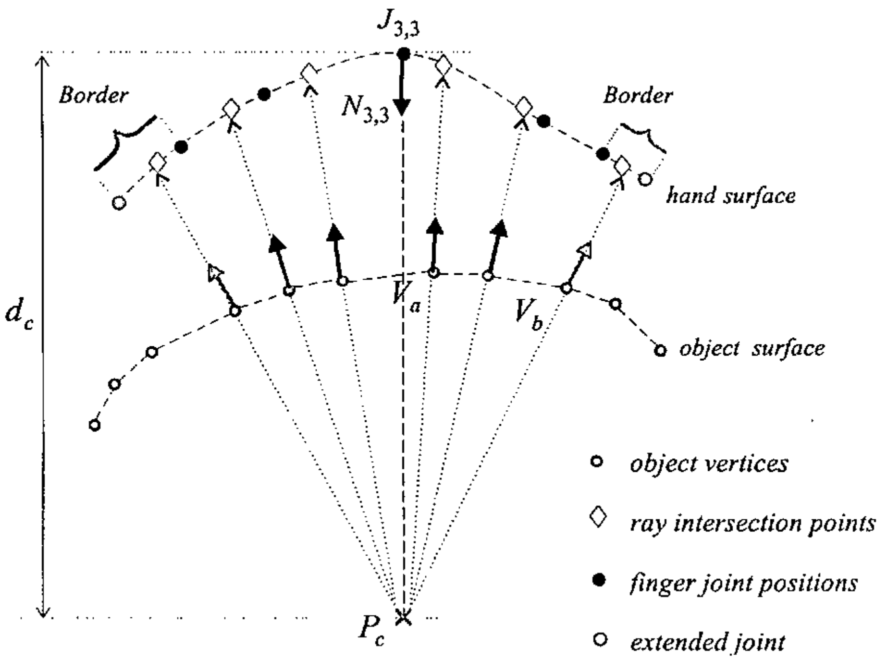


Figure 5-2 Ray-Projection.

In the prototype, each of the object vertices has to be projected onto the *control hand surface*. However, it is difficult to determine the intersection of a ray and a parametric surface. To simplify the clumsy calculations, we propose to approximate the parametric surface by a polygonal hand model which we refer to as the *3D triangulated hand model* in the remaining document.

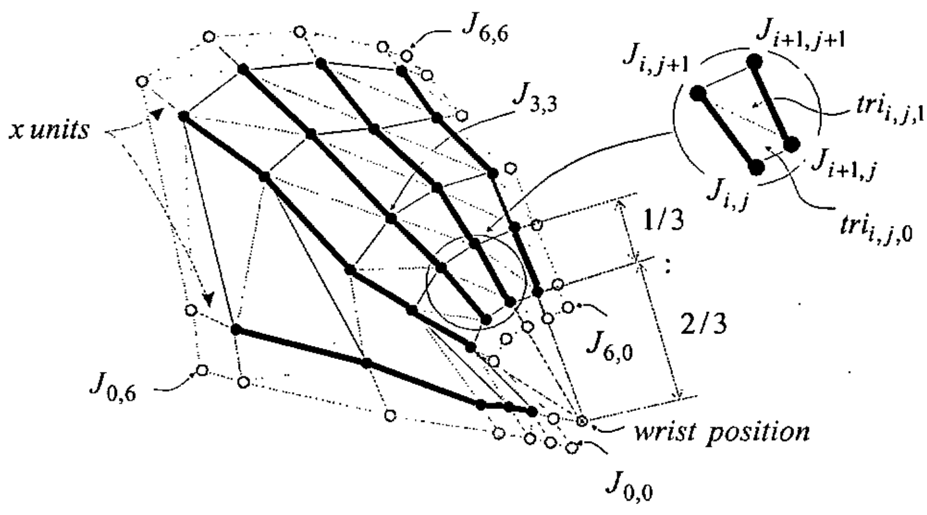


Figure 5-3 3D Triangulated Hand Model.

The *3D triangulated hand model* is constructed by triangulating the data points $J_{i,j}$, where $i = 0,\ldots,6$ and $j = 0,\ldots,6$. Recall that the hand is modelled by a tensor product surface interpolating these data points. By joining neighboring data points in *U*- and *V*- dimension as in Figure 5-3, we have a tetrahedron mesh with 6×6 tetrahedrons. Since it is not guaranteed that the four vertices of a tetrahedron are on a single plane, we further break each of the tetrahedrons into two triangles as follows,

$$\begin{cases} tri_{i,j,0} : \Delta J_{i,j} J_{i+1,j} J_{i,j+1} \\ tri_{i,j,1} : \Delta J_{i+1,j} J_{i+1,j+1} J_{i,j+1} \end{cases}$$

where $i = 0,\ldots,5$ and $j = 0,\ldots,5$. Therefore the *3D triangulated hand model* with

$6 \times 6 \times 2$ triangles is formed (Figure 5-3) and each triangle $tri_{i,j,\#}$ is on a plane $T_{i,j,\#}$ defined as

$$T_{i,j,\#} : (P \cdot N_{i,j,\#}) + d_{i,j,\#} = 0 \tag{5-1}$$

and

$$\begin{cases} N_{i,j,0} = \overrightarrow{J_{i,j}J_{i+1,j}} \times \overrightarrow{J_{i+1,j}J_{i,j+1}} \\ N_{i,j,1} = \overrightarrow{J_{i+1,j}J_{i+1,j+1}} \times \overrightarrow{J_{i+1,j+1}J_{i,j+1}} \end{cases}$$

where $i = 0,\dots,5$, $j = 0,\dots,5$ and $\# = 0,1$. P is any point on the plane $T_{i,j,\#}$. $d_{i,j,\#}$ is the offset of the plane from the origin and $N_{i,j,\#}$ is the normal of the plane.

The centre of projection, P_c, is determined to be at a certain distance, d_c, from the palm centre depending on the deformation effect one desires. (Please refers to Chapter 6 for details.) By employing the *ray-projection* method, a ray directed from P_c through each of the vertices V_a (where $a = 0,\dots,n-1$ of an object consisting of n vertices) is projected onto the *3D triangulated hand model.*

5.2.1.1 Intersection Test

Mathematically, a ray R_a for a particular vertex V_a is defined as:

$$R_a : P = V_a + k \cdot [V_a - P_c]_I \tag{5-2}$$

where $[V_a - P_c]_I$ is the unit vector directed from V_a to P_c, and P is a point on R_a at some scalar variable k.

In order to determine whether R_a intersects the plane $T_{i,j,\#}$ defined by

Equation (5-1), we need to solve for *k* as follows,

$$k = -\frac{d_{i,j,\#} + N_{i,j,\#} \cdot V_a}{N_{i,j,\#} \cdot \left[V_a - P_c\right]_I} \tag{5-3}$$

Notice that if $N_{i,j,\#} \cdot \left[V_a - P_c\right]_I$ equals zero, then R_a and $T_{i,j,\#}$ are parallel to each other. Otherwise, they will intersect at some k and the *ray-intersection point* of R_a and $T_{i,j,\#}$ is given by,

$$V_a^{proj} = V_a - \frac{d_{i,j,\#} + N_{i,j,\#} \cdot V_a}{N_{i,j,\#} \cdot \left[V_a - P_c\right]_I}\left[V_a - P_c\right]_I \tag{5-4}$$

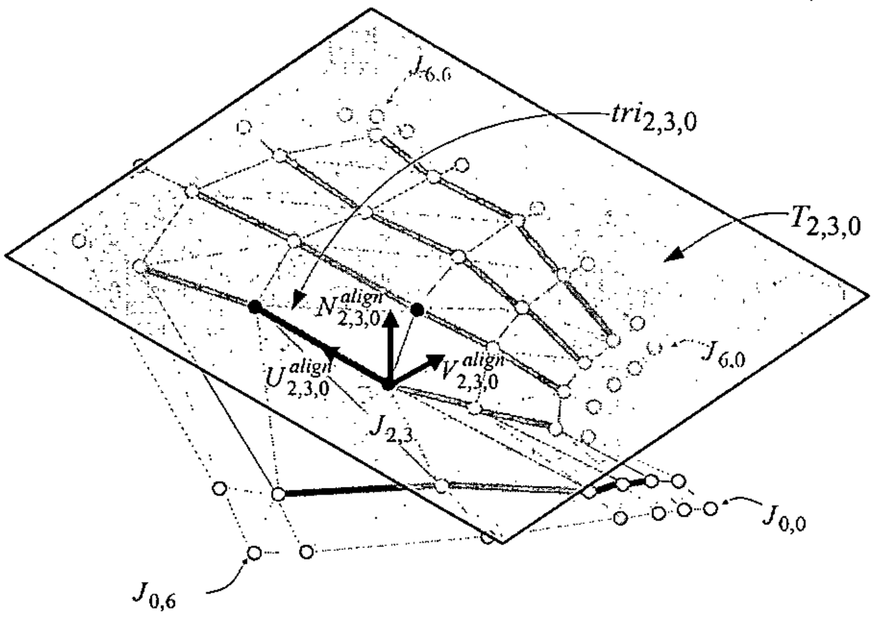


Figure 5-4 2D local coordinate of the plane $T_{i,j,\#}$.

5.2.1.2 Clipping Test

Notice that each projected ray will intersect with any of the $6 \times 6 \times 2$ planes $T_{i,j,\#}$ as long as the ray is not parallel to the planes. However, some of the intersections occur outside the area of the triangle $tri_{i,j,\#}$ and the one going across within the triangle area is the ***ray-intersection point***. To determine the *ray-intersection point*, clipping test is necessary to examine whether V_a^{proj} falls inside $tri_{i,j,\#}$.

To proceed with the clipping test, we must first aligned the *ray-intersection point* to the 2D local coordinate defined by the intersecting plane. Let the Euclidean 3D coordinate of the *ray-intersection point* of V_a at the plane $T_{i,j,\#}$ be $V_a^{proj}(i,j,k)$. And the 2D local coordinate of $T_{i,j,\#}$ is defined by the normal $N_{i,j,\#}^{align}$, and two perpendicular vectors, $U_{i,j,\#}^{align}$ and $V_{i,j,\#}^{align}$, on the plane (Figure 5-4).

- ➢ $N_{i,j,\#}^{align}$ is the normal of the plane $T_{i,j,\#}$;
- ➢ $U_{i,j,\#}^{align}$ equals to the unit vector of one of the edge of the triangle $tri_{i,j,\#}$ and the edge has one of it's endpoint at $J_{i,j}$;
- ➢ $V_{i,j,\#}^{align}$ is the cross product of $N_{i,j,\#}^{align}$ and $U_{i,j,\#}^{align}$.

The alignment can thus be achieved by applying a transformation matrix as follows [Hearn94],

$$\begin{bmatrix} V_a^{align}.i \\ V_a^{align}.j \\ V_a^{align}.k \\ 1 \end{bmatrix} = \begin{bmatrix} U_{i,j,\#}^{align}.i & U_{i,j,\#}^{align}.j & U_{i,j,\#}^{align}.k & -O_{i,j,\#}^{align}.i \\ V_{i,j,\#}^{align}.i & V_{i,j,\#}^{align}.j & V_{i,j,\#}^{align}.k & -O_{i,j,\#}^{align}.j \\ N_{i,j,\#}^{align}.i & N_{i,j,\#}^{align}.j & N_{i,j,\#}^{align}.k & -O_{i,j,\#}^{align}.k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_a^{proj}.i \\ V_a^{proj}.j \\ V_a^{proj}.k \\ 1 \end{bmatrix}$$

where $V_a^{align}(i,j,k)$ is the aligned coordinate of the *ray-intersection point* of vertex V_a on the plane $T_{i,j,\#}$. $O_{i,j,\#}^{align}(i,j,k)$ is the 3D coordinate of the point $J_{i,j}$ and is the origin of the local 2D coordinate of $T_{i,j,\#}$. Since the 3D coordinate $V_a^{proj}(i,j,k)$ is already a point on the plane $T_{i,j,\#}$, $V_a^{align}.k$ should become 0 after the alignment. The 2D local coordinate of the *ray-intersection point* is thus given by $V_a^{align}(i,j)$. For easy reference, we rewrite it to be $P_a(x,y)$.

With such alignment, all the data points concerned in the clipping test fall on the 2D space, or more specifically, the 2D local coordinate system defined by the plane $T_{i,j,\#}$. That is, the test is simplified to a 2D clipping test instead of 3D. We then employ the intersection calculation from [Spoelder90] to facilitate the test. First of all, we create a line, l_a, from the origin of the local 2D coordinate to the aligned coordinate $P_a(x,y)$. Then, the intersection points of l_a with each of the 3 edges of the triangle $tri_{i,j,\#}$ are determined. By examination of the nature of these intersections, we can easily decide whether the aligned coordinate $P_a(x,y)$ falls inside the triangle.

5.2.1.3 Determination of the Parametric Coordinate

Consider a triangle tri with vertices J_a, J_b and J_c. It is always possible to express a fourth point P, in the same plane as T, as a barycentric combination of J_a, J_b and J_c [Farin97].

$$P = \alpha J_a + \beta J_b + \gamma J_c \qquad \text{where } \alpha + \beta + \gamma = 1 \qquad (5\text{-}5)$$

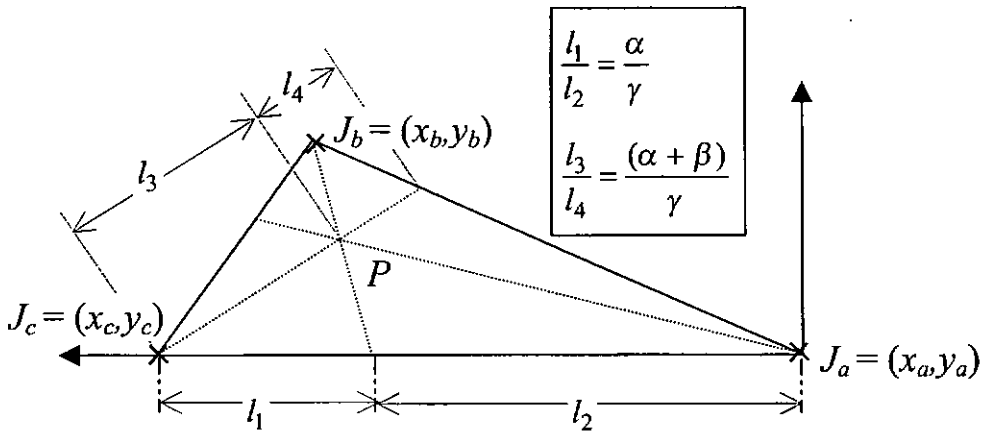
The coefficients (α, β, γ) are called **barycentric coordinates** of P with respect to J_a, J_b and J_c. Equation (5-5) can be viewed as a linear system of three equations. In the object space, we can obtain the 2D local coordinates of J_a, J_b, J_c and P by projecting their 3D coordinates to the 2D space defined by T. The *barycentric coordinates* (α, β, γ) of P is followed by an application of the Cramer's rule:

$$\alpha = \frac{area(P,J_b,J_c)}{area(J_a,J_b,J_c)},\ \beta = \frac{area(J_a,P,J_c)}{area(J_a,J_b,J_c)},\ \gamma = \frac{area(J_a,J_b,P)}{area(J_a,J_b,J_c)} \qquad (5\text{-}6)$$

where $area(J_a, J_b, J_c)$ represents the area of the region enclosed by vertices J_a, J_b and J_c. Actually, Cramer's rule makes use of determinants. They are related to area of triangles by the identity

$$area(J_a, J_b, J_c) = \frac{1}{2}\begin{vmatrix} x_a & x_b & x_c \\ y_a & y_b & y_c \\ 1 & 1 & 1 \end{vmatrix} \qquad (5\text{-}7)$$

where the vertices J_a, J_b and J_c have the local 2D coordinates of (x_a, y_a), (x_b, y_b) and (x_c, y_c) respectively. The local 2D coordinate is defined such that x_a, y_a and y_b equal 0 to simplify the calculation (Figure 5-5). Note that in order for Equation (5-6) to be well defined, $area(J_a, J_b, J_c)$ must not equal to 0, which means that J_a, J_b and J_c must not lie on a straight line. And this is already guaranteed in the *3D triangulated hand model.*



$\frac{l_1}{l_2} = \frac{\alpha}{\gamma}$
$\frac{l_3}{l_4} = \frac{(\alpha+\beta)}{\gamma}$
l_4
l_3
$J_b = (x_b, y_b)$
P
$J_c = (x_c, y_c)$
$J_a = (x_a, y_a)$
l_1
l_2

Figure 5-5 Barycentric Coordinates.

After the *barycentric coordinates* are determined, we can apply Equation (5-4) to determine the mapped parametric coordinate of the vertex.

$$P(u,v) = \alpha J_a(u_a, v_a) + \beta J_b(u_b, v_b) + \gamma J_c(u_c, v_c) \qquad (5\text{-}8)$$

5.2.2 Acceleration: Two-Pass Projection

The *3D triangulated hand model* consists of $6 \times 6 \times 2$ triangles and each of them defines a plane. Therefore, in the worst case, for an object consisting of n vertices, the mapping process could contain a maximum of $6 \times 6 \times 2 \times n$ projections, intersection test and clipping tests. This process could be very computationally expensive. To minimize the number of projections and tests, we separate the process into two steps. In pass 1, a test is set up for determining the triangle that the ray intersects. In pass 2, the *ray-intersection point*, where a given ray intersects the *3D triangulated hand model*, is determined.

Pass 1: Determination of the Intersection Triangle

The first step is to determine which triangle a ray intersects at the *3D triangular hand model.*

This is done by projecting a ray from an object vertex to a single plane instead of directly onto each of the $6 \times 6 \times 2$ triangles. First of all, we set up a plane π which intersects the centre of the palm $J_{3,3}$. The plane has the same normal as that of the centre of palm, $N_{3,3}$. Then, we ray-project all 7×7 data points onto the plane π and a corresponding *2D base mesh* on the plane is constructed. For clear illustration, Figure 5-6 shows the projection of 5×5 data points only.

Each object vertex V_a is then ray-projected to the plane π. Followed by a similar transformation and clipping test as described in section 5.2.1.2, the triangle on the *2D base mesh* that the ray intersects could be determined. Notice that the intersection of a triangle on the *2D base mesh* of plane π indicates the intersection of the corresponding triangle on the *3D triangulated hand model* (Figure 5-6). Therefore, by this initial projection, we can determine which triangle of the *3D triangulated hand model* the ray intersects in a single projection together with the clipping process. As a result, the number of projection process could be effectively reduced.

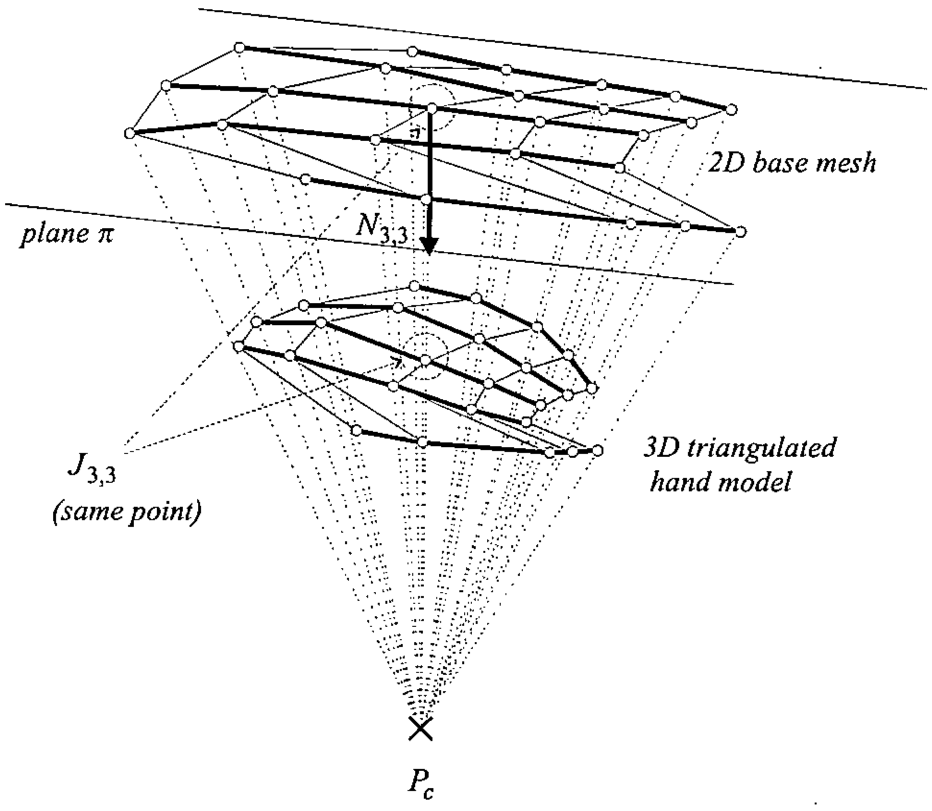


Figure 5-6 Two-Pass Projection.

Pass 2: Determination of Ray-Intersection Point

A second ray-projection is then made to find out the real intersection point on the corresponding triangle of the *3D triangulated hand model* by solving k in Equation (5-3). After determined the intersection point, its 3D position is transformed to the 2D local coordinate defined by the plane $T_{i,j,\#}$ and the parametric coordinate (u,v) of the vertex is evaluated by the *barycentric coordinate* method as mentioned earlier.

Chapter 6

Realistic Sculpturing Transformation

After the mapping is set up by the *ray-projection* method described in chapter 5, we present in this chapter the detail of the shape modification/deformation algorithm which we refer to as the *Sculpturing Transformation.* Moreover, we propose methods for flexible *sculpturing region control* that can be applied to the object being deformed.

6.1 Sculpturing Transformation

Consider at time 0 which is the instance when an object is mapped to the *control hand surface*, the initial position of a vertex is V_a^0. Let the parametric coordinate mapped by the *ray-projection* method of vertex V_a^0 be (u_a, v_a) and the corresponding **mapped ray-intersection point** on the *control hand surface* be $S^0(u_a, v_a)$.

Now, we write the position of the vertex as

$$V_a^0 = S^0(u_a, v_a) + \left(V_a^0 - S^0(u_a, v_a)\right) \tag{6-1}$$

where $V_a^0 - S^0(u_a, v_a)$ is the vector from the *mapped ray-intersection point*

$S^0(u_a, v_a)$ to the object vertex V_a^0.

By rewriting Equation (6-1) as

$$V_a^0 = S^0(u_a, v_a) + d_a^0 \cdot \lfloor V_a^0 - S^0(u_a, v_a) \rfloor_I \qquad (6\text{-}2)$$

where $\lfloor V_a^0 - S^0(u_a, v_a) \rfloor_I$ is the unit vector of the vector from $S^0(u_a, v_a)$ to V_a^0, and $d_a^0 = \left| V_a^0 - S^0(u_a, v_a) \right|$ is the magnitude of the same vector.

By the definition of *ray-projection*, the unit vector $\lfloor V_a^0 - S^0(u_a, v_a) \rfloor_I$ equals $\lfloor P_c - S^0(u_a, v_a) \rfloor_I$. Thus, at time 0, the relationship of the object vertex V_a^0 and the *mapped ray-intersection point* $S^0(u_a, v_a)$ can be rewritten as

$$V_a^0 = S^0(u_a, v_a) + d_a^0 \cdot \lfloor P_c - S^0(u_a, v_a) \rfloor_I \qquad (6\text{-}3)$$

d_a^0 is the distance between the *mapped ray-intersection point* and the object vertex at the time the mapping is established, i.e. time 0. This parameter implies the relationship of the mapping in the sense of keeping the distance between the object vertex and the *mapped ray-intersection point* on the *control hand surface* unchanged over the process of deformation.

After the relationship is established, deformation can be initiated by changing the hand gesture i.e. $S(u, v)$. Let us consider the situation at time t, the change of the *control hand surface* leads to the change of the Euclidean 3D coordinate of the *mapped ray-intersection point* to $S^t(u_a, v_a)$. The *sculpturing transformation* for the object vertex at time t can be evaluated accordingly as

$$V_a^t = S^t(u_a, v_a) + d_a^0 \cdot \lfloor P_c - S^t(u_a, v_a) \rfloor_I \qquad (6\text{-}4)$$

Recall that the *control hand surface* is constructed with a border part by extending x unit(s) from the edge of the initial finger joint data (section 4.3.1). This is used to prevent undesirable sudden change (Plate 6-1) and to maintain the continuity at the edge when local deformation is applied to an object. Figure 6-1 shows the normal object vertex transformation and also the border case.

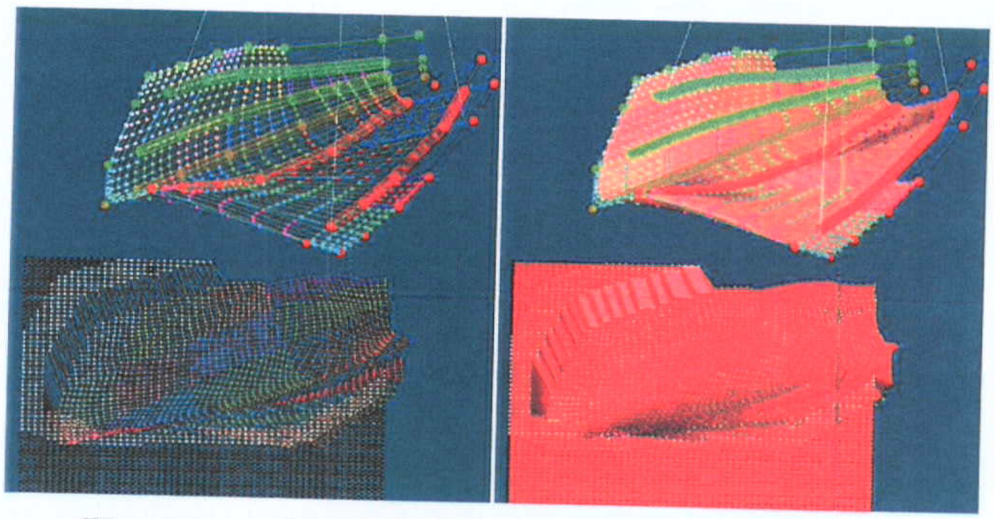


Plate 6-1 Sudden change created Without Weightings.

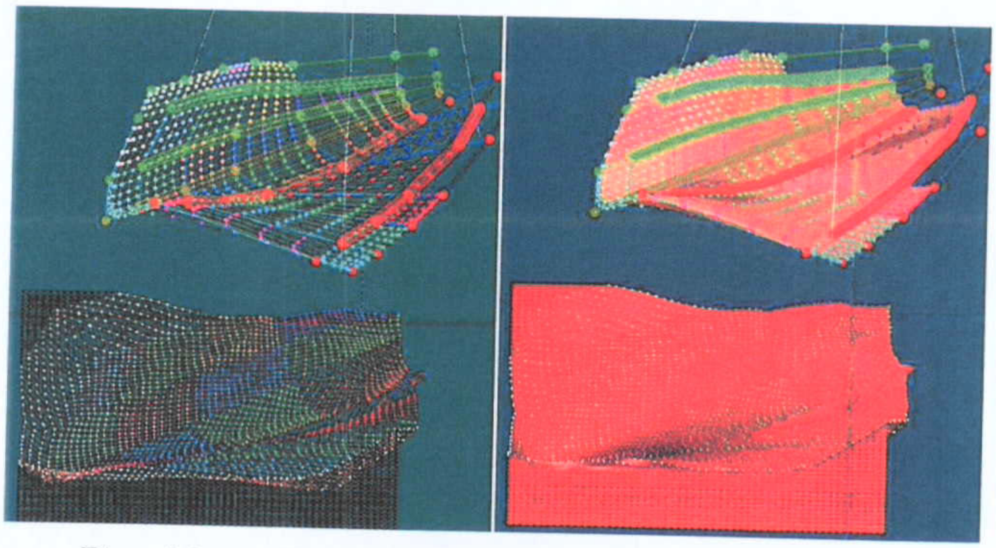


Plate 6-2 Weighted Edge of the *Control Hand Surface.*

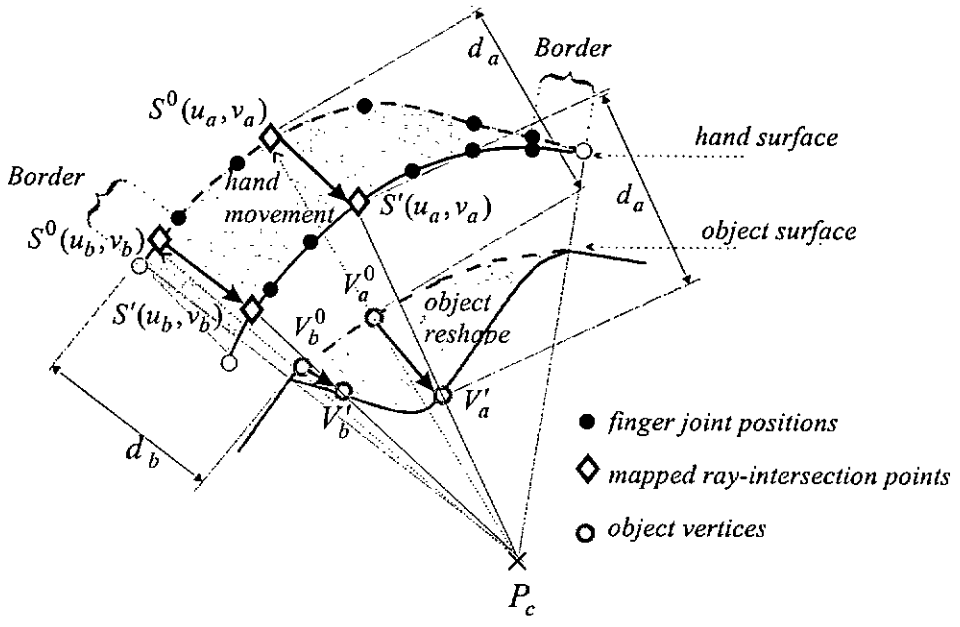


Figure 6-1 Object Vertex Transformation.

At the border, where $u \in [0,\overline{u}_1]$, $u \in [\overline{u}_5,1]$, $v \in [0,\overline{v}_1]$ or $v \in [\overline{v}_5,1]$, we introduce weightings, w, which are defined by equations of the cubic Bernstein-Bézier basis functions (Equations 6-5, 6-6). These weightings are defined to smoothly join the edges of the deformation region by a continuous function. By applying the deformation to these weightings, the deformation effect on the object being deformed will be progressively alleviated (Plate 6-2).

To simplify the discussion, we illustrate the explanation in 1D instead of 2D. Consider in the U-dimension, the border is defined at $u \in [0,\overline{u}_1]$ or $u \in [\overline{u}_5,1]$. We consider the region $u \in [\overline{u}_1,\overline{u}_5]$ as the *effective region R_e*, and $u \in [0,\overline{u}_1]$ and $u \in [\overline{u}_5,1]$ are the *border regions R_b*. Previously, w equals to 1 whenever u falls into any of these regions, otherwise, w equals to 0. This leads to a sudden change at the edge where u equals to 0 or 1. To preserve the continuity of the deformation of the object, a continuous weighting w (Figure 6-2) is introduced in the border region.

$$w(u)=\begin{cases}0 & u\notin[0,1]\\ B_{2,3}\left(\dfrac{u}{\bar{u}_1}\right)+B_{3,3}\left(\dfrac{u}{\bar{u}_1}\right) & u\in[0,\bar{u}_1]\\ 1 & u\in[\bar{u}_1,\bar{u}_5]\\ B_{2,3}\left(\dfrac{u-\bar{u}_5}{1-\bar{u}_5}\right)+B_{3,3}\left(\dfrac{u-\bar{u}_5}{1-\bar{u}_5}\right) & u\in[\bar{u}_5,1]\end{cases} \tag{6-5}$$

where $B_{i,3}(u)=\begin{pmatrix}3\\i\end{pmatrix}(1-u)^{3-i}u^i$ with $u\in[0,1]$ and $i=2,3$.

In fact, w can be represented as a Bézier curve by introducing related control points at $(0,0),(\frac{\bar{u}_1}{3},0),(\frac{2\bar{u}_1}{3},1)$ and $(\bar{u}_1,1)$ over the interval $[0,\bar{u}_1]$. (Figure 6-4).

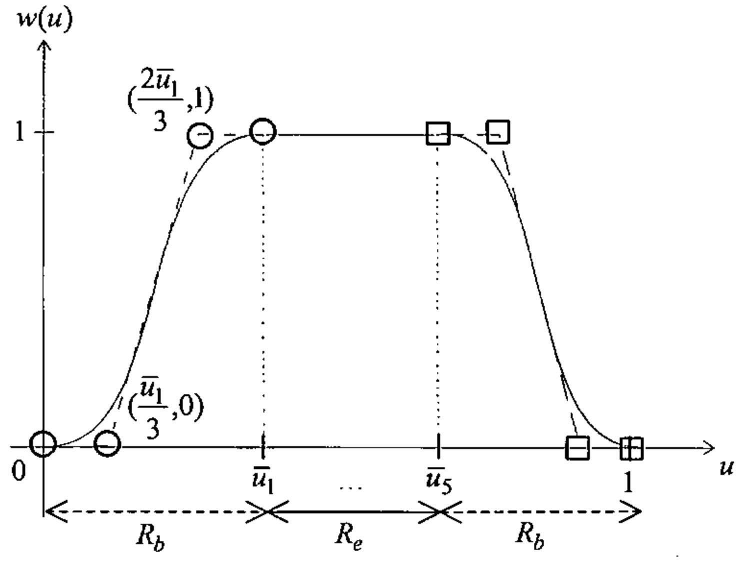


Figure 6-2 1D illustration of the smooth weighting function $w(u)$ and the Bézier representation at the edges.

In the 2D case, the weighting function is a generalized tensor product function of $w(u)$ and $w(v)$. The smooth weighting function $w(u,v)$ is 1 when it is mapped to the *effective region* $[\bar{u}_1,\bar{u}_5;\bar{v}_1,\bar{v}_5]$ and 0 when outside the whole parametric span [0,1;0,1] of the *control band surface*. Over the *border regions,* bicubic Bézier functions are introduced to smoothly introduce the deformation effect (Figure 6-3). Hence, $w(u,v)$ are given by

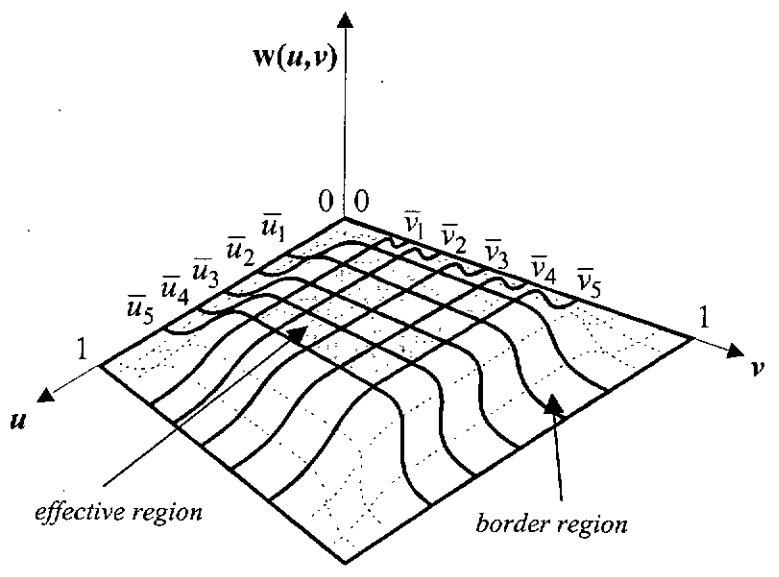


Figure 6-3 2D illustration of the smooth weighting function *w(u,v)*.

$$
w(u,v)=\begin{cases}
0 & u \notin [0,1;0,1] \\
1 & u \in [\bar{u}_1,\bar{u}_5;\bar{v}_1,\bar{v}_5] \\
B_{2,3}\left(\dfrac{u}{\bar{u}_1}\right)+B_{3,3}\left(\dfrac{u}{\bar{u}_1}\right) & (u,v)\in[0,\bar{u}_1;\bar{v}_1,\bar{v}_5] \\
B_{0,3}\left(\dfrac{u-\bar{u}_5}{1-\bar{u}_5}\right)+B_{1,3}\left(\dfrac{u-\bar{u}_5}{1-\bar{u}_5}\right) & (u,v)\in[\bar{u}_5,1;\bar{v}_1,\bar{v}_5] \\
B_{2,3}\left(\dfrac{v}{\bar{v}_1}\right)+B_{3,3}\left(\dfrac{v}{\bar{v}_1}\right) & (u,v)\in[\bar{u}_1,\bar{u}_5;0,\bar{v}_1] \\
B_{0,3}\left(\dfrac{v-\bar{v}_5}{1-\bar{v}_5}\right)+B_{1,3}\left(\dfrac{v-\bar{v}_5}{1-\bar{v}_5}\right) & (u,v)\in[\bar{u}_1,\bar{u}_5;\bar{v}_5,1] \\
\sum_{i=2}^{3}\sum_{j=2}^{3} B_{i,3}\left(\dfrac{u}{\bar{u}_1}\right)B_{j,3}\left(\dfrac{v}{\bar{v}_1}\right) & (u,v)\in[0,\bar{u}_1;0,\bar{v}_1] \\
\sum_{i=0}^{1}\sum_{j=2}^{3} B_{i,3}\left(\dfrac{u-\bar{u}_5}{1-\bar{u}_5}\right)B_{j,3}\left(\dfrac{v}{\bar{v}_1}\right) & (u,v)\in[\bar{u}_5,1;0,\bar{v}_1] \\
\sum_{i=2}^{3}\sum_{j=0}^{1} B_{i,3}\left(\dfrac{u}{\bar{u}_1}\right)B_{j,3}\left(\dfrac{v-\bar{v}_5}{1-\bar{v}_5}\right) & (u,v)\in[0,\bar{u}_1;\bar{v}_5,1] \\
\sum_{i=0}^{1}\sum_{j=0}^{1} B_{i,3}\left(\dfrac{u-\bar{u}_5}{1-\bar{u}_5}\right)B_{j,3}\left(\dfrac{v-\bar{v}_5}{1-\bar{v}_5}\right) & (u,v)\in[\bar{u}_5,1;\bar{v}_5,1\}
\end{cases}
\qquad (6\text{-}6)
$$

And the *sculpturing transformation* is updated as shown below.

$$V_a^t = w(u_a, v_a) \cdot \left(S^t(u_a, v_a) + d_a^0 \cdot \left[P_c - S^t(u_a, v_a)\right]_I\right) \\ + \left(1 - w(u_a, v_a)\right) \cdot V_a^0 \qquad (6\text{-}7)$$

6.2 Sculpturing Region Control

With *ray-projection*, the sculpturing region can be very flexible. It can change either by changing the hand gesture or by changing the location of the centre of projection. By changing the location of the centre of projection, three possible deformation effects may be resulted.

6.2.1 Parallel Ray-Projection

When P_c is set to infinity, all the projection rays may be considered as parallel to each other. This is referred to as parallel projection (Figure 6-4a), the sculpturing region will have the same size as the hand surface. Under this circumstance, $d_a^0\left[P_c - S^t(u_a, v_a)\right]_I$ equals $V_a - S(u_a, v_a)$, and the *sculpturing transformation* becomes

$$V_a^t = S^t(u_a, v_a) + (V_a - S(u_a, v_a)) \qquad (6\text{-}6)$$

That is, $\Delta V_a = \Delta S(u_a, v_a)$. This implies that the change of the *control hand surface* applies directly to the vertices of the object. This leads to a similar effect as touching and deforming the object.

6.2.2 Reduced Sculpturing

Reduced Sculpturing is produced by setting the centre of projection, P_c, in front of both the *control hand surface* and the object being deformed. In Figure 6-4b, a reduced projection area of the *control hand surface* is shown to cast on the object. In fact, the object is mapped to the converged image of the *control hand surface*. As a result, the magnitude of the change in hand gesture passed to the object is reduced. In another words, a *reduced sculpturing* effect is produced. Experimental results of this effect are demonstrated in Plate 1.

6.2.3 *Magnified Sculpturing*

Magnified Sculpturing, on the other hand, is produced by setting the centre of projection, P_c, to the back of the hand surface. As opposite to the above case, an enlarged projection area of the *control hand surface* will be cast on the object (Figure 6-4c). In this case, the magnitude of the change in hand gesture passed to the object will be magnified. We label this effect as the *magnified sculpturing*. Plate 2 shows the experimental result.

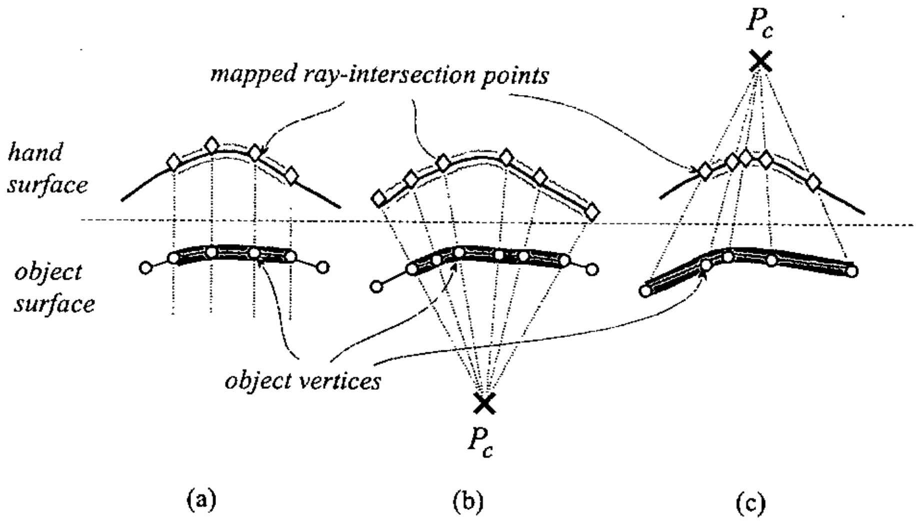


Figure 6-4 Sculpturing Regions – (a) Parallel Ray-Projection; (b) Reduced Sculpturing; (c) Magnified Sculpturing.

Chapter 7

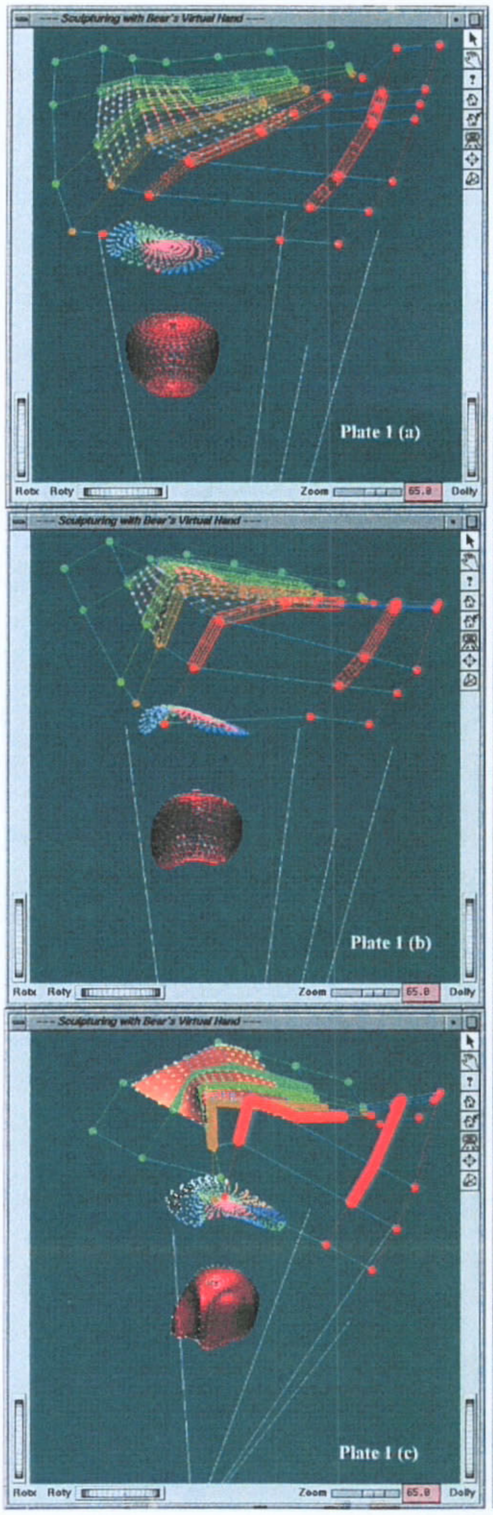
Output & Performance

Our project develops an intuitive method for interactive 3D sculpturing in virtual reality environment based on the use of a sensor glove. To examine the idea and test for the performance, we have implemented the algorithm in C++ with OpenInventor and OpenGL on a SGI Indigo^2 workstation with a 200MHz MIPS 4400 CPU and the Extreme graphics accelerator. In this chapter, we present some output models from the prototype and analyse the performance of the algorithm.

7.1 Prototype

We have tested the system with different object models and some of the results are shown in Plates 7-1 and 7-2. Plate 1 shows the sequence of grasping of an apple model and Plate 7-2 demonstrates the deformation of a human face model. Each diagram contains the *control hand surface*, the object being deformed and a colored cluster in the middle.

On the *control hand surface*, there are 5 colored stick segments which represent the five fingers, the joint positions, are represented by colored stick and balls. The red ones are from the thumb and the green ones are from the pinkie. The *control hand surface* is divided into tetrahedral net. Each of them is rendered by a brass colored surface and is distinguished by different colored points distributed across it. For clear illustration, only those regions that are mapped with object vertices are rendered.



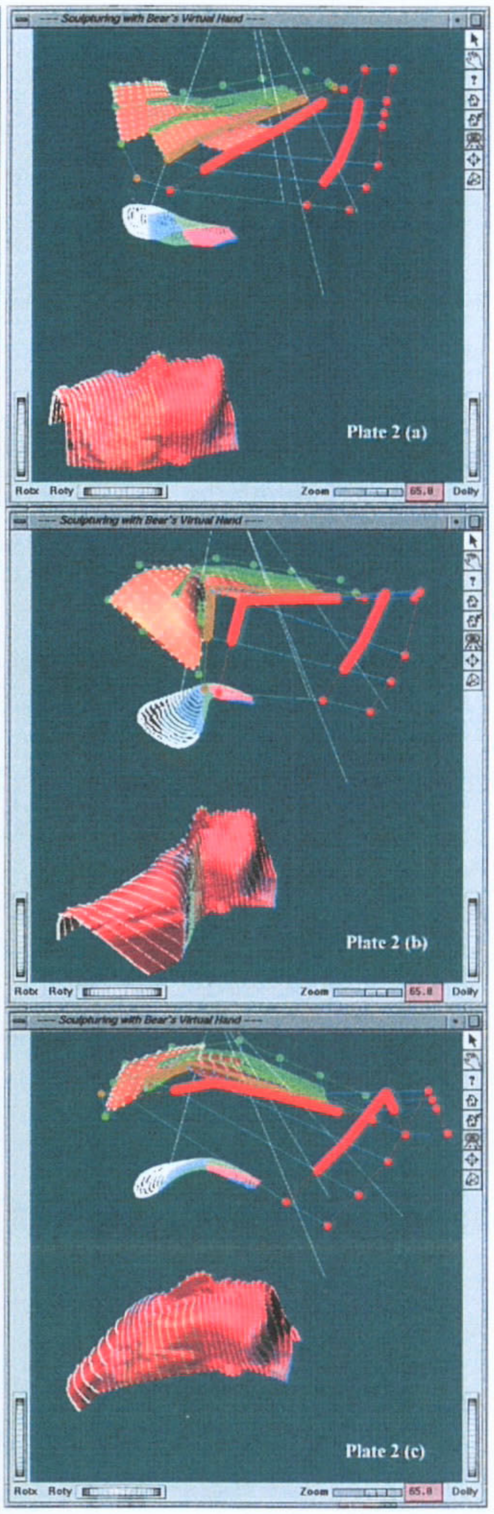


Plate 7-1 Apple Model (Reduced Sculpturing)

Plate 7-2 Human Face Model (Magnified Sculpturing)

The objects we used for the experiments are in the form of triangular meshes. The model is rendered with specific color. The colored points in the object represent the object vertices that are mapped to the *control hand surface*. Colored points can also be found on the *control hand surface*. This is to help the users recognise the correspondence between the *control hand surface* and the *effective region* that can be applied to the object. (In case the correspondence is still not clear, the cluster of the colored points located between the *control hand surface* and the object may help.)

- White points: Indicate those vertices mapped to the *border region*.
- other colored points: Reveal the control of different regions of the *control hand surface* to the object.

There are also white rays on the diagrams which converge to the centre of projection, P_c. The centre of projection in our method can be set flexibly to achieve different control to the object (chapter 6). In Plate 7-1, the centre of projection is set in front of both the hand and the object and thus producing a *reduction* control while in Plate 7-2, the centre of projection is set at the back of the hand and thus producing *magnifying* effect.

Plates 7-1a and 7-2a reveal the mapping of the *control hand surface* to the object. The models are in their initial shape. Plates 7-1b, 7-1c, 7-2b and 7-2c show the sculpturing effect induced by changing of the hand gesture.

7.2 Outputs

Using our prototype, we have experienced the deformation with different models. Plate 7-3 shows some of the output models captured from the experiments. The leftmost models show the initial shape of the object and the others shows the deformed ones.

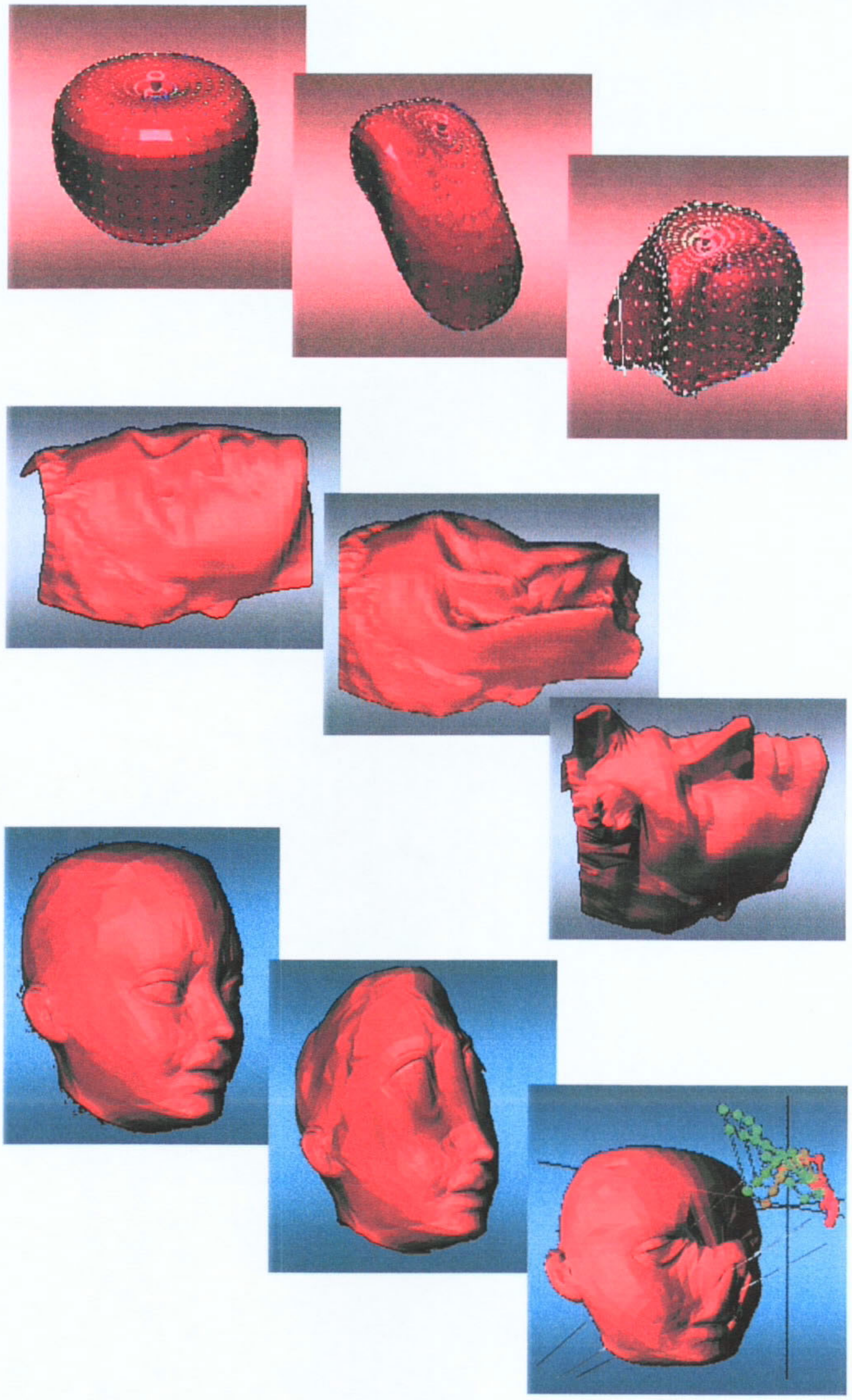


Plate 7-3 Outputs Models.

7.3 Performance

To evaluate the performance of our system, we measured the time for the two main stages during sculpturing, the ***mapping of object vertices to the control hand surface***, and the ***object modification process***. Some of the results are illustrated in Table 7-1 and Table 7-2.

Model	No. of Vertices	Total Time	Average time (per vertex)
Button	701	0.35s	0.50ms
Teapot (high res.)	2081	1.00s	0.48ms
Teapot (low res.)	529	0.26s	0.49ms
Apple	867	0.45s	0.52ms
Face	2278	0.86s	0.38ms

Table 7-1 Performance of Stage 1 – Mapping of Object Vertices to the *Control Hand Surface*.

Model	No. of Vertices	Total Time	Average Time (per vertex)
Button	701	0.035s	0.050ms
Teapot (high res.)	2081	0.110s	0.055ms
Teapot (low res.)	529	0.030s	0.055ms
Apple	867	0.050s	0.050ms
Face	2278	0.130s	0.055ms

Table 7-2 Performance of Stage 2 – Object Modification Process.

From Table 7-1 and Table 7-2, the performance of stage 1: *mapping object vertices to the control hand surface* is at about 0.5ms per vertex and that of stage 2: *object modifying process* is at about 0.05ms per vertex. Also, by employing the effective interpolation method described in chapter 4, the time required for constructing the *control hand surface* is only 0.005s to 0.01s.

These results show that our method is able to provide a real time interactive experience for object deformation and we believe that the new method is efficient enough for *Virtual 3D Sculpturing.*

Chapter 8

Extension & Applications

Currently we have implemented a prototype to test the feasibility and efficiency of the algorithm. However, in order to allow the algorithms be used in the real world applications, we have the following suggestions for extending the algorithms to a more complete deformation system.

8.1 Extension

8.1.1 Mode Selection Interface

Currently, the prototype of the algorithm makes use of mouse and keyboard trigger to confirm the mode change in the course of the deformation. This limits the ease of use of the system. To further improve the intuitivity of the interface, we suggest to have mode selection implemented with gesture recognition. The input could be captured from the other hand which is free from the deformation tasks in our work.

8.1.2 Resolution Refinement

The reader may find some notable discrepancies in some of the deformed polygonal output models. This is because for all polygonal models, the resolution cannot be refined automatically when the model is being deformed substantially. This problem exists in all kinds of deformation algorithms for deforming polygonal models. In order to provide better output models, further works should be done to facilitate automatic resolution refinement of the deformed polygonal models.

8.1.3 Sculpturing Level Control

With our method, the sculpturing level, which refers to the fineness of the deformation, can be controlled along with the sculpturing region by adjusting the position of centre of projection as stated in section 6.2. Also, scaling up or down the proportion between the *control hand surface* and the object model can attain coarser or finer deformation. We believe by providing effective switching function for changing the sculpturing scales, the systems can further improved to accomplish a variety of sculpturing level experience.

8.2 Applications

Nowadays, more and more industrial products are being designed with artistic free-form or sculptured models in order to make the look better or even function better. Sculptured surfaces in household products such as telephones serve to enhance their appearance, while these in aero-dynamic components such as turbine blade are to meet the functional requirements. However, it is difficult for the designer to model such a free-form model with conventional 3D mechanical CAD systems. This is because these systems make use of 2D input devices, like mouse. They rely on mapping of 3D motion to a 2D device. The operations become complicated and are difficult for the designer to use.

The *Virtual 3D Sculpturing* technique proposed in this dissertation serves as an alternative to the existing methods. The interactive and realistic deformation experience produces the reality of getting something in contact with the hand, which is important in stimulating one's creativity. In addition of the intuitive interface, our algorithm also provides a basis tools for *Virtual 3D Sculpturing* systems so that traditional designers do not need to learn any geometric knowledge of computer graphics in advance. Besides, traditional methods of sketches and physical mock-ups used in presenting the products or ideas cannot be directly utilized in other automatic manufacturing processes. With our method, well-defined data models can be produced. Moreover, our method could simplify the task of alteration of existing design. In fact, alteration to artistic free-form mock-up is nearly impossible since to modify a physical mock-up usually means to reproduce a new one.

More importantly, the potential of *Virtual 3D Sculpturing* can go beyond real simulation. In traditional sculpturing, the properties of object force a designer to consider issues of construction, strength of materials, ergonomics, methods and process of finishing. We suggest that further work can be done to exploit the immateriality of virtual sculpturing which allows the designer to accomplish design work without considering the real world constraints.

Chapter 9
Conclusion

This dissertation presents the *Virtual 3D Sculpturing* method. It is basically an object deformation algorithm using the glove device as an input tool.

The main idea of our method is to create a parametric *control hand surface* defined by an *open-uniform B-Spline tensor product surface.* By making use of the properties of the parametric representation of the surface, we simplify the clumsy processes necessary in the construction of the *control hand surface.* Further, numerical methods are introduced to optimize the calculations.

After the *control hand surface* is set up, each of the geometric attributes of the object in the Euclidean 3D space will be mapped to the parametric domain of the *control hand surface* through a novel *ray-projection* method. This mapping introduces a relationship between the object and the *control hand surface.* By maintaining the distance between the mapped pairs, change of hand gesture can be efficiently passed to control the deformation of the object. In addition, with the flexible ray-projection mapping, the algorithm is able to produce different levels of effect, which is a valuable feature for *virtual sculpturing.*

As demonstrated by the experimental results, the new method is very efficient and the deformation operation is intuitive to use. We suggest that this algorithm could be used in virtual reality applications such as free-form product design.

References

[Brooks89] M. Brooks, "The DataGlove as a Man-Machine Interface for Robotics," *Tech. Memorandum*, AT&T Bell Laboratory, Holmdel, N.J., September 19, 1989.

[Burdea92] G. Burdea, J. Zhuang, E. Roskos, D. Silver and N. Langrana, "A Portable Dextrous Master with Force Feedback," *Presence: Teleoperators and Virtual Environments*, **1**(1), pp.18-28, 1992.

[Coquillart90] S. Coquillart, "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modelling," *ACM Computer Graphics*, **24**(4), pp.187-193, August 1990.

[*Cyber*Glove94] *Cyber*Glove™ *User's Manual*, Copyright © 1991-1994 Virtual Technologies.

[Engeln96] G. Engeln-Mullges and F. Uhlig, *Numerical Algorithms with C.* Springer, pp.89-92, 1996.

[Farin97] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide*, 4th Edition, Academic Press, 1997.

[Galyean91] T. A. Galyean and J. F. Hughes, "Sculpting: An interactive volumetric modelling technique," *ACM Computer Graphics*, **25**(4), pp.267-274, 1991.

[Grimes83] G. J. Grimes, "Digital Data Entry Glove Interface Device," *Bell Telephone Laboratories*, Murray Hill, N.J., US Patent 4,414,537, November 8 1983.

[Hand97] C. Hand, "A Survey of 3D Interaction Techniques," *Computer Graphics Forum*, **16**(5), pp.269-281, 1997.

[Hearn94] D. Hearn and M.P. Baker, *Computer Graphics*, 2nd Edition, Prentice Hall, 1994.

[Hong89] J. Hong and X. Tan, "Calibrating a VPL DataGlove for Teleoperating the Utah/MIT Hand," *Proceedings of IEEE International Conference, Robotics and Automation*, Vol. 3, IEEE CS Press, Los Alamitos, California, pp.1752-1757, 1989.

[Hsu92] W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct Manipulation of Free-Form Deformation," *ACM Computer Graphics*, **26**(2), pp.177-184, July 1992.

[Kameyama97] K. Kameyama, "Virtual Clay Modelling System," *ACM Symposium on Virtual Reality Software and Technology*, pp.197-200, Sept. 1997.

[Kramer89] J. Kramer and L. Leifer, "The Talking Glove: An Expressive and Receptive 'Verbal' Communication Aid for the Deaf, Deaf-Blind, and Non-vocal", *Technical Report*, Stanford University, Department of Electronic Engineering. Standford, California, 1989.

[Kurmann96] D. Kurmann and M. Engeli, "Modelling Virtual Space in Architecture," *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pp.77-82, July 1996.

[Lamousin94] H. J. Lamousin and W. N. Waggenspack, "NURBS-Based Free-Form Deformations," *IEEE Computer Graphics and Applications*, pp.59-65, 1994.

[Li97] F. Li, R. Lau, and M. Green, "Interactive Rendering of Deformating NURBS Surfaces," *EUROGRAPHICS'97*, **16**(3), pp.C-47-56, September 1997.

[Liang93] J. Liang and M. Green, "Geometric modelling using six degrees of freedom input devices," *Proceedings of 3rd International Conference on CAD and Computer Graphics*, pp.217-222, August 1993.

[Morita91] H. Morita, S. Hashimoto, and S. Ohteru, "A Computer Music System that Follows a Human Conductor," *Computer*, **24**(7), pp.44-53, July 1991.

[Murakami94] T. Murakami and N. Nakejima, "Direct and Intuitive Input Device for 3-D shape Deformation," *Proceedings of ACM CHI'94*, pp.465-470, 1994.

[Pao89] L. Pao and T.H. Speeter, "Transformation of Human Hand Positions for Robotic Hand Control," *Proceedings of IEEE International Conference of Robotics and Automation*, Vol. 3, IEEE CS press, Los Alamitos, California, pp.1758-1763, 1989.

[Piegl95] L. Piegl and W. Tiller, *The NURBS Book*, Springer-Verlag, 1995.

[Sederberg86] T. W. Sederberg and S. R. Parry, "Free-Form Deformation of Solid Geometric Models," *ACM Computer Graphics*, 20(4), pp.151-160, August 1986.

[Shaw93] C. Shaw, M. Green, J. Liang, and Y. Sun, "Decoupled Simulation in Virtual reality with MR Toolkit," *ACM Transactions on Information Systems*, **11**(3), pp.135-142, May 1993.

[Spoelder90] H. J. W. Spoelder and F. H. Ulllings, "Two-Dimensional Clipping: A Vector-based Approach," *Graphics GEMS I*, Academic Press, pp.121-128, 1990.

[Spong89] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*, John Wiley and Sons Inc., 1989.

[Sturman92] D. J. Sturman, "Whole-Hand Input," *Doctoral dissertation*, MediaLab, Massachusetts Institute of Technology, Cambridge, Mass., February 1992.

[Sturman94] D. J. Sturman and D. Zeltzer, "A Survey of Glove-based Input," *IEEE Computer Graphics and Applications*, pp.30-39, 1994.

[*Virtual*Hand94] *Virtual*Hand^{TM} *Software Library Reference Manual*, Copyright © 1994 Virtual Technologies.

Brief Curriculum Vital

The author Wong Pui Yee, Janis is currently a research assistant at the Department of Computer Science at the City University of Hong Kong where she is working on the "Virtual Campus" project. Her research interests include virtual reality and computer animation.

Janis received a BA in Computer Studies from the Hong Kong Polytechnic University in 1995. She is presently registered as a MPhil. student in the Hong Kong Polytechnic University.

Readers may find online demonstration at http://hello.to/bearSculpt or contact Janis at janiswong@

Publications

Janis Wong and Rynson Lau, "Virtual 3D Sculpturing," *submitted to Journal of Visualization and Computer Animations* (1999)

Janis Wong, Rynson Lau, and Lizhuang Ma, "Engineering Design with a Virtual Hand Surface," *IEEE Third International Conference on Multi-Media Engineering and Education*, July 1998, pp. 178-186 (1998)

Janis Wong, Rynson Lau, and Lizhuang Ma, "Virtual 3D Sculpturing with a Parametric Hand Surface," *Computer Graphics International, IEEE Computer Society Press*, June 1998 (1988)

Lizhuang Ma, Rynson Lau, Jieqing Feng, Qunsheng Peng, and Janis Wong, "Surface Deformation Using the Sensor Glove," *ACM Symposium on Virtual Reality Software and Technology*, September 1997, pp. 189-196 (1997)

Rynson Lau, Mark Green, Danny To and Janis Wong, "Real-Time Continuous Multi-Resolution Method for Models of Arbitrary Topology," *Presence: Teleoperators and Virtual Environments, MIT Press, 7(1)*, February 1998 (1998)