Free-Form Deformation of Solid Models in CSR

LAI Chi-fai

A Thesis Submitted in Partial Fulfilment

Of the Requirements for the Degree of

Master of Philosophy

in

Mechanical and Automation Engineering

DECLARATION

I hereby declare that this M.Phil. thesis titled "Free-Form Deformation of Solid Models in CSR" is composed by myself and represents my own work. I also declare that the work reported in this thesis has not been previously included in a thesis, dissertation or report submitted to this University or any other institutions for a degree, diploma, or other qualification.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Hui Kin-Chuen, for his dedication and high standards of professional excellence that have been a constant source of encouragement and inspiration. He has provided much motivation, guidance and support for this work. Moreover, his high standard of technical writing has enhanced the presentation of the results in this thesis. I thank him for the time spent with me on my dissertation research.

In addition, I would also like to thank Adrian Bowyer, the developer of *"SvLis"*, who provide the modeler as a freeware for download.

Finally, I also wish to thank the Department of Mechanical and Automation Engineering, the Chinese University of Hong Kong, for providing computing facilities and financial support for carrying out this research.

ABSTRACT

Existing free-form deformation techniques deform an object by deforming the space enclosing the object. This provides good visual effect for the deformed objects. However, further operation such as *Boolean operations* on the deformed object may not be possible. This thesis is concerned with the techniques of applying free-form deformation on solid models represented by the Constructive Shell Representation (CSR). The primitives of CSR are truncated tetrahedron, called "trunctet", which is formed by intersecting a quadric surface with a tetrahedron. The quadric surface passes through two or three vertices of the tetrahedron. A CSR solid object is formed by the union and subtraction of trunctets from a polyhedron core.

By applying free-form deformation on the surface points of the trunctets of a CSR object so that the vertices and the quadric patch polynomial of the trunctets are changed, the shape of the object can be modified. The polyhedron core is also deformed. This technique can be used to deform globally smooth solid models or general solid models with sharp edges. The deformation can be applied either globally or locally. Techniques for the deformation are discussed in detail. Experiments are conducted and the results are also presented.

概論

現存之物件變形技術只能把物件所在的空間改變，雖然這種方法能造出不錯的視覺效果，但變形後的物件並不能和其他物件結合(例如不能使用 Boolean Operation)，這份論文便是針對這個問題提出解決方法，把"Free-Form Deformation (FFD)"變形技術應用於"Constructive Shell Representation (CSR)"上. CSR 的基本元件稱作"trunctet", Trunctet 是由一個四面体和一個二次元曲面相交而成. 一個CSR物件便是由一個多面体減去和結合 trunctet 所造成.

只要利用 FFD 改變 trunctet 表面上的點，同時改變多面体，便能造出整個物件的變形效果. 這種技術能應用於整個物件或物件某部份的變形. 詳細技術及實驗結果均詳述於本論文.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation and Objectives

Despite the popularity of the free-form deformation (FFD) techniques for generating special effects in graphics and animations, its use as a modeling tool has not been fully explored. This is a result of the fact that the deformed object is not converted to an existing object representation scheme. For instance, solid objects represented in Boundary representation (B-rep) or Constructive Solid Geometry (CSG) can be deformed using the FFD techniques. However, further operations such as Boolean operations on the deformed object cannot be performed since the deformed object is not converted to a B-rep or CSG model.

In order to overcome this problem, a proper representation scheme for solid model, especially free-form objects, is needed. Traditionally, free-form surfaces are represented as parametric patches of high degree in B-rep, which are very successful as far as design and rendering are concerned. However, manipulating and reasoning about physical objects with parametric patches poses fundamental difficulties. For example, the difficulty of evaluating and representing the intersection of parametric patches has hindered the development of solid modeling systems based on parametric patches. In this thesis, the Constructive Shell Representation (CSR) is adopted for representing solid

objects. This allows free-form solids to be represented as CSG models so that existing modeling techniques can be applied.

In Constructive Shell Representation, a free form solid model is represented by the union and subtraction of trunctets from a polyhedron core, where the trunctets form a "thick shell" that contains the free-form surface.

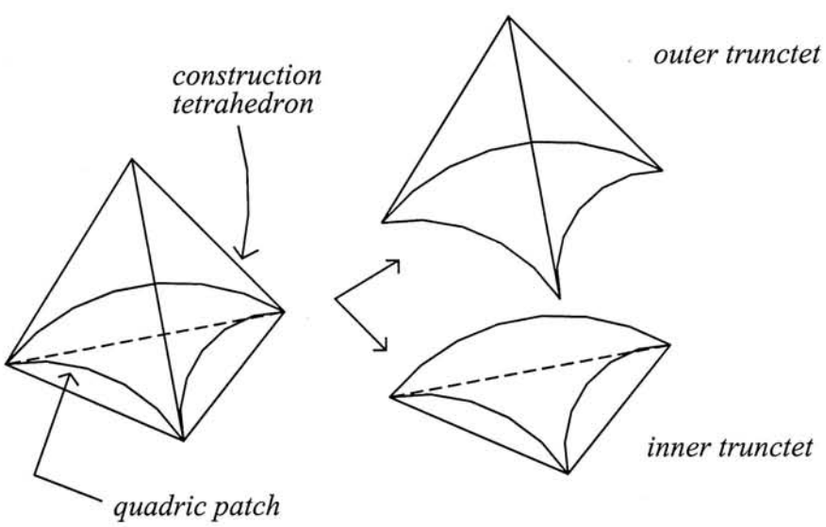


Figure 1.1 *Trunctets and its construction tetrahedron.*

One of the bounding faces of each trunctet is a quadric patch, which is a subset of the free-form surface; the other faces are planar. Wherever the surface is convex, the corresponding trunctet is unioned to the polyhedron core. Wherever the surface patch is concave, the corresponding trunctet is subtracted from the polyhedron core.

Traditional Constructive Solid Geometry (CSG) is weak in representing free-formed solid model. However, by combining CSG with CSR, most solid models can be represented.

In the deformation of solid models, the free-form deformation (FFD) technique presented by Thomas W. Sederberg [3] is adopted. In applying FFD to a solid modelled with Constructive Shell Representation, it is observed that there are several ways to combine FFD with CSR to achieve the objective of deforming a general solid model in a free-form manner. These approaches include the deformation of vertices and normals, the use of vertices' neighborhoods, and surface fitting approach. Details of these approaches will be discussed in Chapter 4.

The objective of this research is to develop an approach for the free form deformation of a CSR object so as to provide a complete representation of the solid model before and after the deformation. The three approaches as stated above are considered. Among these approaches, the surface fitting approach is found to be most effective. This approach can also be applied to ordinary CSG primitives by converting them into CSR first.

1.2 Thesis Organization

Chapter 2 describes the different deformation techniques and their major applications. Algebraic patch techniques is also given. Then, the essential background knowledge on CSR, algebraic patches and FFD is given in Chapter 3.

Chapter 4 discusses several possible ways to link CSR and FFD together. Then, the idea of connecting CSR and FFD by using surface fitting is presented. It starts with the deformation of a single trunctet. The use of double tetrahedrons, and their subdivision is discussed. An algorithm is presented. The method is then extended for general CSG solid models. Representation of the core of a CSR object is also presented.

Finally, Chapter 5 discusses the implementation of the proposed techniques. Experimental results are presented.

CHAPTER 2

RELATED WORKS

2.1 Deformation Techniques

In general, the deformation techniques can be classified into those using a deformation tool, and those working directly on the object.

2.1.1 Deformation Techniques Requiring a Deformation Tool

Deformation techniques requiring a deformation tool refer to those techniques that require a lattice or an axis. In these approaches, the deformed object is computed from the deformation applied to the deformation tool. These models involve a mapping represented by the deformation function D: $R^3 \rightarrow R^3$ that associates with each point $U \in R^3$ and its new position $D(U)$.

The deformation tool used by ***free-form deformation (FFD)***, which was developed by T. W. Sederberg and S. R. Parry in 1986 [3], is a parallelepipedical volume called *lattice*. To deform an object the user deforms the lattice by moving its control points. Any point lying inside the lattice is deformed according to the lattice deformation. In particular, the deformation of an object inside the lattice follows the displacement of the lattice control points.

The deformation of the object "follows" the displacement of the control points of the lattice. Thus, it is not easy to get a precise displacement of a given

point of the object. This technique is particularly well-suited to global deformation.

Extended Free-Form Deformation (EFFD), developed by S. Coquillart in 1990 [4], extends the FFD technique to allow non-parallelepipedical lattices. In particular, elementary or composite prismatic lattices are defined. Elementary prismatic lattices are obtained by moving or merging control points of a parallepipedical lattice. For example, the cylindrical lattice is obtained by welding two opposite faces of a parallelepipedical lattice and by merging all control points of the cylinder axis.

Rational Free-Form Deformation (RFFD), developed by P. Kalra, A. Mangili, N. M. Thalmann and D. Thalmann in 1992 [5], is also another extension of FFD. It allows incorporation of weights defined at each control point of the parallelepipedical lattice. However, when the weights at each control point are unity, the deformations are equivalent to that obtained with FFD. To control the deformation, the user either moves the lattice control points or modifies their associated weights. The use of a weight associated at each control point provides one more degree of freedom to define the deformation. However, the unpredictability of the deformation obtained by changing the weight at a control point could be a limitation of this technique.

In ***Axial Deformation***, developed by F. Lazarus, S. Coquillart and P. Jancene in 1992 [6], the object deformation is controlled by a new deformation tool: an axis. The shape of the deformation is linked to the axis deformation.

The only deformations that can be obtained are specified by bending or stretching the axis. In addition, scaling and twisting operations are possible by associating to each point of the axis a scale and a twist factor before and after the deformation.

2.1.2 Directly Specified Deformation Techniques

The deformation of the object is simply specified by the displacement of arbitrary selected points called constraints. The size and the boundary of a bounding box centered around each constraint point allows control of the extent of the deformation. Depending on this extent, the whole object can be included (global deformation) or only a limited area around the constraint point (local deformation).

The ***Space Deformation***, was developed by P. Borrel and D. Bechmann in 1990 [7]. Using space deformation, the displacement of any point in space is computed so as to satisfy certain constraints. A constraint is simply the displacement vector of an arbitrary selected point called constraint point. In particular, a constraint point can be an object point. Then, it is trivial to achieve exact placement of object points. The system can satisfy as many constraints as the user specifies unless two conflicting constraints are applied to a unique point in space. In that case, the system computes the best approximation to the solution.

The ***Direct Free-Form Deformation (DFFD)*** technique, developed by W. H. Hsu, J. F. Hughes and H. Kaufman in 1992 [8] embeds the object to be deformed inside a trivariate lattice defined by an array of control points. The object deformation follows the lattice deformation but the displacements of the lattice control points are computed from actions such that: "move this point of the object to there."

Using DFFD it is trivial to achieve exact placement of object points. The computation of the deformed lattice satisfying the displacement of object points is transparent to the user. However, the displacement of a given object point affects the surrounding points.

2.1.3 Comparison on Different Deformation Technique.

The following table shows a summary on the features of different deformation techniques.

	Best form of deformation	*Displacement Accuracy*	*Shape of lattice*
FFD	Global	Low	Parallelepiped
EFFD	Global and local	Low	Prismatic or any shape
RFFD	Global	Low	Parallelepiped
AXIAL	Global	Low	Curve
DFFD	Global and local	High	Parallelepiped

2.2 Application of Deformation

The techniques of deformation have been used in many different areas. In this section, some applications of different deformation techniques on models or surfaces are described.

2.2.1 Deforming Superquadrics

Eric Bardinet, Laurent D. Cohen and Nicholas Ayache [9] described a two-step method to fit a parametric deformable surface to 3-D points. Their approach for shape reconstruction applied to 3-D medical data is based on a two steps approach. In the first step, the best fit with a superquadric model is evaluated. This is followed by a second step for refining the details of the model by using *free-form deformation (FFD)*. The idea is to put the superquadric model in a rubber-like box and deform that box by moving its control points.

The interesting aspect of this approach is that it gives a description of a complex shape with only a small number of parameters.

2.2.2 Volume Warping

Thomas J.Ture and John F. Hughes [10] presented a technique for deforming sampled volumetric data using B-splines that is related to image warping and to the free-form deformation techniques.

The volume warping described is feasible only for relatively small data sets, in order to attain interactive speeds. They presented only a method for implementing volume warping, and not a user interface for it. Their technique

has an advantage over direct deformations of polygonal models, in which boundaries between large polygons tend to crease. Since their isosurfaces comprise polygons of approximately constant size, creases do not tend to appear.

Volume warping acts on the space in which a model lies rather than the parameters used to define the model. It is thus not appropriate for spline patch models, for instance. In contrast, volume warping is a powerful tool for deforming volumetric data.

2.2.3 Deforming Linear Object

Hidefumi Wakamatsu, Shinichi Hirai, and Kazuaki Iwata [11] described a systematic approach to the modeling of deformable fine linear objects. They develop an analytical method to model the shape of a deformable linear object such as cords and tubes.

The geometric representation of large deformation of a linear object in 3-dimensional space is established. Then the potential energy of a linear object and the geometric constraints imposed on it are formulated so as to obtain the stable shape of the object based on the formulation.

2.2.4 FFD for Animation Synthesis

Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos [12] extended the use of free-form deformation to a dynamic setting. They implemented a system that is capable of transforming a wide range of objects into dynamic characters. Their goal is to enable normally inanimate graphics

objects, such as teapots and tables, to become animated, and learn to move about in a charming, cartoon-like manner.

Their formulation is based on parameterized hierarchical FFDs augmented with lagrangian dynamics, and provides an efficient way to animate and control the simulated characters. Objects are assigned mass distributions and elastic deformation properties, which allow them to translate, rotate, and deform according to internal and external forces. In addition, they implemented an automated optimization process that searches for suitable control strategies.

2.2.5 Using FFD on feature-based surface

James C Cavendish [13] presented the results that combine the feature-based and free-form deformation design techniques into one 2-stage CAD design approach. They illustrated how the integration produces significant value in practical industrial surface design by combining the advantages offered by each method.

2.2.6 NURBS-BASED Free-Form Deformation (NFFD)

Henry J. Lamousin[18] described a technique that logically extends the FFD by basing them on nonuniform rational B-Splines. He used an NFFD to model the deformation of a human leg through a limited range of motion. NFFDs combine easily with global and local deformations, they provide an intuitive technique for deforming solids and surfaces, and they easily produce properties inherent in the deformation of physical materials, such as tapering and

necking. The combination of rational bases and a nonuniform mesh provides addional functionality for controlling the deformation of an embedded objects

All the above deformation techniques demonstrated the power and usage of free-form deformation in various research and applications. However, most of them only focus on the deformation technique while the representation of the deformed model is not considered. This research aims at maintaining a consistent solid representation scheme, CSR, before and after a deformation. Further operations such as Boolean operations between the deformed solid models is thus possible.

2.3 Algebraic Patch Technique

The survey of known techniques for constructing free-form surfaces by using 3-sided patches and 2-sided blend-patches are well described in [1]. The following survey focuses on quadric patches.

2.3.1 Dahmen's Scheme

In Dahmen's scheme [15], each facet of an input polyhedron, P, is split into six sub-facets, resulting in a 6-path macro-path. The resulting two-sided hole problem is then solved using four blend-patches. This scheme results in C^1 continuous patches, but cannot handle arbitrary P (because it requires the existence of a "transversal system" for P), and most importantly, the scheme fails to produce surfaces of pleasing shapes due to oscillations within each macro-patch resulting from the split.

2.3.2 Lodha and Warren's technique

Lodha and Warren's technique [16] differs from all other technique, in that it does not interpolate the vertices of P, but instead works with a different Bezier control net. Their approach produces quadric surfaces of the explicit form $z = F(x,y)$, for some quadratic polynomial F. This restricts patches to those whose extended surfaces a_b always interpolate the apex or "focal vertex" of the construction tetrahedra, but provides dual parametric/implicit representation for these patches. To build an extended mesh of C^1 continuous patches, 6-patch macro-patches are used together with four blend-patches, similar to Dahmen's approach. Due to the restricted nature of the patches, the kinds of surfaces that

can be modeled are limited mainly to "star-shaped objects, in which each point in the object is visible from a focal vertex common to all construction tetrahedra.

2.3.3 Guo's method

Guo's method [2] proposed to use quadrics for constructing free-form surfaces from an arbitrary polyhedron P. The technique assigns a single patch to each triangular facet of P without any patch-splitting. The resulting two sided holes are filled with blend-patches. Patch-splitting is avoided at the expense of continuity, i.e. G^1 continuous patches are generated. This method interpolates all the vertices of P.

Among all the techniques, Guo's method is promising for constructing free-form surface from an arbitrary input polyhedron P. Since FFD is the basic technique for various free-form deformation techniques, it is adopted in this research to deform algebraic patches constructed by Guo's method.

CHAPTER 3

BACKGROUND THEORIES

3.1 Algebraic Patches

In the construction of quadric patches used in CSR, Guo's approach [2] is applied. The main target of his approach is to fit smooth piecewise algebraic surfaces on an arbitrary polyhedron with normals prescribed at its vertices, so that the surfaces smoothly contain the vertices of the polyhedron.

In this section, representation of a single patch is reviewed first. Then, general steps in constructing free-form object are given.

3.1.1 Bernstein-Bezier representation of a single patch

The patch used is defined as a low degree implicit polynomial (degree two is assumed in this research) that is "clipped" by the walls of a tetrahedron. Control points for the patch are prescribed in the boundary of the tetrahedron, typically by the vertices of tetrahedron and additional points on the tetrahedral edges. Weights are associated with control points. The shape of the patch can be modified by changing the weights.

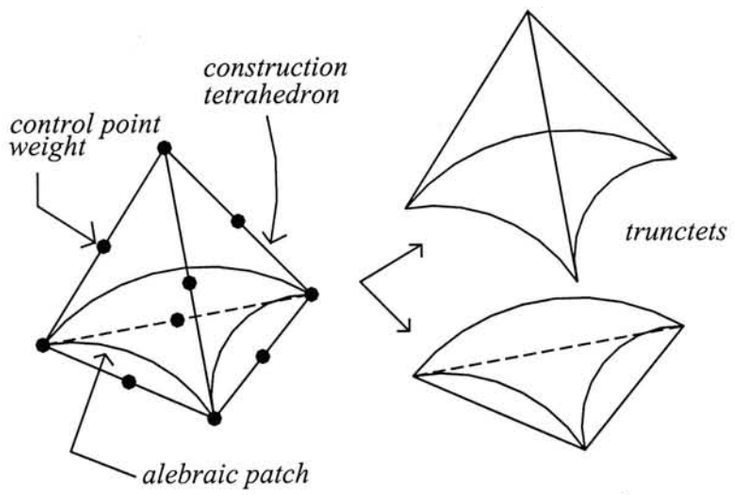


Figure 3.1 *An algebraic patch and associated trunctet solids.*

Consider a tetrahedron with vertices V_{2000}, V_{0200}, V_{0020} and V_{0002}, where the V's are non-coplanar points in E^3. Let *(s, t, u, v)* denote the local barycentric coordinates in the tetrahedron. A point q is expressed as

$$q = sV_{2000} + tV_{0200} + uV_{0020} + vV_{0002}\ ;\ s + t + u + v = 1. \qquad (3.1)$$

Let $a_b(s,t,u,v)$denote a polynomial scalar function. A contour surface of the function comprises all points for which $a_b(s,t,u,v)$ is constant. The algebraic patch p is defined as the zero contour of the function that is clipped by the tetrahedron, i.e.,

$$p = \{q \mid q \in a_b(s,t,u,v) = 0\ ;\ s,t,u,v \geq 0\}, \qquad (3.2)$$

where *(s, t, u, v)* are the barycentric coordinates of q.

The *Bernstein-Bezier* basis polynomial is used to define the polynomial in a convenient basis that provides a reasonable handle on the behavior of the zero

contour. Specifically, a degree N algebraic patch can be defined by first imposing a lattice of $(N+1)(N+2)(N+3)/6$ control points C_{ijk}, such that

$$C_{ijk} = \frac{i}{N}V_{N000} + \frac{j}{N}V_{0N00} + \frac{k}{N}V_{00N0} + \frac{l}{N}V_{000N};$$

$$i,j,k,l \geq 0;\ 1+j+k+l = N \tag{3.3}$$

For quadric patch, $N=2$, the lattice of control points is shown in *Figure 3.2* below. This lattice defines the control net for the patch, and its convex hull is the tetrahedron itself.

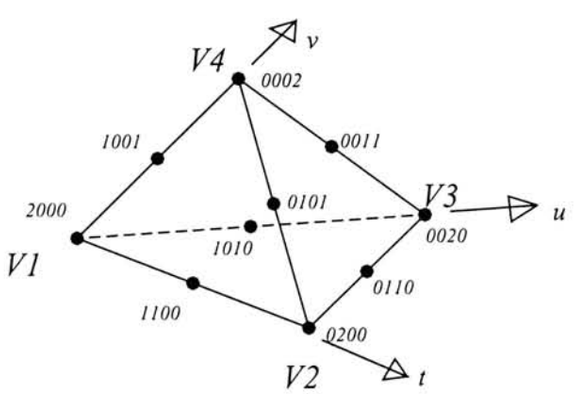


Figure 3.2 *Lattice of control points for quadric patch.*

A weight w_{ijkl} is assigned to each control points, and the algebraic patch is defined using Bernstein-Bezier basis functions as

$$a_b(s,t,u,v) = \sum_{i,j,k,l \geq 0} w_{ijkl} \frac{N!}{i!\,j!\,k!\,l!} s^i t^j u^k v^l;$$

$$i + j + k + l = N;\ s,\ t,\ u,\ v = 1-s-t-u \geq 0. \tag{3.4}$$

For quadric patch,

$$a_b(s,t,u,v) = w_{2000}s^2 + w_{0002}t^2 + w_{0020}u^2 + w_{0002}v^2 + 2w_{1100}st + 2w_{1001}sv + 2w_{0011}uv + 2w_{0110}tu + 2w_{1010}su + 2w_{0101}tv \quad (3.5)$$

To construct a three-sided patches of a trunctet, the problem becomes: Given three noncollinear points, x_i ($i = 1, 2, 3$), and their corresponding normals n_i, find a quadric surface that smoothly contains the given points.

In general, a quadric surface $f(x)=0$, where x is a point on the surface, has nine degrees of freedom. A quadric surface $f(x)=0$ is a solution of the above problem if and only if $f(x)$ satisfies the constraints

$$f(x_i) = 0 \quad (i = 1, 2, 3) \tag{3.6}$$

and

$$\nabla f(x_i) \parallel n_i \quad (i = 1, 2, 3) \tag{3.7}$$

Nine linear constraints result from *(3.6)* and *(3.7)*. For each *i*, one of the following cases is selected for the constraint *(3.7)*

1. If $n_{ix} \neq 0$, use $n_{ix}f_y(x_i) - n_{iy}f_x(x_i) = 0$ and $n_{ix}f_z(x_i) - n_{iz}f_x(x_I) = 0$. *(3.8a)*

2. If $n_{iy} \neq 0$, use $n_{iy}f_x(x_i) - n_{ix}f_y(x_i) = 0$ and $n_{iy}f_z(x_i) - n_{iz}f_y(x_I) = 0$. *(3.8b)*

3. If $n_{iz} \neq 0$, use $n_{iz}f_x(x_i) - n_{ix}f_z(x_i) = 0$ and $n_{iy}f_z(x_i) - n_{iz}f_y(x_I) = 0$. *(3.8c)*

Here n_{ix} is the x-component of n_i, $f_x = \partial f / \partial x$, etc.

Although there are nine constraints for nine unknowns in $f(x)$, solution not always exist. The solution exists if and only if the following conditions are satisfied (proof can be found in [2]).

There are i and j such that

$$(x_i - x_j) \bullet n_j = (x_j - x_i) \bullet n_i = 0 \ , \ (i \neq j) \tag{3.9}$$

or

$$\frac{\dfrac{(x_2 - x_1) \bullet n_1}{(x_1 - x_2) \bullet n_2}}{\dfrac{(x_3 - x_1) \bullet n_1}{(x_1 - x_3) \bullet n_3}} = \frac{(x_2 - x_3) \bullet n_3}{(x_3 - x_2) \bullet n_2} \tag{3.17}$$

In order to construct the required quadric patch, an apex vertex, x_4, that is not coplanar with the triangle $[\, x_1\, x_2\, x_3\,]$, is specified. Thus, a construction tetrahedron $[\, x_1 x_2 x_3 x_4\,]$ is formed.

The required quadric surface is represented in Bernstein-Bezier form as follows:

$$f(s,t,u,v) = w_{2000}s^2 + w_{0002}t^2 + w_{0020}u^2 + w_{0002}v^2 + 2w_{1100}st + 2w_{1001}sv + 2w_{0011}uv + 2w_{0110}tu + 2w_{1010}su + 2w_{0101}tv = 0 \qquad (3.11)$$

where (s, t, u, v) is the local barycentric coordinate in the tetrahedron $[x_1 x_2 x_3 x_4]$.

The surface always passes through vertices x_1, x_2 *and* x_3 (with corresponding barycentric coordinate *(1, 0, 0, 0)*, *(0, 1, 0, 0)* and *(0, 0, 1, 0)*). Substitute those barycentric coordinates into $f(s,t,u,v) = 0$ and get $w_{2000} = w_{0200} = w_{0020} = 0$.

To decide the remaining control points so that *S(f)* smoothly contains the vertices of the triangle $[\, x_1 x_2 x_3\,]$, the conditions *(3.9)* and *(3.10)* is used. When *(3.9)* is satisfied, for example, at $i = 2$ and $j = 3$, then the following can be assumed

$$\nabla f(x_1) = n_1,$$

$$\nabla f(x_2) = \frac{(x_2 - x_1)\bullet n_1}{(x_1 - x_2)\bullet n_2} n_2 \ when\ (x_1 - x_2)\bullet n_2 \neq 0,\ otherwise\ \nabla f(x_2) = n_2 \qquad (3.12)$$

$$\nabla f(x_3) = \frac{(x_3 - x_1)\bullet n_1}{(x_1 - x_3)\bullet n_3} n_3 \ when\ (x_1 - x_3)\bullet n_3 \neq 0,\ otherwise\ \nabla f(x_3) = n_3$$

Otherwise, the condition *(3.10)* must be satisfied, and the following can be assumed

$$\nabla f(x_1) = n_1,$$

$$\nabla f(x_2) = \frac{(x_2 - x_1) \bullet n_1}{(x_1 - x_2) \bullet n_2} n_2 \qquad (3.13)$$

$$\nabla f(x_3) = \frac{(x_3 - x_1) \bullet n_1}{(x_1 - x_3) \bullet n_3} n_3$$

The gradients of the polynomial f defined at x_i $(i = 1, 2, 3)$ give rise to the control points of f.

$$\begin{aligned}
&w_{2000} = w_{0200} = w_{0020} = 0 \\
&w_{0002} = 1 \quad (\textit{free parameter}) \\
&w_{1100} = (x_2 - x_1) \bullet \nabla f(x_1)/2 \\
&w_{1010} = (x_1 - x_3) \bullet \nabla f(x_3)/2 \\
&w_{0110} = (x_3 - x_2) \bullet \nabla f(x_2)/2 \\
&w_{1001} = (x_4 - x_1) \bullet \nabla f(x_1)/2 \\
&w_{0101} = (x_4 - x_2) \bullet \nabla f(x_2)/2 \\
&w_{0011} = (x_4 - x_3) \bullet \nabla f(x_3)/2
\end{aligned} \qquad (3.14)$$

w_{0002} is a free-parameter and this means there is a family of quadric surfaces satisfying the given constraints.

3.1.2 Constructing free-form objects

This section presents an algorithm that construct free-form objects. By applying the *quadric patch* and *blend-patch* concepts [2], a free-form object can be constructed with the following steps:

Step1: Specify a triangular faceted polyhedron core P whose vertices and associated surface normals $\{n_1, \ldots, n_k\}$ are to be interpolated by the surface.

Step2: Provide a bounding volume for each quadric patch to be used. i.e., construct tetrahedrons for each triangular facet of P.

Step3: (Interpolation Step) Replaces each facet of P by trunctet(s) that smoothly contain the vertices of the facet.

Step4: (Smoothing Step) Create smooth transitions between the adjacent trunctets produced in the interpolation step by filling the two-sided holes with two-sided trunctets.

Step5: Obtain the required solid model by the union and subtraction trunctets to and from polyhedron P respectively.

Details will be given in the following subsections.

3.1.2.1 Bounding volumes for quadric patches

A bounding volume is composed of tetrahedrons. The union of bounding volumes is a neighborhood N_b of the polyhedron core P. The free-form surface to be constructed lies in N_b.

The bounding volume is constructed as follows. For each facet $F_i = [\, x_1\ x_2\ x_3\,]$ of the polyhedron P, two points x_4 and y_4 of each side of the plane containing F are chosen, and they determine two tetrahedrons, $[\, x_1\ x_2\ x_3\ x_4\,]$ *(Upper tetrahedron,* T^U_i*)* and $[\, x_1 x_2 x_3 y_4\,]$ *(Lower tetrahedron,* T^L_i*)*. These two tetrahedrons form the double tetrahedron of the facet F, denoted as T^D_i. A three-sided trunctet can be constructed within the outer or inner tetrahedrons. Consider an adjacent facet $F_j = [\, x_1' x_2 x_3\,]$ and its double tetrahedron as T^D_i. Between the double tetrahedrons of facets F_i and F_j, there are two gaps. One gap lies between the tetrahedrons $[\, x_1' x_2 x_3 x_4'\,]$ and $[\, x_1 x_2 x_3 x_4\,]$); the other lies between $[\, x_1 x_2 x_3\ y_4\,]$ and $[\, x_1' x_2 x_3 y_4'\,]$. The first gap is filled with a pair of tetrahedrons $[\, x_1'' x_2\ x_3\ x_4\,]$ and $[\, x_1''\ x_2\ x_3\ x_4'\,]$, and the second gap is filled with another pair of tetrahedrons $[\, y_1''\ x_2\ x_3\ y_4\,]$ and $[\, y_1''\ x_2\ x_3\ y_4'\,]$. Two-sided trunctets are constructed within these tetrahedrons. Here x_1'' and y_1'' are points on the line segments $[x_4 x_4']$ and $[y_4 y_4']$ respectively. All these are shown in the following figure.

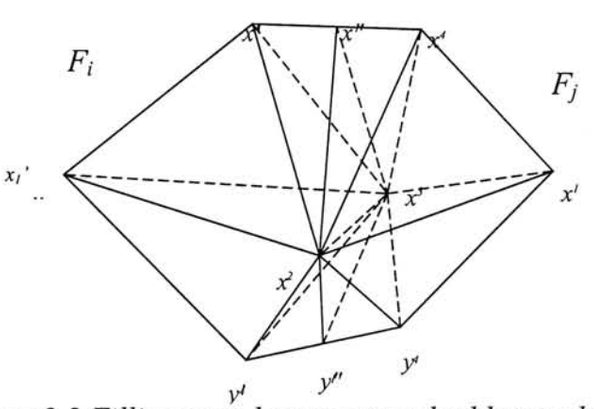


Figure 3.3 *Filling gaps between two double tetrahedrons.*

After constructing these bounding volumes, patches as well as trunctets can be formed based on the given normals and tetrahedrons (using the technique described in Section 3.1.1).

3.1.2.2 Filling two-sided gaps

Two-sided trunctets provide smooth transitions between three-sided trunctets.

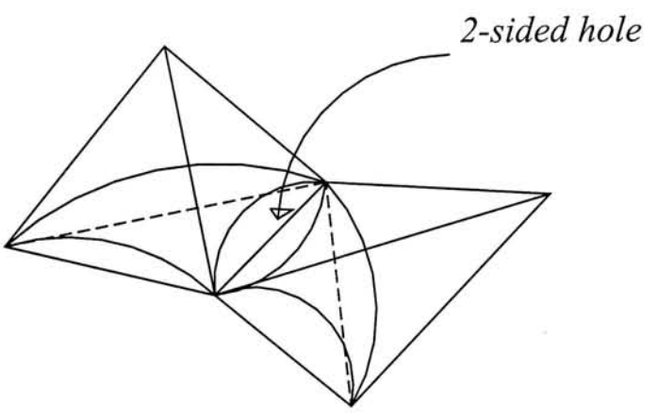


Figure 3.4 *A 2-sided hole between trunctets.*

Here smooth means tangent-plane continuity. This smoothing step is best illustrated using the two adjacent facets F_i and F_j as discussed in the above section. After the interpolating step, two quadratic polynomials f_i and f_j are defined over the tetrahedron of F_i and F_j respectively. The smoothing step produces quadratic polynomials over the tetrahedrons used to fill the gaps between the two tetrahedrons so that the quadric patches defined by these polynomials and their bounding volumes provide a smooth transition between the quadric patches defined by f_i and f_j and their bounding volumes.

Consider the problem of constructing the quadratic polynomials g_i and g_j in the tetrahedrons $W_i = [x_1'' x_2 x_3 x_4]$ and $W_j = [x_1'' x_2 x_3 x_4']$ which fill the gap between the tetrahedrons $V_i = [x_1 x_2 x_3 x_4]$ and $V_j = [x_1' x_2 x_3 x_4']$.

Suppose the two quadric surfaces f_i and f_j are tangent at the points x_2 and x_3, and t is the unit vector along $[x_2x_3]$. If $(t \bullet \nabla f_j(x_3)) \neq 0$, then there is a constant c such that

$$\nabla f_i(x_2) = c\, \nabla f_j(x_2) \qquad (3.15)$$

$$\nabla f_i(x_3) = c\, \nabla f_j(x_3) \qquad (3.16)$$

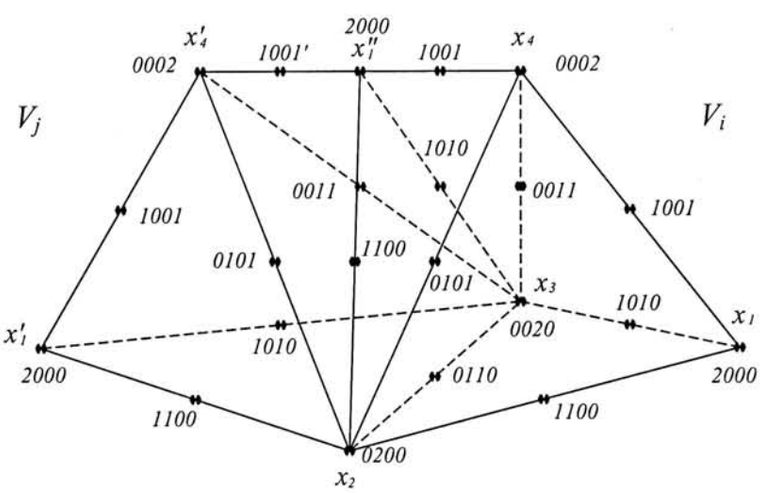


Figure 3.5 *The smoothing step.*

It is assumed that $\nabla f_i(x_2) = \nabla f_j(x_2) = n_2$ and $\nabla f_i(x_3) = \nabla f_j(x_3) = n_3$, (i.e. $c=1$) so that the gradients of g_i and g_j at the points x_2 and x_3 can be defined.

The next step is to construct g_i and g_j with G^1 conditions across the faces $[\, x_2\, x_3\, x_4\,]$, $[\, x_2\, x_3\, x_1''\,]$, and $[\, x_2\, x_3\, x_4'\,]$. Suppose f_i and f_j are expressed in Bernstein-Bezier representation as follows:

$$\begin{aligned} f_i(s,t,u,v) =& (b_{2000})_i s^2 + (b_{0002})_i t^2 + (b_{0020})_i u^2 + (b_{0002})_i v^2 \\ &+ 2(b_{1100})_i st + 2(b_{1001})_i sv + 2(b_{0011})_i uv \\ &+ 2(b_{0110})_i tu + 2(b_{1010})_i su + 2(b_{0101})_i tv = 0 \end{aligned} \qquad (3.17a)$$

and

$$f_j(s,t,u,v) = (b_{2000})_j s^2 + (b_{0002})_j t^2 + (b_{0020})_j u^2 + (b_{0002})_j v^2 + 2(b_{1100})_j st + 2(b_{1001})_j sv + 2(b_{0011})_j uv + 2(b_{0110})_j tu + 2(b_{1010})_j su + 2(b_{0101})_j tv = 0 \qquad (3.17b)$$

Similarly, g_i and g_j is constructed in Bernstein-Bezier representation as well.

$$g_i(s,t,u,v) = (a_{2000})_i s^2 + (a_{0002})_i t^2 + (a_{0020})_i u^2 + (a_{0002})_i v^2 + 2(a_{1100})_i st + 2(a_{1001})_i sv + 2(a_{0011})_i uv + 2(a_{0110})_i tu + 2(a_{1010})_i su + 2(a_{0101})_i tv = 0 \qquad (3.18a)$$

and

$$g_j(s,t,u,v) = (a_{2000})_j s^2 + (a_{0002})_j t^2 + (a_{0020})_j u^2 + (a_{0002})_j v^2 + 2(a_{1100})_j st + 2(a_{1001})_j sv + 2(a_{0011})_j uv + 2(a_{0110})_j tu + 2(a_{1010})_j su + 2(a_{0101})_j tv = 0 \qquad (3.18b)$$

The control points of g_i and g_j are defined as follows. The gradients at x_2 and x_3 determine the following control points.

$$(a_{1100})_i = \frac{1}{2}(n_2 \bullet (x_1'' - x_2)) \qquad (3.19a)$$

and

$$(a_{1010})_i = \frac{1}{2}(n_3 \bullet (x_1'' - x_3)) \qquad (3.19b)$$

If the barycentric coordinates of x_1'' with respect to V_i and V_j are $(\gamma_1, \gamma_2, \gamma_3, \gamma_4)$ and $(\gamma_1', \gamma_2', \gamma_3', \gamma_4')$ respectively, then

$$(a_{1001})_i = \gamma_1(b_{1001})_i + \gamma_2(b_{0101})_i + \gamma_3(b_{1001})_i + \gamma_4(b_{0002})_i \qquad (3.20a)$$

and

$$(a_{1001})_j = \gamma_1'(b_{1001})_j + \gamma_2'(b_{0101})_j + \gamma_3'(b_{1001})_j + \gamma_4'(b_{0002})_j \qquad (3.20b)$$

Finally,

$$(a_{2000})_i = (a_{2000})_j = \delta(a_{1001})_i + \delta'(a_{1001})_j \qquad (3.26c)$$

where δ and δ' satisfy

$$x_1'' = \delta x_4 + \delta x_4'$$

Similarly, one can construct the quadratic polynomials over the tetrahedrons which fill the gap between the tetrahedrons $[x_1 x_2 x_3 y_4]$ and $[x'_1 x_2 x_3 y'_4]$ *(Figure 3.4)*. The quadric patches defined by the quadratic polynomials constructed in the smoothing step provide smooth transitions between the adjacent patches produced by the interpolating step.

3.2 Constructive Shell Representation

Among all the solid representation schemes, the CSG and B-Rep are the most commonly used solids representation schemes.

Traditionally, free-form solids are represented in B-Rep composed of parametric patches. However, exact solutions for patch/patch intersection and curve/patch intersection cannot be obtained in general so that numerical method has to be used. On the other hand, the geometric coverage of CSG representation is limited by the primitive solids of sphere, cylinders, cones, blocks and torus so that it is not suitable for representing free-form objects.

A CSG representation with "trunctets" as primitives provides a representation scheme for free-form solids, called the Constructive Shell Representation (CSR) as introduced by J. P. Menon in 1992.

Definition of CSR: A CSR of a free-form solid is a binary tree with union operators for nodes, and trunctet-subtrees for leaves; each trunctet-subtree in turn has an intersection operator at its root, since trunctets are defined as the intersection of halfspaces.

In this research, a simplified version of CSR is used, that is more suitable for free-form deformation. A CSR solid object is made up of trunctets and a polyhedron core P, whose vertices and associated surface normals determine a set of algebraic surfaces interpolating the vertices. A trunctet is formed by

intersecting an algebraic patch with a construction tetrahedron, where two or three vertices of the tetrahedron lie on the surface of the algebraic patch.

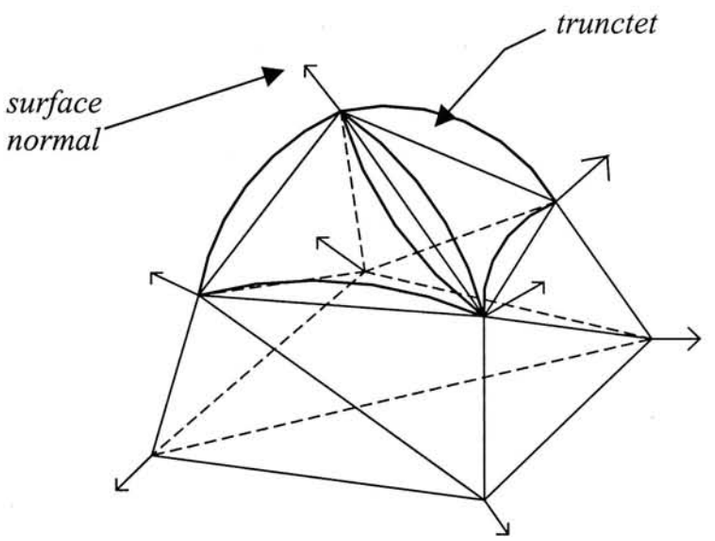


Figure 3.6 *Fitting trunctets on a polyhedron.*

By the union and subtraction of trunctets from P, a free-form solid is formed. Consider an object A, the CSR representation of A can be expressed as

$$CSR(A) = (P \cup t^P_1 \cup \ldots \cup t^P_n) - (t^D_1 \cup \ldots \cup t^D_m). \quad (3.21)$$

where t^P_i *and* t^D_j are the *"protrusion trunctets"* and *"depression trunctets"* respectively. A protrusion trunctet t^P_i is used to represent a convex portion of an object's surface, and is to be unioned to the polyhedron core. A depression trunctet t^D_j is used to represent a concave portion of an object's surface, and is to be subtracted from the polyhedron core.

A construction tetrahedron outside the solid is defined as an upper tetrahedron, while the one inside the solid is defined as a lower tetrahedron.

Protrusion trunctets are constructed inside upper tetrahedron and depression trunctets are constructed inside lower tetrahedron.

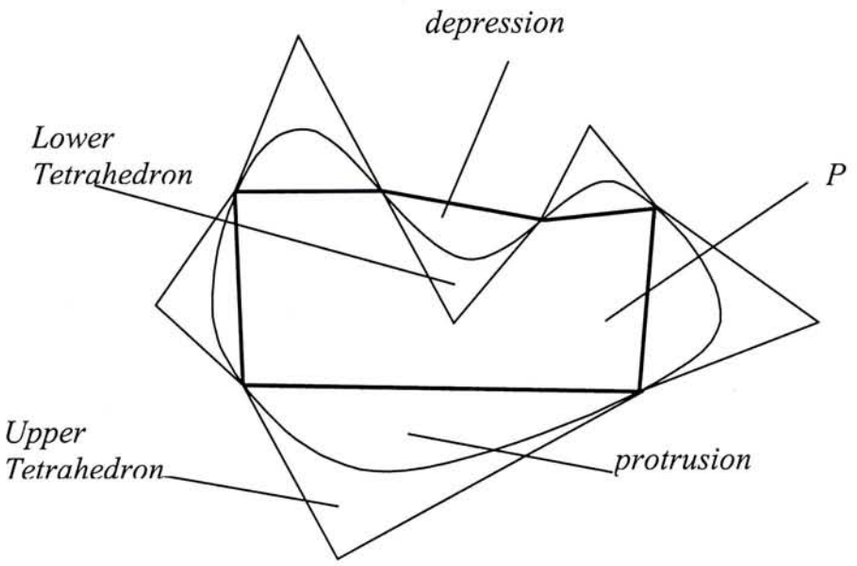


Figure 3.7 *Depression and protrusion trunctets.*

A CSR of a free-form solid is a binary tree with trunctets at the leaf nodes. Each trunctet in turn is a subtree with an intersection operator at its root, and the five intersecting halfspaces at the leaves (4 from the construction tetrahedron and 1 from the algebraic patch).

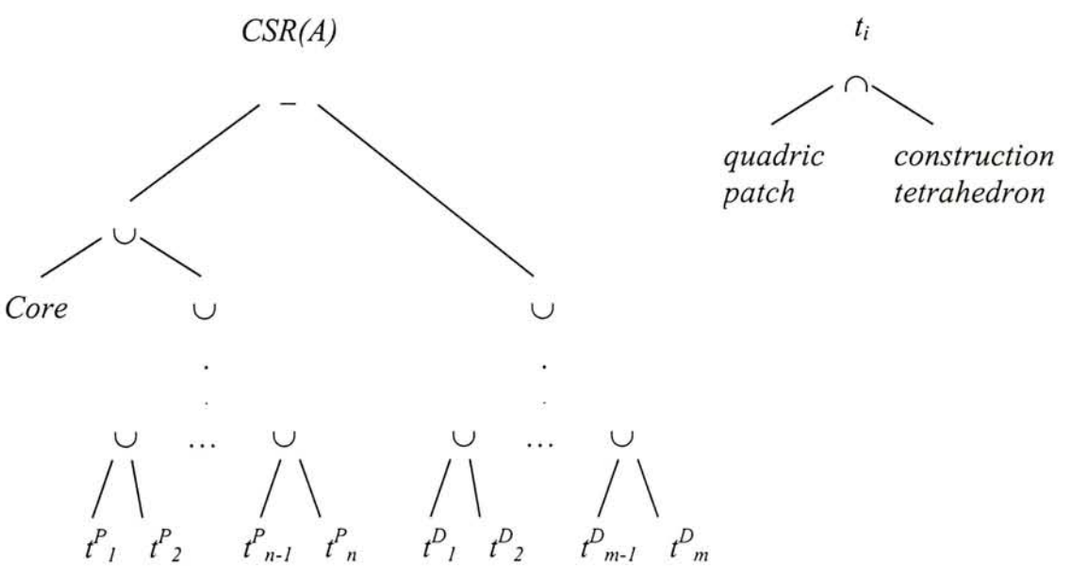


Figure 3.8 *Binary tree of a CSR object and a trunctet.*

3.2.1 Properties of quadric patches and its construction tetrahedron and trunctets

1. The boundary of a free-form solid, S, is expressed as the union of n patches, p_i of the trunctets. i.e.

$$B(S) = p_1 \cup p_2 \cup \ldots \cup p_n. \qquad (3.22)$$

where $B(S)$ denotes the boundary of S.

2. Every patch is contained in its tetrahedron.

i.e. for any point q on the patch p,

$$q = sV_1 + tV_2 + uV_3 + vV_4; \qquad (3.23)$$

$$0 \le s,t,u,v \le 1$$

$$s + t + u + v = 1.$$

where V_1, V_2, V_3, V_4 are the vertices of the construction tetrahedron of the patch p.

3. All patches are not overlapped. i.e.

$$p_i \cap p_j = \phi; \ \forall i,j \ ; \ i \neq j. \qquad (3.24)$$

4. Tetrahedrons may overlap.

5. The patch halfspace

The patch representation in barycentric coordinates $p(s,t,u,v) = 0$ *(with respect to the construction tetrahedron)* can be transformed into cartesian coordinates $f(x,y,z) = 0$ by a linear transformation L.

i.e.

$$f(x,y,z) = L(p(s,t,u,v)) \qquad (3.25)$$

An algebraic patch halfspace is hence defined as $f(x,y,z) \leq 0$.

6. Construction Tetrahedron

The construction tetrahedron, T, is a regularized intersection of 4 planar halfspaces h_i. i.e.

$$T = h_1 \cap h_2 \cap h_3 \cap h_4. \qquad (3.26)$$

where h_i's boundary contains the triangular face f_i of the tetrahedron.

7. Trunctet

A trunctet, t_i ($t^P{}_i$ *or* $t^D{}_j$) is defined as the intersection of the patch halfspace f and the construction tetrahedron T. i.e.

$$t_i = f \cap T. \qquad (3.27)$$

3.3 Free-Form Deformation

Free-form deformation (FFD) is a technique for deforming geometric models in a free-form manner. The technique can be used with any solid modeling system, such as CSG or B-rep. It involves a mapping from R^3 to R^3 through a trivariate tensor product Bernstein polynomial. The deformations can be applied either globally or locally. Local deformations can be imposed with any desired degree of derivative continuity with adjacent, undeformed regions of the model.

3.3.1 Formulating Free-Form Deformation

Mathematically, the FFD is defined in terms of a tensor product trivariate Bernstein polynomial[3]. By imposing a local coordinate system on a parallelepiped region, a point $\boldsymbol{X}$ can be expressed as

$$X = X_o + s\,S + t\,T + u\,U \qquad (3.27)$$

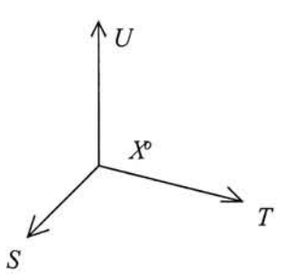


Figure 3.8 *FFD Coordinate System*

The *(s, t, u)* coordinates of X can easily be found using linear algebra. A vector solution is

$$
\begin{aligned}
s &= \frac{T \times U(X - X_o)}{T \times U \cdot S} \\
t &= \frac{S \times U(X - X_o)}{S \times U \cdot T} \\
u &= \frac{S \times T(X - X_o)}{S \times T \cdot U}
\end{aligned}
\tag{3.28}
$$

Note that for any point interior to the parallelepiped, $0 \le s \le 1$, $0 \le t \le 1$, $0 \le u \le 1$.

A grid of control points P_{ijk} can be constructed and imposed on the parallelpipe, so that

$$
P_{ijk} = X_o + \frac{i}{l} S + \frac{j}{m} T + \frac{k}{n} U \tag{3.29}
$$

where l, m, n are the dimension of the grid in the S, T, U direction respectively.

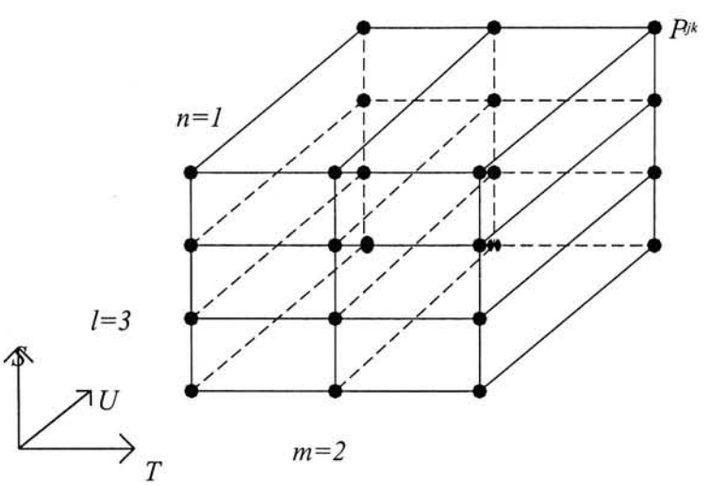


Figure 3.9 *Imposed Control Points.*

The deformation is specified by moving the control point P_{ijk}. The deformation function is defined by a trivariate tensor product Bernstein polynomial. The deformed position X_{ffd} of an arbitrary point $\boldsymbol{X}$ is found by first computing its *(s, t, u)* coordinates from equation (3.28), and then evaluating the vector valued trivariate Bernstein polynomial:

$$X_{ffd} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^i [\sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^j [\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^k P_{ijk}]] \qquad (3.30)$$

The control points P_{ijk} are actually the coefficients of the Bernstein polynomial. As in the case of Bezier curves and surface patches, there are relationships between the deformation and the control point positions.

The basic idea of applying FFD on CSR is to enclose a CSR solid model by a parallelpiped lattice, as showed in Figure 3.9. Then deform the CSR model by adjusting the position of control points on the lattice. As the positions of control points are changed, the corresponding CSR primitives, polyhedron core and trunctets, of the model are modified. Details are given in Chapter 4.

CHAPTER 4

FREE-FORM DEFORMATION

OF

CSR SOLID MODELS

4.1 Construction of a 3D Lattice

The first step in applying FFD on a CSR model is to construct a standard parallelpiped lattice enclosing the given CSR solid model. The dimension of the lattice is W*H*D where

$$W = max(x_i - x_j)$$

$$H = max(y_p - y_q)$$

$$D = max(z_m - z_n)$$

The origin of the FFD coordinate system is located at *(min(x), min(y), min(z))* where x, y and z are the x, y and z component of any point on the given model.

In addition to the size of the lattice, limitation has to be set on the minimum distance between any two adjacent control points of a lattice. If the control points are too close comparing to the size of a trunctet, the deformation effect may be insignificant or the deformed model cannot be formed normally. This constraint ensures the deformed surface of the given object can still be represented by a set of quadric patches.

Figure 4.1a shows an example where an improper lattice is used. The expected result cannot be represented by a quadric patch. On the other hand, if the size of a trunctet is small enough, the expected deformation can be attained *(Figure 4.1b)*.

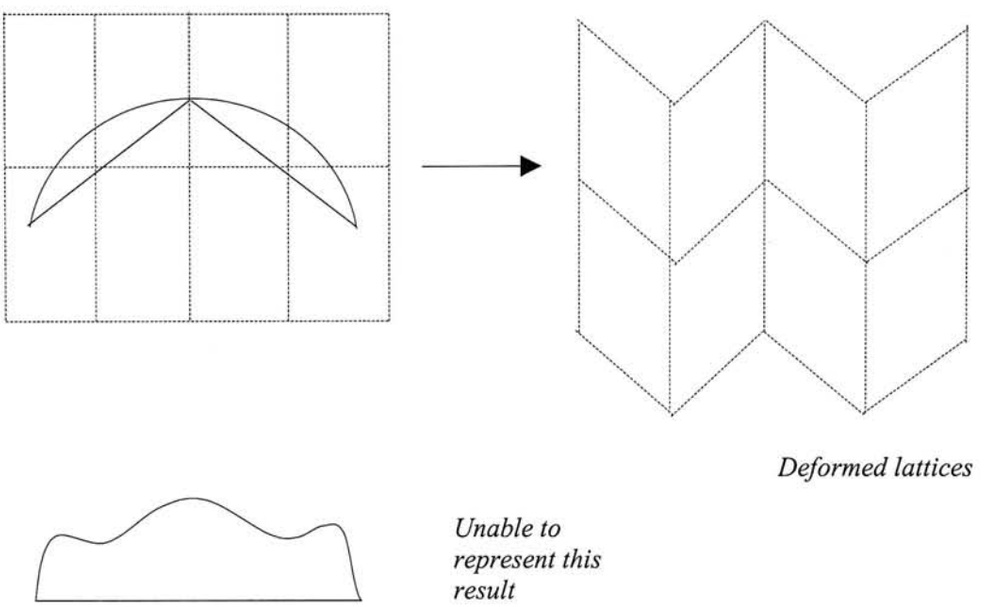


Figure 4.1a *Cross-section of a single trunctet.*

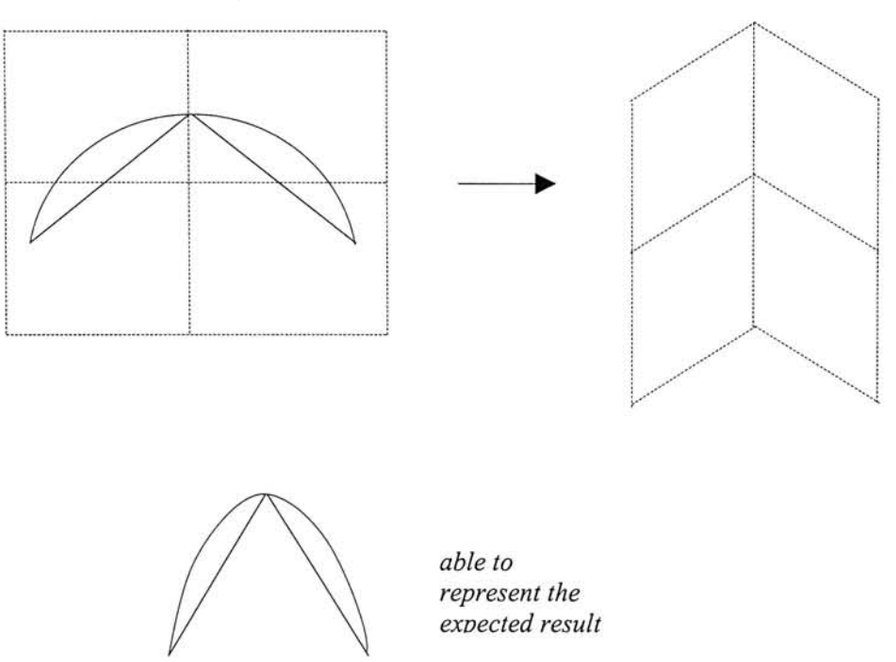


Figure 4.1b *Cross-section of a multiple trunctet.*

The constraint is set as follows:

Let x_i be a vertex of an input polyhedron P,

x_j be an adjacent vertex of x_i.

and c_m be a control point on the FFD lattice,

c_n be an adjacent control point of c_m.

Then, $Dist(x_i, x_j) \leq Dist(c_m, c_n)$, $\forall$ $i, j, m, n.$ *(4.1)*

4.2 Relation between Weights, Normals and Shape of a Trunctet

Once a CSR object is deformed, it not only changes the location of the vertices of each trunctet, but the shape of trunctets is also changed. The shape of a trunctet is closely related to the weights on its construction tetrahedron. Weights are determined by the surface normals at the base vertices of the construction tetrahedron.

The relation between weights and normal is given in (3.21) and is restated here.

$$
\begin{aligned}
&w_{2000} = w_{0200} = w_{0020} = 0 \\
&w_{0002} = 1 \quad (\textit{free parameter}) \\
&w_{1100} = (x_2 - x_1) \bullet \nabla f(x_1)/2 \\
&w_{1010} = (x_1 - x_3) \bullet \nabla f(x_3)/2 \\
&w_{0110} = (x_3 - x_2) \bullet \nabla f(x_2)/2 \\
&w_{1001} = (x_4 - x_1) \bullet \nabla f(x_1)/2 \\
&w_{0101} = (x_4 - x_2) \bullet \nabla f(x_2)/2 \\
&w_{0011} = (x_4 - x_3) \bullet \nabla f(x_3)/2
\end{aligned}
$$

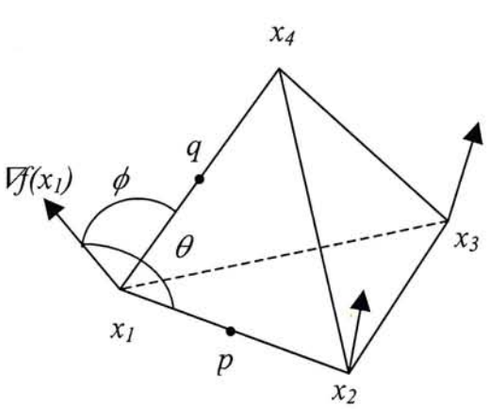


Figure 4.2

Assume that w_{1100}, w_{1001} are the weights associated with the control point p and q respectively as shown in Figure 4.2.

Let the angle between $\nabla f(x_1)$ and (x_2-x_1) be θ, and the angle between $\nabla f(x_1)$ and (x_4-x_1) be ϕ. Then,

$$w_{1100} = \frac{1}{2}|(x_2 - x_1)||\nabla f(x_1)|\cos\theta \qquad (4.2a)$$

$$w_{1001} = \frac{1}{2}|(x_4 - x_1)||\nabla f(x_1)|\cos\phi \qquad (4.2b)$$

If the control point p is located below the quadric surface interpolating x_1, x_2 and x_3, the surface is locally convex and angle θ lies between 90° and 180°. From (4.2a), w_{1100} will be negative. On the other hand, if p is located above the surface *(Figure 4.3)*, the surface is locally concave and the angle θ lies between 0° and 90°. Then, w_{1100} will be positive.

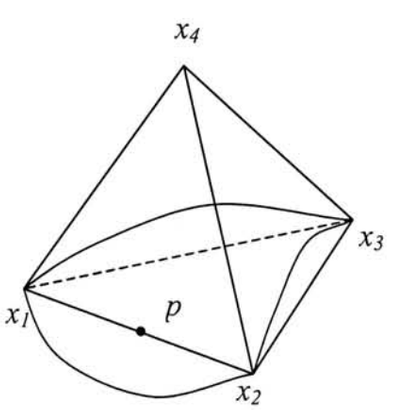


Figure 4.3 *Point p is locate above the surface.*

In some extreme cases, the surface may be deformed such that the surface patch of a trunctet is no longer bounded by its construction tetrahedron. For example, if the surface is deformed and so that the control point q lies below the surface *(Figure 4.4)*, then angle ϕ will be >90° and w_{1001} is negative.

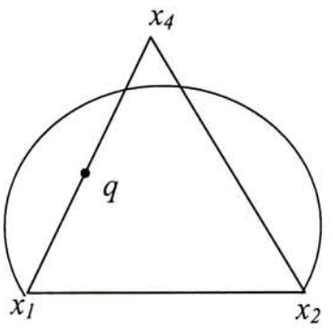


Figure 4.4 *Point q is located below the surface.*

By observing the relationship between weights and shape of surface, it can be concluded that if a control point p is located above the surface, the corresponding weight will be positive, otherwise, the weight will be negative. In this way, the shape of a deformed trunctet can be roughly predicted. This prediction is useful when applying FFD on CSR objects. Details will be discussed in sections 4.4.1.5 and 4.4.1.6.

4.3 Applying FFD on CSR solid models

In this section, two different approaches for applying the Free-Form Deformation technique on CSR solid model are discussed. In *Section 4.4*, another approach, surface fitting which is adopted in this research, will be described.

As described in Chapter 3, a CSR solid model contains trunctets and core. The unions of trunctets form a thick shell that constitutes the boundary (and the surface) of the solid. In order to analyze the deformation of a solid model, it is important to study the ways of deforming a trunctet. The parameters of a trunctet include the location of vertices, V_1, V_2, V_3 and V_4 of the construction tetrahedron, and the three surface normals, n_1, n_2 and n_3 of the patch as well as the surface patch *(Figure 4.5)*. In order to deform a trunctet in a proper way, it is crucial to determine how FFD should be applied to those parameters.

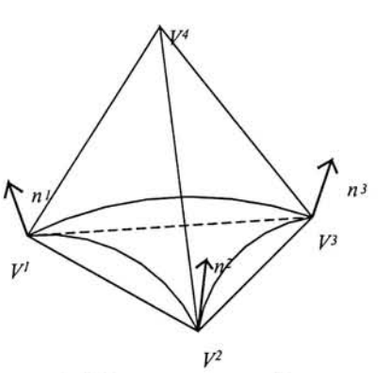


Figure 4.5 *Parameters of a trunctet.*

The Free-Form Deformation technique is a function f that takes a point X as input and gives a new point X_{ffd} as output.

$$\text{i.e., } f(X) = X_{ffd}$$

Traditionally, FFD takes every surface point of the original object as input, and deform the object by changing the position of all the surface points. When applying FFD (or in other words, the function *f*) on a CSR solid model, the kinds of input to be given to *f* have to be decided.

Deforming the polyhedron core directly and reconstruct normals (by averaging adjacent facet normals [2]) is not possible since there is no control on surface normals at polyhedron vertices, while the surface shape of a model is determined by those surface normals. The reconstructed normals cannot lead to expected result since the actual deformed surface normals may not be the same as the reconstructed one.

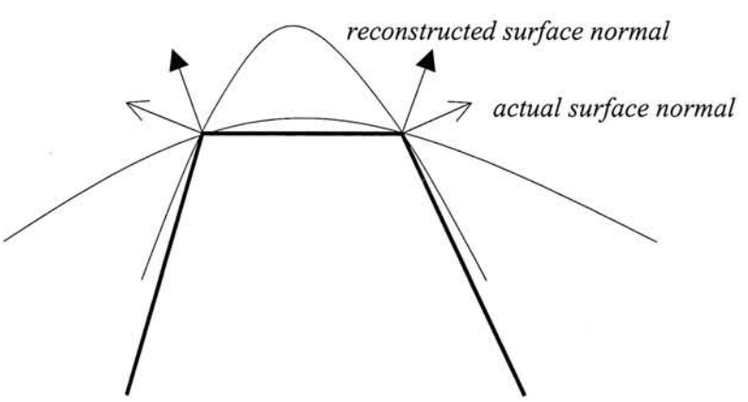


Figure 4.6 *Cross-section of part of a polyhedron core.*

The first approach focuses on deforming the surface normals at the vertices directly. The second approach focuses on deforming the normals indirectly by using surface points around the vertices. Whilst, the third method in *Section 4.4* aims at deforming the surface patches directly.

Problem statement:

Given a CSR free-form solid *A, (i.e., a triangular faceted polyhedron P with surface normals* $\vec{n}_i$ *specified at the vertices* V_i*, and each triangular facet of the polyhedron is "replaced" by a three-sided trunctets* t_i*.)*, find a way to deform it using FFD by changing the trunctet parameters as well as the location of the vertices of the triangular faceted polyhedron.

An example is shown in *Figure 4.7*, where the two-sided holes are not filled for simplicity. The FFD lattice is shown in *Figure 4.8*, and the deformed object is shown in *Figure 4.10*.

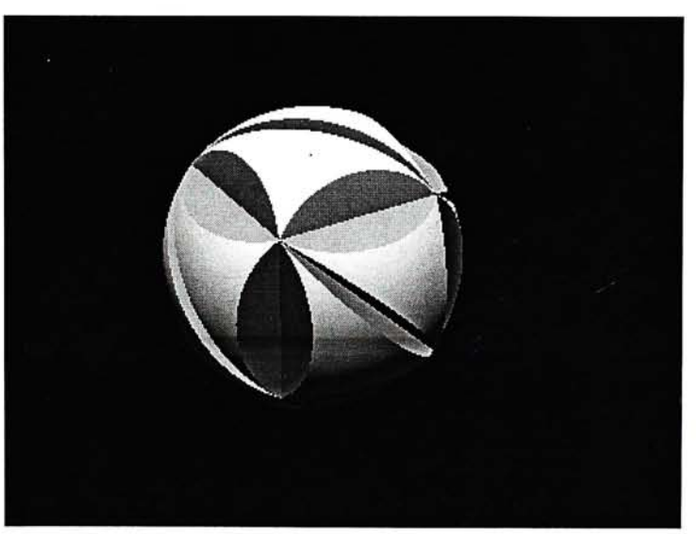


Figure 4.7 *Original solid model A.*

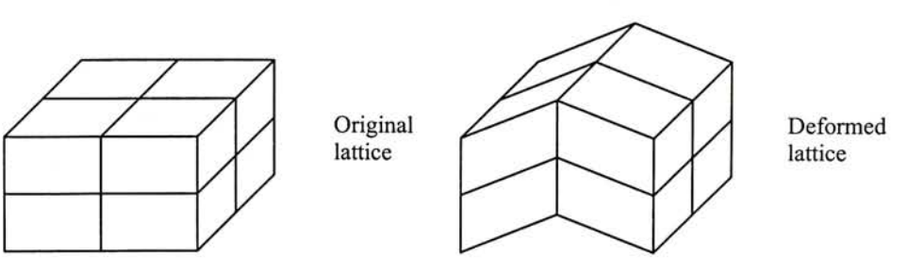


Figure 4.8 *FFD Lattice used.*

4.3.1 Deforming Normal at Vertices

When deforming a solid model, it is obvious that the direction of any surface normals on the model will be changed. Those changes depend on how the model is deformed.

Consider a solid model A, constructed with polyhedron P and the associated trunctets t_1, t_2, ..., t_m. Assume the solid A to be a simple convex model. Then

$$A = P \cup (t_1 \cup t_2 \cup \ldots \cup t_m) \qquad (4.3)$$

(*Note*: the trunctet t_i is defined by the polygon vertices on P and the associated surface normals at the vertices.)

The polyhedron P with vertices V_1, V_2, ..., V_n is deformed by changing the location of all its vertices so that the vertices of the deformed polyhedron P_{ffd} are $(V_1)_{ffd}$, $(V_2)_{ffd}$, ..., $(V_n)_{ffd}$.

The surface normal $\vec{n}_i$ at vertex V_i is treated as a unit vector starting from V_i to V_i'. A surface normal is "deformed" by applying the FFD function f on the corresponding V_i and V_i'. The deformed surface normal, $(\vec{n}_i)_{ffd}$ is thus a vector defined by $(V_i')_{ffd} - (V_i)_{ffd}$. *(Figure 4.9)*

The deformed trunctet $(t_i)_{ffd}$ is constructed by using the corresponding "deformed" vertices $(V_i)_{ffd}$ on P_{ffd} and the associated "deformed" surface normals $(\vec{n}_i)_{ffd}$.

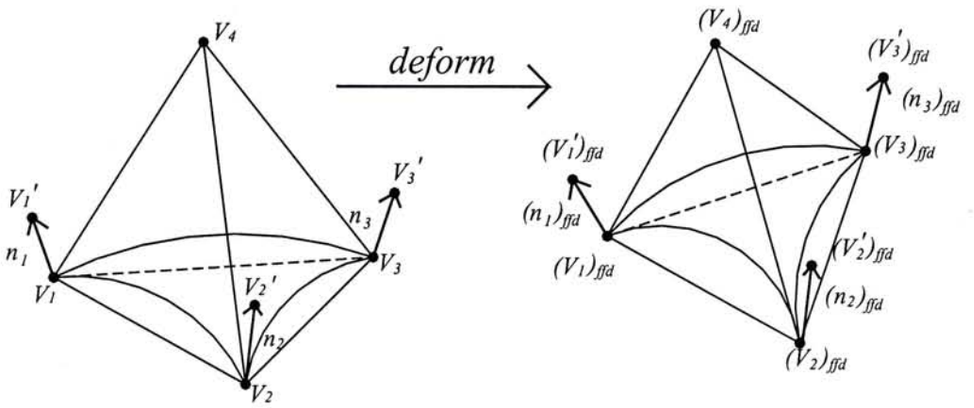


Figure 4.9 *Deformed parameters.*

Finally, the deformed solid model A_{ffd} is generated by the union of P_{ffd} and t_i s. i.e.,

$$A_{ffd} = P_{ffd} \cup ((t_1)_{ffd} \cup (t_2)_{ffd} \cup \ldots \cup (t_m)_{ffd}) \qquad (4.4)$$

Figure 4.10 shows the result of deforming the object in *Figure 4.7* by deforming normals at the vertices.

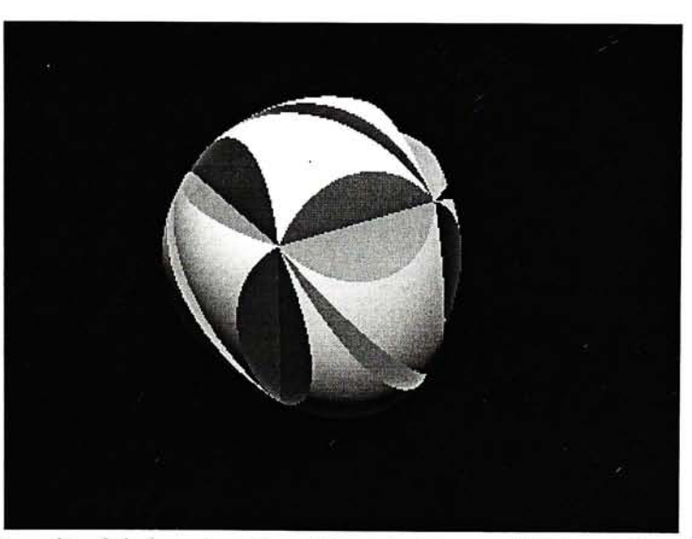


Figure 4.10 *Result of deforming the object in Figure 4.7 by deforming normals.*

4.3.2 Using Vertices' Neighborhoods

Similar to the method discussed in the previous section, an object is deformed by adjusting the surface normals. In this approach, the "deformed" surface normals are found by making use of the neighboring point of the vertices.

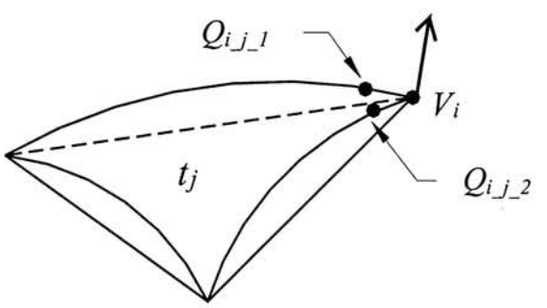


Figure 4.11 *Two adjacent points of* V_i.

Consider a trunctet t_j with vertex V_i and surface normal $\vec{n}_i$ as one of its base vertices and normals respectively. Two points, $Q_{i\_j\_1}$ and $Q_{i\_j\_2}$, adjacent to V_i are selected. If $Q_{i\_j\_1}$ and $Q_{i\_j\_2}$ were chosen such that they are very close to V_i, the cross product of vector $\overrightarrow{V_iQ_{i\_j\_1}}$ and $\overrightarrow{V_iQ_{i\_j\_2}}$ would produce a vector having same direction as $\vec{n}_i$. i.e.,

$$\overrightarrow{V_iQ_{i\_j\_1}} \times \overrightarrow{V_iQ_{i\_j\_2}} = \alpha\, \vec{n}_i \qquad (4.5)$$

where α is a scalar.

Applying the FFD function f to points V_i, $Q_{i\_j\_1}$ and $Q_{i\_j\_2}$ gives the "deformed" points $(V_i)_{ffd}$, $(Q_{i\_j\_1})_{ffd}$ and $(Q_{i\_j\_2})_{ffd}$. The direction of the "deformed" surface normal at $(V_i)_{ffd}$ is found by the cross product of $\overrightarrow{(V_i)_{ffd}(Q_{i\_j\_1})_{ffd}}$ and $\overrightarrow{(V_i)_{ffd}(Q_{i\_j\_2})_{ffd}}$.

i.e.,

$$(\vec{n}_i)_{ffd} = \beta(\overrightarrow{(V_i)_{ffd}(Q_{i\_j\_1})_{ffd}} \times \overrightarrow{(V_i)_{ffd}(Q_{i\_j\_2})_{ffd}}) \qquad (4.6)$$

where β is a scalar.

In this approach, $(\vec{n}_i)_{ffd}$ is assumed to be a unit vector.

$$\text{So, } \beta = \frac{1}{\left| \overrightarrow{(V_i)_{ffd}(Q_{i\_j\_1})_{ffd}} \times \overrightarrow{(V_i)_{ffd}(Q_{i\_j\_2})_{ffd}} \right|} \qquad (4.7)$$

Under this circumstance, deformed trunctets sharing the same vertex $(V_i)_{ffd}$ would have the same surface normal at $(V_i)_{ffd}$. Hence, the surface of the solid model will be smooth at the "vertices" with G^1 continuous. *Figure 4.12* shows the result of deforming the object in *Figure 4.7* using vertices' neighborhoods.

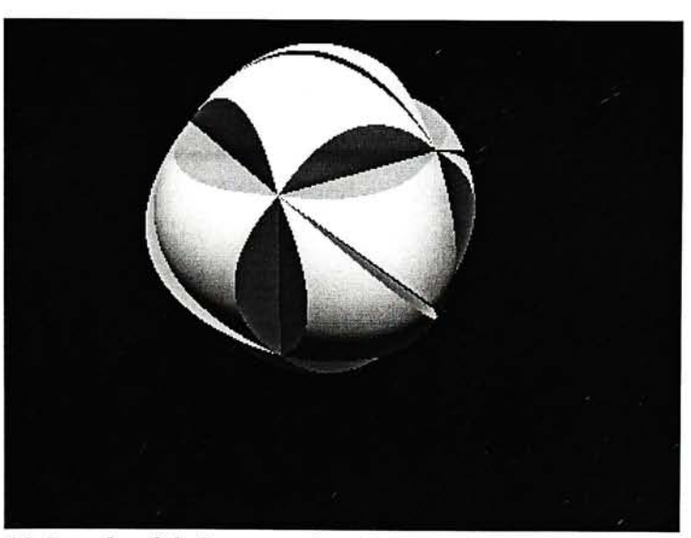


Figure 4.12 *Result of deforming the object in Figure 4.7 using vertices' neighborhood.*

The techniques described in Section 4.3.1 only concentrate on deforming surface normals but no surface points are concerned. In addition, unit surface normals are assumed, but if surface normal with other magnitude are used, different results are obtained. *Figure 4.13* shows two normals *(p_1p_2 and p_1p_3)* with different magnitudes would give different results. The deformation result does not only depend on FFD, but also the magnitude of surface normals used. However, there are no rules on choosing the magnitude of surface normals that affects the deformation result.

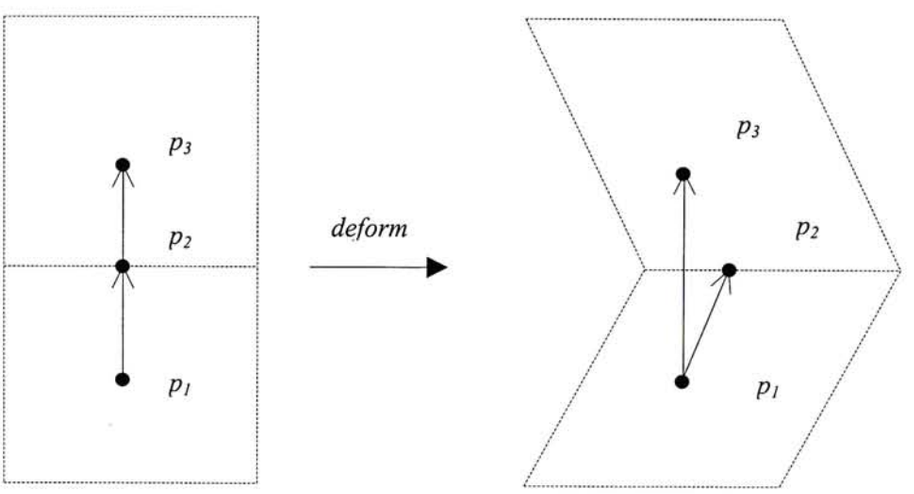


Figure 4.13 *A FFD Lattice applied on two normals.*

In order to take surface points into account when deforming surface normals, technique in Section 4.3.2 is adopted. However, the main problem of this approach is that the magnitude of the surface normals cannot be determined in a simple and proper way. In other words, there is not enough parameter to construct the expected model. This leads to the surface fitting approach to be discussed in the following section.

4.4 Free-Form Deformation of CSR Objects by Surface Fitting

In order to deform a CSR solid model, both the trunctets and the cores of the object have to be deformed. Since a trunctet is the intersection of four planes (of the construction tetrahedron) and a surface patch, deformation of the planes is determined by the positions of the tetrahedron vertices. In the surface fitting approach, the surface patches are deformed directly.

4.4.1 Deforming a Single Surface Patch

As discussed in section 3.2.1 (Equation 3.18), a quadric patch is represented by the surface equation.

$$a_b(s,t,u,v) = w_{2000}s^2+w_{0200}t^2+w_{0020}u^2+w_{0002}v^2+2w_{1100}st+2w_{1001}sv+$$
$$2w_{0011}uv+2w_{0110}tu+2w_{1010}su+2w_{0101}tv = 0, \qquad (4.8)$$

where s, t, u and v are expressed in barycentric coordinates. The variables w_{2000}, w_{0200}, w_{0020}, w_{0002}, ..., w_{0101} are the weights associated with the control points lying on the construction tetrahedron *(ref to Figure 3.5)*.

Altogether, there are 10 variables in $a_b(s,t,u,v)$. However, the surface patch always passes through the base vertices V_1 $(s=1, t=0, u=0, v=0)$, $V_2(s=0, t=1, u=0, v=0)$ and $V_3(s=0, t=0, u=1, v=0)$ of the trunctet so that the weights at these vertices are zero. i.e.,

$$w_{2000} = w_{0200} = w_{0020} = 0$$

Hence, Equation *(4.8)* is rewritten as

$$a_b(s,t,u,v) = w_{0002}v^2+2w_{1100}st+2w_{1001}sv+ 2w_{0011}uv +$$
$$2w_{0110}tu+2w_{1010}su+2w_{0101}tv = 0. \qquad (4.9)$$

Without loss of generality, w_{0002} is set to 1. The number of unknowns is thus reduced to 6. Six non-coplanar points on the surface of the deformed trunctet are thus sufficient for determining the deformed surface.

A general procedure of deforming a quadric surface patch is given below (details will be given in the following subsections):

Step 1: Locate 6 points p_1, p_2, p_3, q_1, q_2 and q_3 **on the base triangle**.

Step 2: Find 6 surface points **on the quadric surface**. A surface point P_1 is found by intersecting a line l (passing through p_1 with direction of the normal vector of the base triangle) with the quadric surface. Other surface points are found in the same way.

Step 3: Apply FFD on 6 surface points.

Step 4: Solve the unknown weights w_{1100}, w_{1010}, w_{0110}, w_{1001}, w_{0101} and w_{0011} of the new deformed quadric patch by using deformed surface points.

4.4.1.1 Locating Surface Points

In order to avoid numerically unstable result, the surface points, P_1, P_2, P_3, Q_1, Q_2 and Q_3, are chosen so that they are approximately evenly distributed. First, locate six points p_1, p_2, p_3, q_1, q_2 and q_3 *(as shown in Figure 4.14)*, on the base triangle of the construction tetrahedron. The points are defined as follows:

$$p_1 = \frac{x_1 + x_2}{2} \qquad (4.10a)$$

$$p_2 = \frac{x_2 + x_3}{2} \qquad (4.10b)$$

$$p_3 = \frac{x_3 + x_1}{2} \qquad (4.10c)$$

$$q_1 = x_1 + r(p_2 - x_1) \qquad (4.10d)$$

$$q_2 = x_2 + r(p_3 - x_2) \qquad (4.10e)$$

$$q_3 = x_3 + r(p_1 - x_3) \qquad (4.10f)$$

where r is a factor controlling the location of q_1, q_2 and q_3. Normally, set $r = 1/4$ and check if the resulted 6 surface points are coplanar or not. If they are coplanar, use $r = 1/3$ or other values instead to produce non-coplanar surface points.

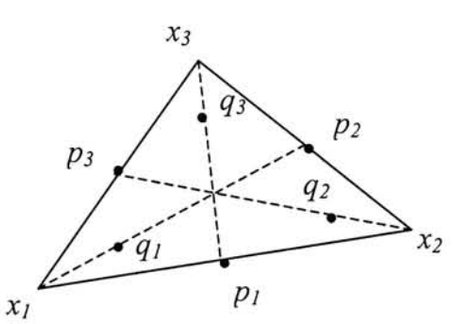


Figure 4.14** Points **on base triangle.

Let a vector n *(with direction (a,b,c))* be the normal of the plane containing x_1, x_2 and x_3. A surface point P_1 is found by intersecting the surface by a line l passing through p_1, (p_{1x}, p_{1y}, p_{1z}), with direction n, the surface point P_1 is the intersecting point of line l and the quadric surface (Line-Surface intersection).

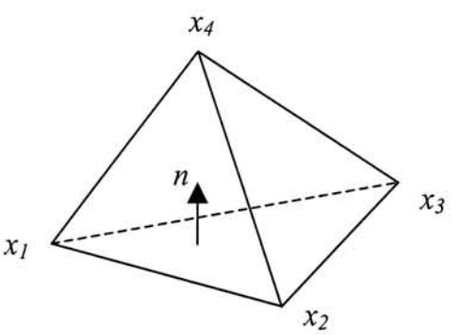


Figure 4.15 *Plane normal on base triangle.*

That is, any point *(x,y,z)* on the line l is $(p_{1x}+ar, p_{1y}+br, p_{1z}+cr)$ where r is a variable. The conversion between Barycentric and Cartesian Coordinates is a linear transformation, as discussed in the following sections.

4.4.1.2 Conversion Between Barycentric and Cartesian Coordinates

In a deformation process, the vertices of the trunctet and the surface points in Cartesian coordinates are transformed using the free-form deformation function of *Equation 3.30*. The Cartesian coordinates of all the surface points are obtained by the linear transformation,

$$P_s(x,y,z) = V_1 + t(V_2 - V_1) + u(V_3 - V_1) + v(V_4 - V_1) \qquad (4.11)$$

The weights of the deformed truncets are obtained by solving *Equation 4.12*.

$$\begin{cases} a_b(s,t,u,v) = 0 \\ s+t+u+v = 1 \end{cases} \qquad (4.12)$$

This requires converting the deformed surface points into barycentric coordinates with the following equations:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} (V_2^x - V_1^x) & (V_3^x - V_1^x) & (V_4^x - V_1^x) \\ (V_2^y - V_1^y) & (V_3^y - V_1^y) & (V_4^y - V_1^y) \\ (V_2^z - V_1^z) & (V_3^z - V_1^z) & (V_4^z - V_1^z) \end{bmatrix}^{-1} \begin{bmatrix} x - V_1^x \\ y - V_1^y \\ z - V_1^z \end{bmatrix}, \qquad (4.13)$$

$$s = 1 - t - u - v \qquad (4.14)$$

where $V_i^x\ V_i^y\ V_i^z$ are the *x, y, and z* components of V_i.

4.4.1.3 Evaluating the Deformed Surface Patch

Use *Equation 4.9* and assuming the deformed surface points are $p_1' = (s_1, t_1, u_1, v_1)$, $p_2' = (s_2, t_2, u_2, v_2)$, $p_3' = (s_3, t_3, u_3, v_3)$, $q_1' = (s_4, t_4, u_4, v_4)$, $q_2' = (s_5, t_5, u_5, v_5)$, and $q_3' = (s_6, t_6, u_6, v_6)$, the weights of the deformed surface are obtained by solving the equation

$$\begin{bmatrix} v_1^2 \\ v_2^2 \\ v_3^2 \\ v_4^2 \\ v_5^2 \\ v_6^2 \end{bmatrix} + \begin{bmatrix} 2s_1t_1 & 2s_1v_1 & 2u_1v_1 & 2t_1u_1 & 2s_1u_1 & 2t_1v_1 \\ 2s_2t_2 & 2s_2v_2 & 2u_2v_2 & 2t_2u_2 & 2s_2u_2 & 2t_2v_2 \\ 2s_3t_3 & 2s_3v_3 & 2u_3v_3 & 2t_3u_3 & 2s_3u_3 & 2t_3v_3 \\ 2s_4t_4 & 2s_4v_4 & 2u_4v_4 & 2t_4u_4 & 2s_4u_4 & 2t_4v_4 \\ 2s_5t_5 & 2s_5v_5 & 2u_5v_5 & 2t_5u_5 & 2s_5u_5 & 2t_5v_5 \\ 2s_6t_6 & 2s_6v_6 & 2u_6v_6 & 2t_6u_6 & 2s_6u_6 & 2t_6v_6 \end{bmatrix} \begin{bmatrix} w_{1100} \\ w_{1001} \\ w_{0011} \\ w_{0110} \\ w_{1010} \\ w_{0101} \end{bmatrix} = 0 \qquad (4.15a)$$

Hence,

$$\begin{bmatrix} w_{1100} \\ w_{1001} \\ w_{0011} \\ w_{0110} \\ w_{1010} \\ w_{0101} \end{bmatrix} = - \begin{bmatrix} 2s_1t_1 & 2s_1v_1 & 2u_1v_1 & 2t_1u_1 & 2s_1u_1 & 2t_1v_1 \\ 2s_2t_2 & 2s_2v_2 & 2u_2v_2 & 2t_2u_2 & 2s_2u_2 & 2t_2v_2 \\ 2s_3t_3 & 2s_3v_3 & 2u_3v_3 & 2t_3u_3 & 2s_3u_3 & 2t_3v_3 \\ 2s_4t_4 & 2s_4v_4 & 2u_4v_4 & 2t_4u_4 & 2s_4u_4 & 2t_4v_4 \\ 2s_5t_5 & 2s_5v_5 & 2u_5v_5 & 2t_5u_5 & 2s_5u_5 & 2t_5v_5 \\ 2s_6t_6 & 2s_6v_6 & 2u_6v_6 & 2t_6u_6 & 2s_6u_6 & 2t_6v_6 \end{bmatrix}^{-1} \begin{bmatrix} v_1^2 \\ v_2^2 \\ v_3^2 \\ v_4^2 \\ v_5^2 \\ v_6^2 \end{bmatrix} \qquad (4.15b)$$

Finally, the normals n_i at the vertices are determined with the following sets of equations.

$$\begin{cases} w_{1100} = \frac{1}{2}(V_2 - V_1) \bullet (n_1)_{ffd} \\ w_{1010} = \frac{1}{2}(V_3 - V_1) \bullet (n_1)_{ffd} \\ w_{1001} = \frac{1}{2}(V_4 - V_1) \bullet (n_1)_{ffd} \end{cases} \qquad (4.16a)$$

$$\begin{cases} w_{1100} = \frac{1}{2}(V_1 - V_2) \bullet (n_2)_{ffd} \\ w_{0110} = \frac{1}{2}(V_3 - V_2) \bullet (n_2)_{ffd} \\ w_{0101} = \frac{1}{2}(V_4 - V_2) \bullet (n_2)_{ffd} \end{cases} \qquad (4.16b)$$

$$\begin{cases} w_{1010} = \frac{1}{2}(V_1 - V_3) \bullet (n_3)_{ffd} \\ w_{0110} = \frac{1}{2}(V_2 - V_3) \bullet (n_3)_{ffd} \\ w_{0011} = \frac{1}{2}(V_4 - V_3) \bullet (n_3)_{ffd} \end{cases} \qquad (4.16c)$$

Hence,

$$n_1' = \begin{bmatrix} (V_2 - V_1)_x & (V_2 - V_1)_y & (V_2 - V_1)_z \\ (V_3 - V_1)_x & (V_3 - V_1)_y & (V_3 - V_1)_z \\ (V_4 - V_1)_x & (V_4 - V_1)_y & (V_4 - V_1)_z \end{bmatrix}^{-1} \begin{bmatrix} w_{1100} \\ w_{1010} \\ w_{1001} \end{bmatrix} \qquad (4.17a)$$

$$n_2' = \begin{bmatrix} (V_1 - V_2)_x & (V_1 - V_2)_y & (V_1 - V_2)_z \\ (V_3 - V_2)_x & (V_3 - V_2)_y & (V_3 - V_2)_z \\ (V_4 - V_2)_x & (V_4 - V_2)_y & (V_4 - V_2)_z \end{bmatrix}^{-1} \begin{bmatrix} w_{1100} \\ w_{0110} \\ w_{0101} \end{bmatrix} \qquad (4.17b)$$

$$n_3' = \begin{bmatrix} (V_1 - V_3)_x & (V_1 - V_3)_y & (V_1 - V_3)_z \\ (V_2 - V_3)_x & (V_2 - V_3)_y & (V_2 - V_3)_z \\ (V_4 - V_3)_x & (V_4 - V_3)_y & (V_4 - V_3)_z \end{bmatrix}^{-1} \begin{bmatrix} w_{1010} \\ w_{0110} \\ w_{0011} \end{bmatrix} \qquad (4.17c)$$

4.4.1.4 Saddle Shape Trunctet

Apart from pure concave and pure convex trunctet within a single construction tetrahedron, a saddle shape trunctet can also be formed inside a tetrahedron. It is constructed as an ordinary trunctet by using the vertices and their normals. It is independent of whether the saddle point is located at a vertex, on an edge or on the surface of a trunctet. No special treatment is needed. *Figure 4.16a* shows a single trunctet containing the saddle point while *Figure 4.16b* to *4.16d* show a wider saddle shape surface with the saddle point located at different part of a trunctet. The equation of the surface is $z = 1 + x^2 - y^2$ and the saddle point is located at (0,0).

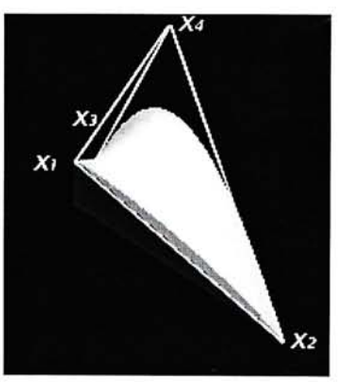


Figure 4.16a *A saddle shape trunctet.*

Figure 4.16b *Saddle point at a vertex.*

Figure 4.16c *Saddle point on an edge.*

Figure 4.16d *Saddle point on surface.*

4.4.1.5 Using Double Tetrahedrons

Normally, in the process of replacing a triangular facet of Polyhedron with a trunctet, one trunctet is used if the corresponding surface does not intersect the base triangle of the trunctet. If the surface does intersect the base triangle, a double tetrahedron has to be used. A double tetrahedron is composed of two opposite tetrahedron sharing the same base triangle.

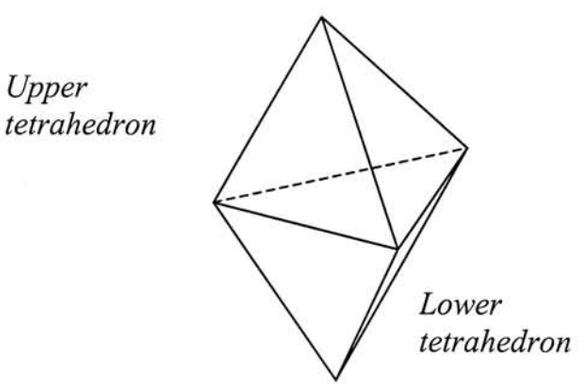


Figure 4.17 *Double Tetrahedron composes of upper and lower tetrahedron.*

The upper tetrahedron is the one on the outside of an input polyhedron while the lower tetrahedron is the one in the inside of an input polyhedron. A convex surface relative to the upper tetrahedron is a concave surface relative to the lower tetrahedron and vice versa.

The following tables explain which tetrahedron(s) should be used for building trunctets after deformation. The determination is based on the sign of the weights, w_{1100}, w_{1010} *and* w_{0110}, of the control points on the edges of the base triangle of a construction tetrahedron (the relation is described in Section 4.2).

Case 1: The original trunctet is built inside the ***upper*** *trunctet only.*

Sign of weights after deformation	***Construction Tetrahedron to be used***
All of w_{1100}, w_{1010} and w_{0110} <0.	Upper Tetrahedron
All of w_{1100}, w_{1010} and w_{0110} >0.	Lower Tetrahedron
w_{1100}, w_{1010} and w_{0110} have different signs.	Double Tetrahedron

Case2: The original trunctet is built inside the ***lower*** *trunctet only.*

Sign of weights after deformation	***Construction Tetrahedron to be used***
All of w_{1100}, w_{1010} and w_{0110} <0.	Lower Tetrahedron
All of w_{1100}, w_{1010} and w_{0110} >0.	Upper Tetrahedron
w_{1100}, w_{1010} and w_{0110} have different signs.	Double Tetrahedron

Case 3: The original trunctet is built inside a double tetrahedron.

In this case, check both upper tetrahedron and lower tetrahedron using the tables in *Case 1* and *Case 2*.

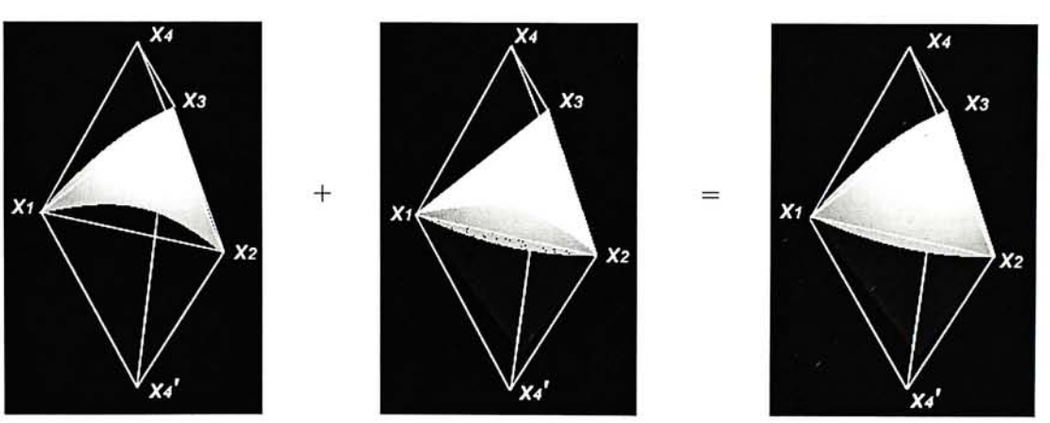


Figure 4.18 *One of the edge weights (w_{1100}) is positive.*

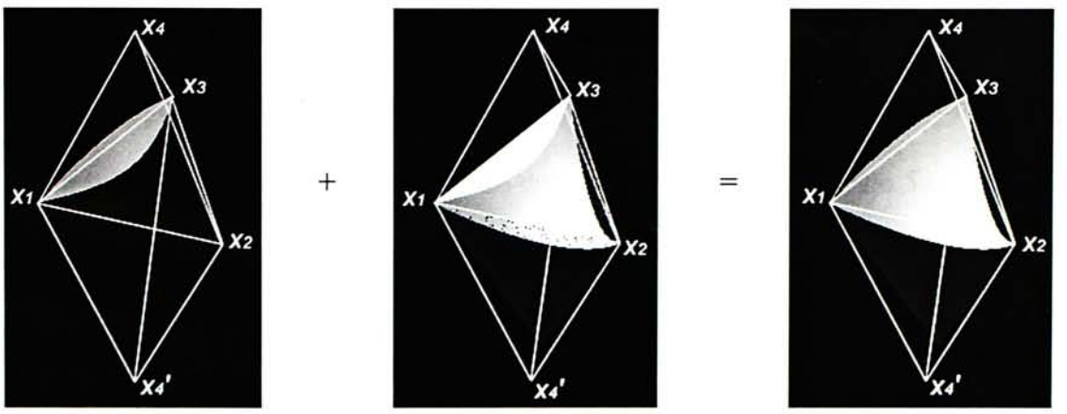


Figure 4.19 *Two of the edge weights (w_{1010} and w_{0110}) are positive.*

4.4.1.6 Surface Subdivision

In some extreme cases, the original surface, S, within a single tetrahedron (with base triangle (x_1, x_2, x_3)) may be deformed (deformed surface S_{ffd}) so that it is no longer bounded by its construction tetrahedron (Section 4.2 explained how this case can be detected). That means more than one trunctet is needed to represent that surface. This is done by subdividing the surface.

The subdivision algorithm is to pick a central surface point, c on S_{ffd}. This is found in the same manner as locating surface point in Section 4.4.1.1 with the "intersecting line" passing through the centroid of the base triangle. Three new construction tetrahedrons with base triangles $(x_1, x2, c)$, (x_2, x_3, c) and (x_3, x_1, c) are then constructed *(Figure 20a)*.

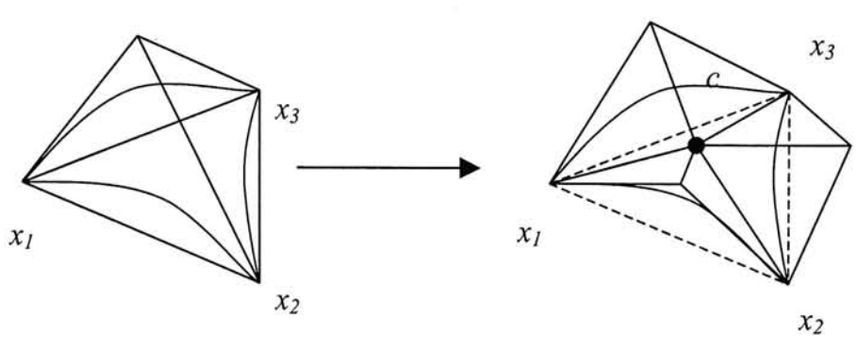


Figure 4.20a *Subdividing a surface into three.*

Repeat this process if needed until all divided surfaces can be bounded by its construction tetrahedron. The following figure shows a 2D example explain how the subdivision algorithm work.

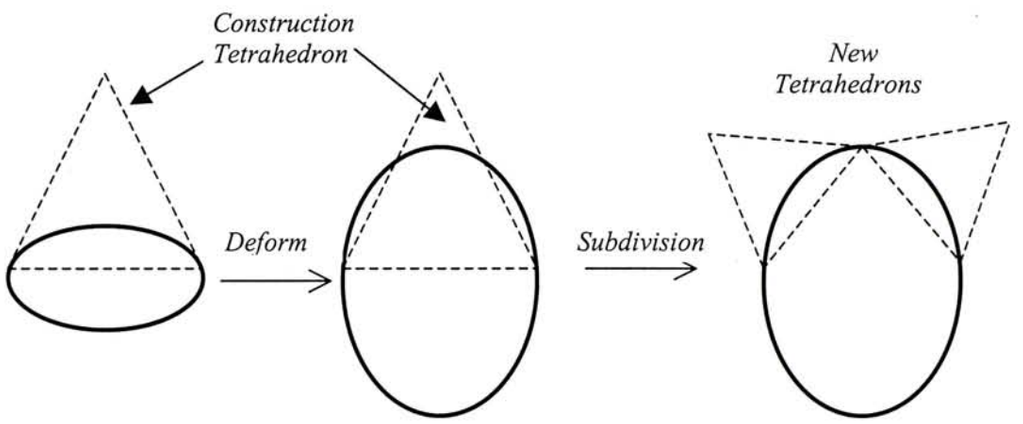


Figure 4.20b *Subdivision Process.*

This "unbounded surface problem" can always be solved by the subdivision process. Consider the angle $\phi(90^\circ<\phi<180^\circ)$ in the following figure.

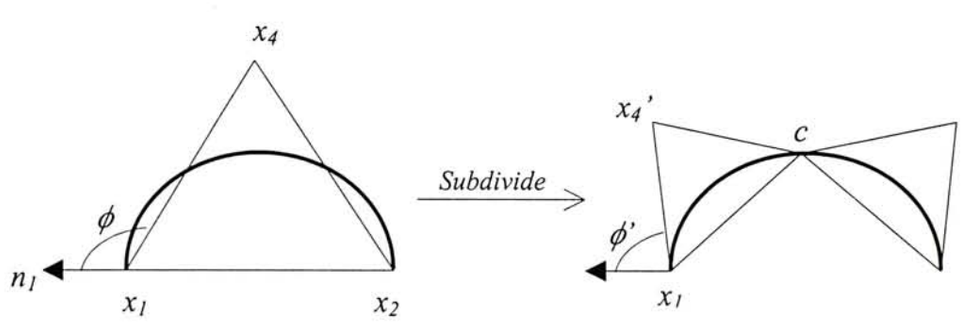


Figure 4.21 *Subdivision Process.*

After subdivision, new ϕ' is always smaller than ϕ. Repeat the subdivision process until $\phi < 90^\circ$. The problem is always solvable.

Figure 4.22a *Subdivide one quadric surface into 15 pieces.*

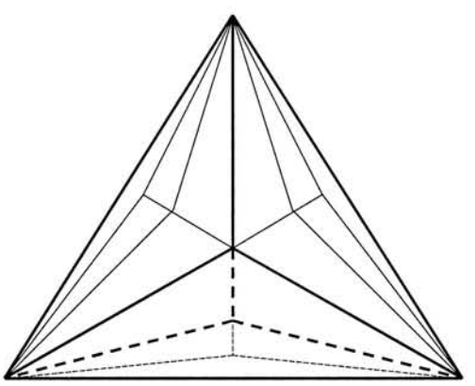


Figure 4.22b *Top view of subdivided base triangles of the model in figure 4.22a.*

4.4.2 Deforming Entire Solid Model

Given a solid model in CSR, the algorithm for deforming the solid is listed below.

Symbols:
S – CSR Solid.
P – Polyhedron core.
f – Number of facets on P.
F_i – Facet on P, for $i = 1$ to f.
v – Number of vertices on P.
V_q – Vertex on P, for $q = 1$ to v.
t^P_i – Protrusion trunctet on F_i.
t^D_i – Depression trunctet on F_i.
T^U – Upper tetrahedron.
T^L – Lower tetrahedron.
$A^{P/D}$ – $\{w_{1100}, w_{1010}, w_{0110}$ of t^P_i or t^D_i (set of weights on base triangle's edges)$\}$.
$B^{P/D}$ – $\{w_{1001}, w_{0101}, w_{0011}$ of t^P_i or t^D_i (set of weights on side edges)$\}$.
$()_{ffd}$ – Deformed points or object.
$FFD()$ – Free-form deformation function.

Input:
$S, P, f, F_i, v, V_q, t^P_i, t^D_i$.

Output:
$(S)_{ffd}$.

// Lattice Construction
Size of lattice: W*H*D
W = max x_q
H = max y_q
D = max z_q
$\forall$ $V_q = (x_q, y_q, z_q)$.

Origin of the Lattice: (X, Y, Z)
X = min x_q
Y = min y_q
Z = min z_q
$\forall$ $V_q = (x_q, y_q, z_q)$.

Number of control points of lattices in :
x-direction: int(W/D_{max})
y-direction: int(H/D_{max})
z-direction: int(D/D_{max})
where $D_{max} = Max(V_i - V_j)$ $\forall i, j$

// Deform polyhedron core.
for q = 1 to v
{
$(V_q)_{ffd} = FFD(V_q)$
}
Construct a deformed polyhedron core using $(V_1)_{ffd}, \ldots, (V_v)_{ffd}$.

```
// Deform trunctets on each facet.
for each facet
 {
  if (F_i is a facet not obtained by subdivision)
    {
     //Deform three base vertices vt_1, vt_2, vt_3 of F_i.
     (vt_1)_ffd = FFD(vt_1)
     (vt_2)_ffd = FFD(vt_2)
     (vt_3)_ffd = FFD(vt_3)

     // Locate 6 surface points.
     Get 6 surface points pt1, pt2, pt3, pt4, pt5, pt6 from t^P_i or t^D_i.

     // Deform 6 surface points.
     (pt_1)_ffd = FFD(pt_1)
     (pt_2)_ffd = FFD(pt_2)
     (pt_3)_ffd = FFD(pt_3)
     (pt_4)_ffd = FFD(pt_4)
     (pt_5)_ffd = FFD(pt_5)
     (pt_6)_ffd = FFD(pt_6)
    }
  // Create deformed trunctets.
  // Assume the trunctet is constructed inside the upper tetrahedron.
  Construct T^U over F_i (base vertices: (vt_1)_ffd, (vt_2)_ffd, (vt_3)_ffd).
  Solve A^P and B^P of t^P_i inside T^u using (pt_1)_ffd, ..., (pt_6)_ffd.

  if (any weights in B^P < 0) // Surface is unbounded by the tetrahedron and subdivision needed.
    {
     Subdivide(F_i) // Call subdivision function.
    }
  else if (all weights in A^P < 0) // No depression trunctet required.
    {
     Construct t^P_i
     t^D_i = null
    }
  else if (all weights in Set A^P > 0 and all weights in Set B^P > 0) // Surface on the other side.
    {
     // The trunctet is constructed inside the lower tetrahedron.
     Construct T^L over F_i (base vertices: (vt_1)_ffd, (vt_2)_ffd, (vt_3)_ffd).
     Solve A^D and B^D of t^D_i inside T^L using (pt_1)_ffd, ..., (pt_6)_ffd.

     if (any weights in B^D < 0) // Subdivision needed.
       {
         Subdivide(F_i) // Call subdivision function.
       }
     else if (all weights in Set A^P < 0) // No protrusion trunctet required .
       {
         Construct t^D_i
         t^P_i = null
       }
    }
  else if (weights in A^P have different signs) // double tetrahedron needed.
    {
     Construct t^P_i
     Construct T^L over F_i (base vertices: (vt_1)_ffd, (vt_2)_ffd, (vt_3)_ffd).
     Solve A^D and B^D of t^D_i inside T^L using (pt_1)_ffd, ..., (pt_6)_ffd.
     Construct t^D_i
    }
}
Fill two–sided gaps
```

// Obtain deformed Solid.
$(S)_{ffd} = ((P)_{ffd} \cup t^P_i) - t^D_j \quad \forall i, j$

The algorithm for subdividing a surface is listed below:
// Subdivision Function.
Input: A facet F_i (with $(vt_1)_{ffd}$, $(vt_2)_{ffd}$, $(vt_3)_{ffd}$ and $(pt_1)_{ffd}$, …, $(pt_6)_{ffd}$).
output: new base vertices and $(pt_1)_{ffd}$, …, $(pt_6)_{ffd}$.

Locate a "centroid" on the surface.

// Three new facets formed
F_{f+1} : $(vt_1)_{ffd}$, $(vt_2)_{ffd}$, centroid.
F_{f+2} : $(vt_1)_{ffd}$, $(vt_3)_{ffd}$, centroid.
F_{f+3} : $(vt_2)_{ffd}$, $(vt_3)_{ffd}$, centroid.

Add F_1, F_2, F_3 to the list of facets.

4.4.3 Comparison on Different Approaches

The following table gives a comparison on the three different approaches for applying FFD on CSR solid model. Those three approaches are described in Section 4.3.1 (method 1), 4.3.2 (method 2) and 4.4 (method 3).

	Method 1	*Method 2*	*Method3*
Lattice Structure	Parallelpiped Lattice	Parallelpiped Lattice	Parallelpiped Lattice
CSR parameters for controlling deformation	1. Normals directions at vertices 2. Vertices of Polyhedron core	1. Surface points around vertices 2. Vertices of Polyhedron core	1. Surface points over entire solid model. 2. Vertices of Polyhedron core.
Relative Computation Effort	Low	High	High

In fact, method 1 and method 2 can be considered as counter examples of applying FFD on CSR and the reasons are given in the corresponding sections. (Section 4.3.1 and 4.3.2)

4.5 Conversion of CSG solid models into CSR

General steps of deforming a CSG solid model:

1. Locate the primitives (leaves), such as cylinder, sphere or block, of a CSG solid model in its binary tree. (Example given in *Figure 4.23*)

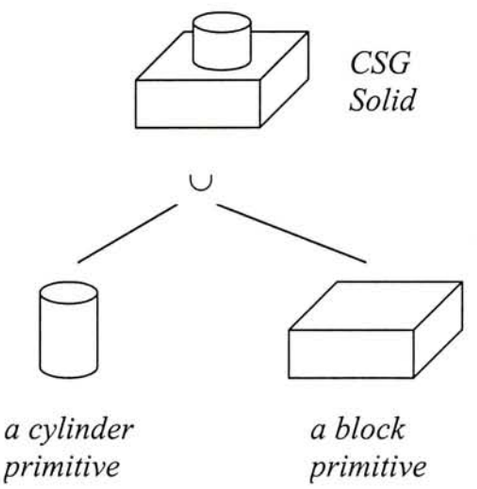


Figure 4.23 *A CSG binary tree.*

2. Reconstruct those primitives as subtrees of intersection of halfspaces.

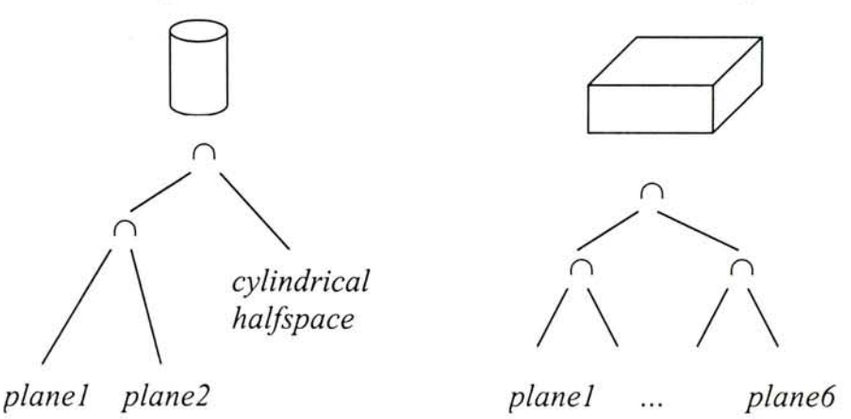


Figure 4.24 *Subtree of intersection of halfspaces.*

3. Convert the halfspaces into CSR objects. (will be explained in Section 4.5.1)
4. Apply FFD on the converted halfspaces and the core of the CSR object.

5. Check each converted halfspaces to see whether they are deformed properly by monitoring the sign of the weights of the trunctets. Subdivide trunctets if needed. Then, fill the two-sided gaps.
6. Generate a deformed solid model by using the CSG tree with the deformed primitive core and halfspaces at the leaves. (*Figure 4.25*)

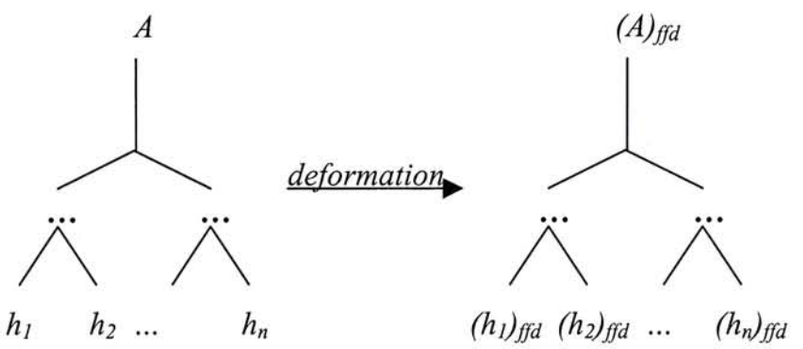


Figure 4.25 *Deforming halfspaces.*

4.5.1 Converting Halfspaces into CSR objects

In this section, the idea of converting halfspaces into CSR object is given. The number of patches used to model a halfspace is not fixed. Greater flexibility for the Free-Form Deformation can be attained if more trunctets are used. However, the time taken for deforming and displaying the deformed object will be increased. The basic idea of converting CSG primitive to CSR is to use a polyhedron core to approximate the shape of the CSG primitive, where the vertices on the polyhedron core should lie on the surface of the CSG primitive. Trunctets are then constructed on the vertices of the core to model the shape of the halfspaces. A few examples are shown illustrating the conversion of halfspaces. Experiment results are also given.

CSR model of a sphere:

The input polyhedron core is composed of two pyramids. 3-sided trunctets are attached to the triangular faceted surfaces to obtain a sphere represented in CSR.

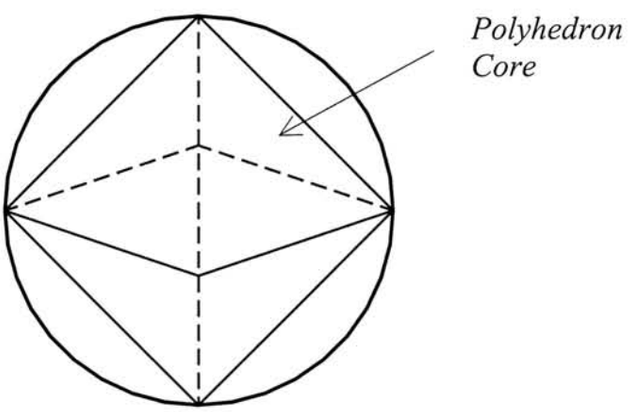


Figure 4.26 *A CSR Sphere.*

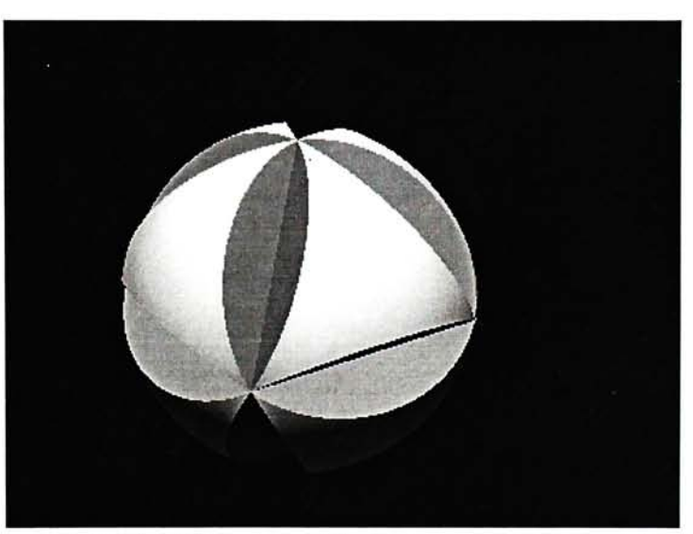


Figure 4.27 *A CSR sphere (Two-sided gaps are not filled).*

CSR model of a cylindrical surface:

Similar to the modeling of a sphere, an input polyhedron is constructed approximating the cylindrical surface as shown in *Figure 4.28*.

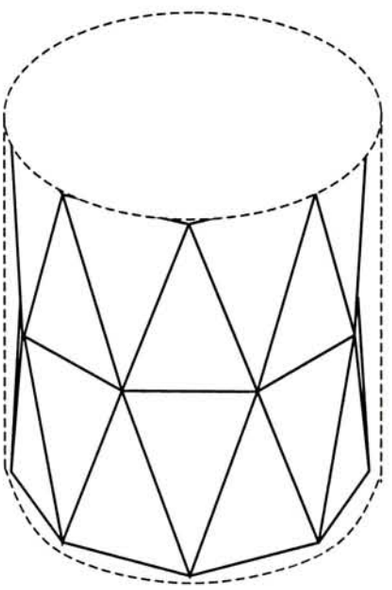


Figure 4.28 *Polyhedron core of a CSR cylinder.*

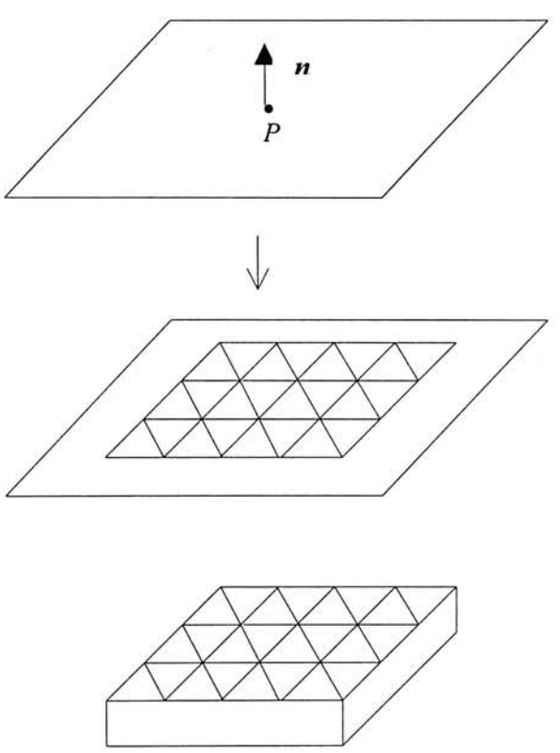
3-sided trunctets are then constructed on each of the triangular facet of the polyhedron giving a CSR representation of the cylindrical surface as shown in *Figure 4.29*.

Figure 4.29 *A CSR cylindrical surface.*

CSR model of planes:

After deformation, a plane may become a curved surface and has to be represented by a CSR object. It is thus essential to model a plane by CSR object before the deformation in order to tackle the possible shape change.

All primitives, trunctets and polyhedron core, in CSR are finite elements that occupy finite volume in 3D space. However, a plane is a halfspace defining infinite volume in 3D space. Hence, it is impossible to model a *"true plane"* in CSR. An alternative is to decide how much "space" is needed and model a *"virtual plane"* which is used to model a CSG plane for deformatoin. A *"virtual plane"* is a volume with a planar surface, and formed in a similar way as other CSR objects. The following figure shows the step of modeling a *"virtual plane"*.



Given a CSG plane.

Define a faceted area around P

Create a "box" to represent the solid part.

Figure 4.30 *Construct "virtual plane".*

Construct tetrahedrons over facets so that the box become a polyhedron core with five rectangular side and one faceted side which is used to represent the plane.

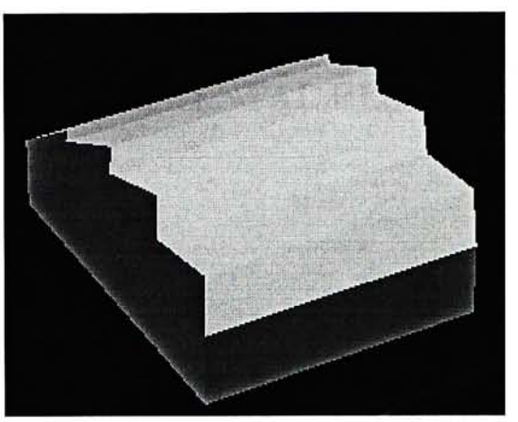


Figure 4.31 *A Deformed polyhedron core.*

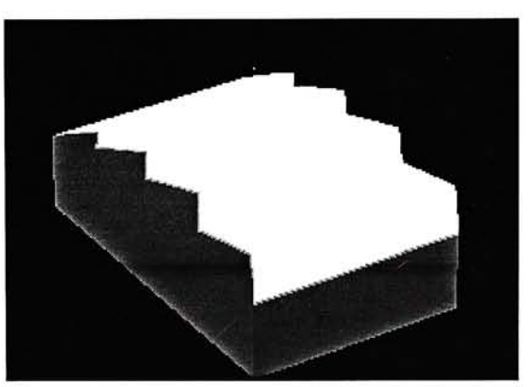


Figure 4.32 *Attach trunctets on the polyhedron core and gives a deformed virtual plane.*

CHAPTER 5

IMPLEMENTATION

AND

EXPERIMENTAL RESULTS

5.1 Implementation

An experimental system was implemented using a CSG geometric kernel *SvLis*. It is written in C++ and available as a library of header files and procedures that can be used to build applications programs.

An important feature of *SvLis* is that it can be used to model solids that can be expressed as an implicit polynomial inequality of up to a polynomial degree of eight.

SvLis models polynomial primitive inequalities by performing arithmetic with planes. Polynomial of a quadric patch is obtained by doing arithmetic with 4 planes:

$$s = L_s(x,y,z) = 0 \qquad t = L_t(x,y,z) = 0$$

$$u = L_u(x,y,z) = 0 \qquad v = L_v(x,y,z) = 0$$

which are the conversion between barycentric and Cartesian coordinates. At the same time, these 4 planes are the faces of the construction tetrahedron.

Consider a trunctet with vertices v_1, v_2, v_3 and v_4. The polynomial of the quadric patch is obtained by substituting $s = \frac{L_s(x,y,z)}{D_s}$, $t = \frac{L_t(x,y,z)}{D_t}$, $u = \frac{L_u(x,y,z)}{D_u}$ and $v = \frac{L_v(x,y,z)}{D_v}$ into

$$a_b(s,t,u,v) = w_{2000}s^2 + w_{0002}t^2 + w_{0020}u^2 + w_{0002}v^2 + 2w_{1100}st + 2w_{1001}sv$$
$$+2w_{0011}uv + 2w_{0110}tu + 2w_{1010}su + 2w_{0101}tv$$

where D_s is the distance from the vertex *v1* to the plane containing *v2, v3* and *v4.*

D_t is the distance from the vertex *v2* to the plane containing *v1, v3* and *v4.*

D_u is the distance from the vertex *v3* to the plane containing *v1, v2* and *v4.*

D_v is the distance from the vertex *v4* to the plane containing *v1, v2* and *v3.*

Hence, the quadric patch is a function of the four planes. i.e.,

$$a(x,y,z) = a_b(L_s(x,y,z), L_s(x,y,z), L_s(x,y,z), L_s(x,y,z))$$

Finally, by intersecting *a(x,y,z)* with the construction tetrahedron, a trunctet is obtained.

5.2 Experimental Results

Experiment 1: Local and Global deformation of a CSR object.

A CSR representation of a cylindrical object is constructed. The object is enclosed within an FFD lattice as shown in Figure 5.1. The object is deformed globally be moving four of the lattice vertices as illustrated in Figure 5.2. By using a lattice with more control points, the object can be deformed locally as shown in Figure 5.3

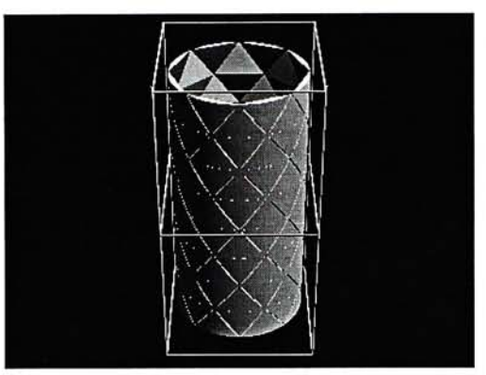


Figure 5.1 *Impose a FFD Lattice on an input object.*

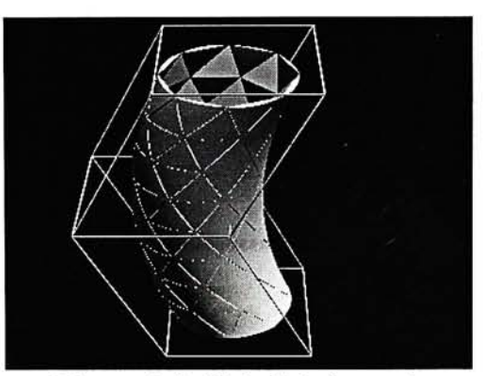


Figure 5.2 *Global deformation.*

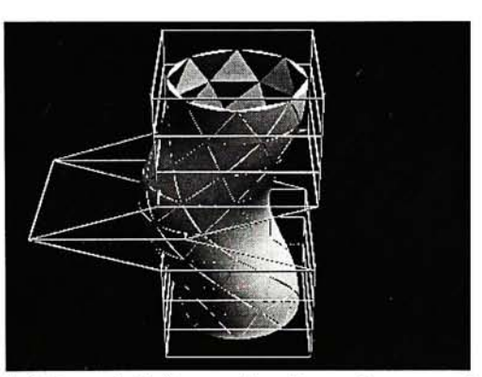


Figure 5.3 *Local deformation by using another lattice.*

Experiment 2: Construct a "Hair Dryer" by using FFD.

A hair dryer is composed of a nozzle, a body and a handle. Both the nozzle and handle are made by deforming a cylinder.

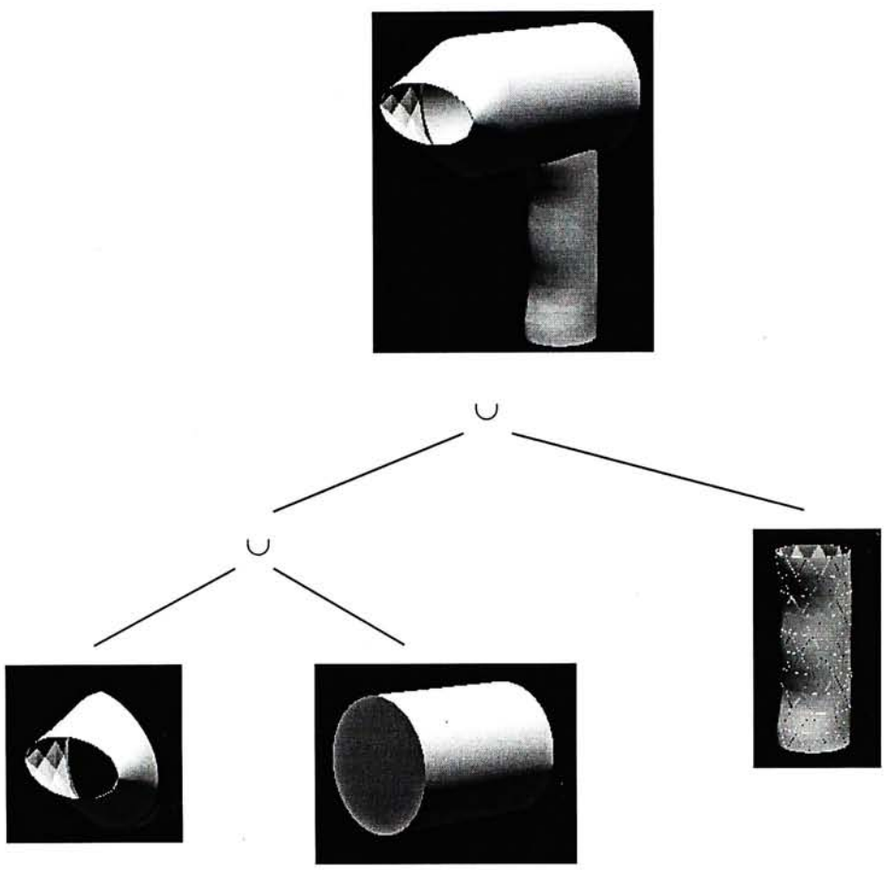


Figure 5.4 *Binary tree of a Hair Dryer.*

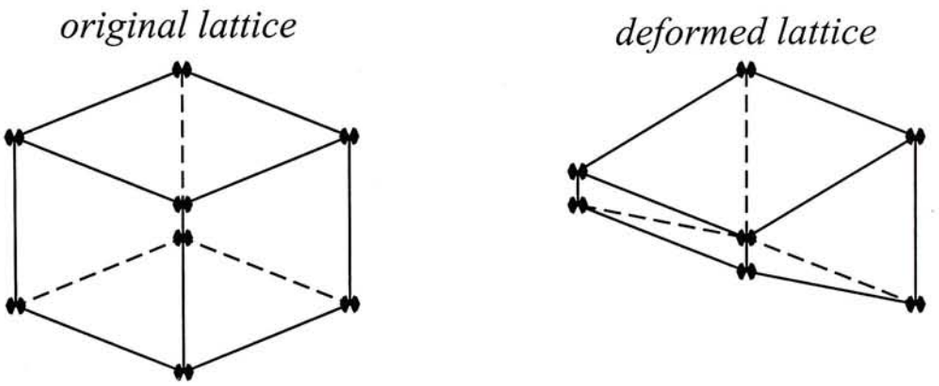


Figure 5.5 *Lattice applied on a cylinder to form the nozzle.*

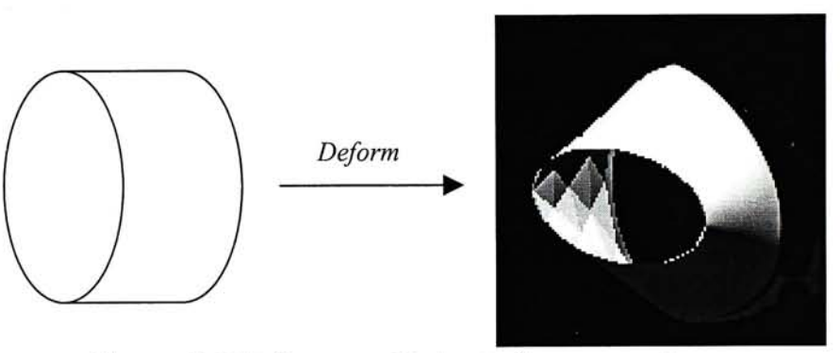


Figure 5.6 *Deform a cylinder to form a nozzle.*

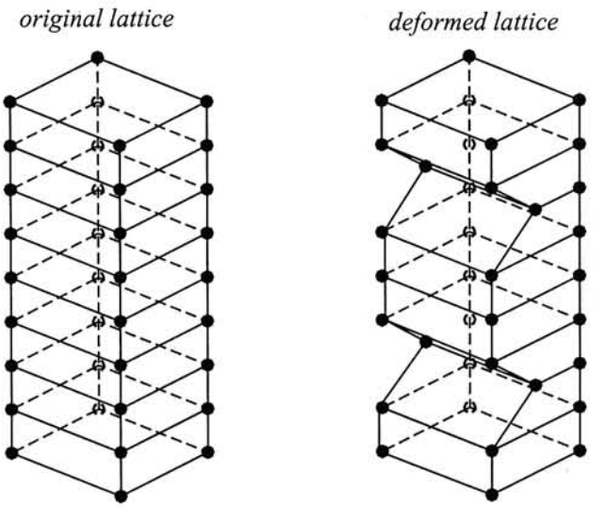


Figure 5.7 *Lattice applied on a cylinder to form the handle.*

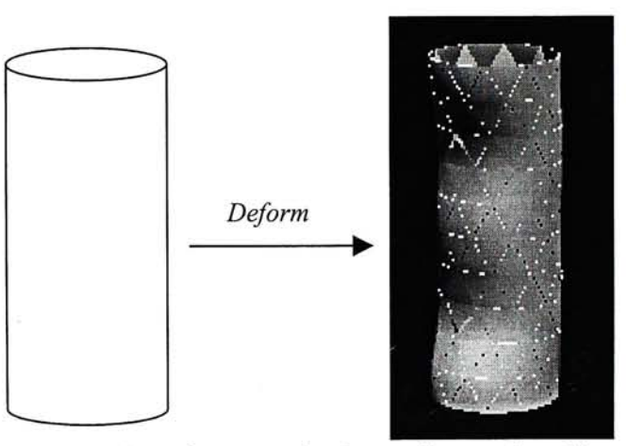


Figure 5.8 *Deform a cylinder to form a handle.*

Experiment 3: Deforming a toothpaste. Similar to the construction of the hair-dryer handle, the body of the toothpaste is constructed by deforming a cylinder.

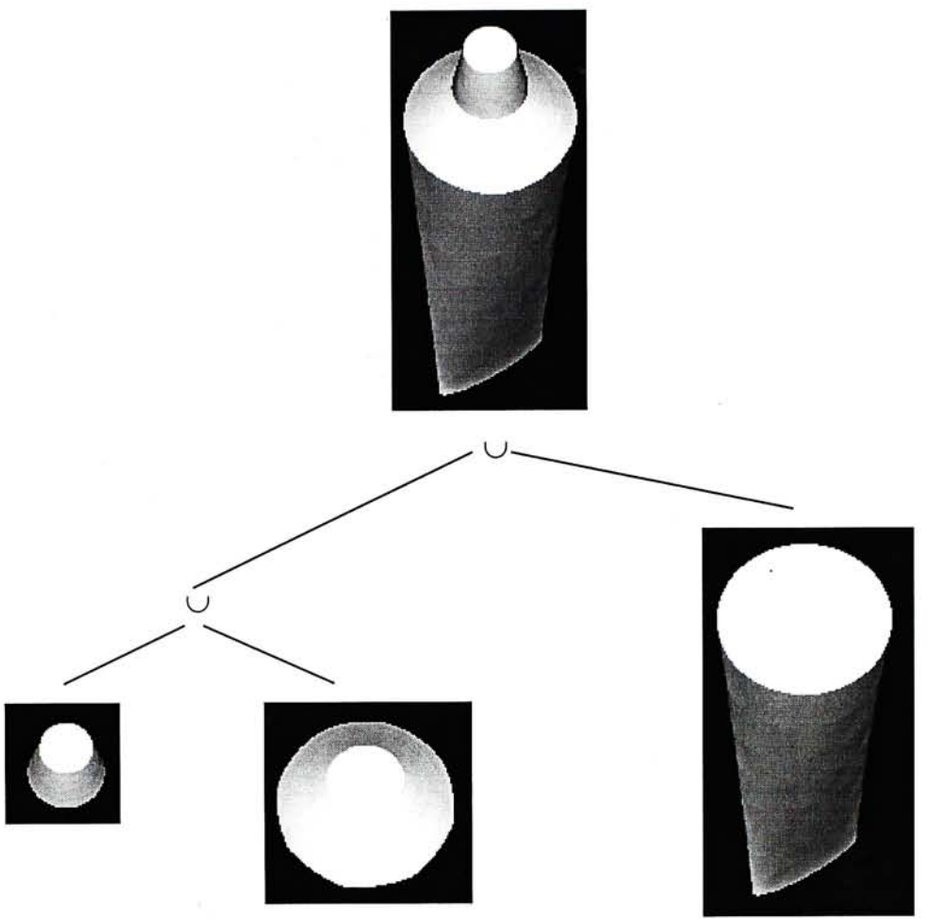


Figure 5.9 *Binary tree of a toothpaste.*

The toothpaste is deformed *(Figure 5.8)* by using a lattice similar to the one used in *Figure 5.2*. This experiment demonstrated the capability of applying FFD on a deformed model

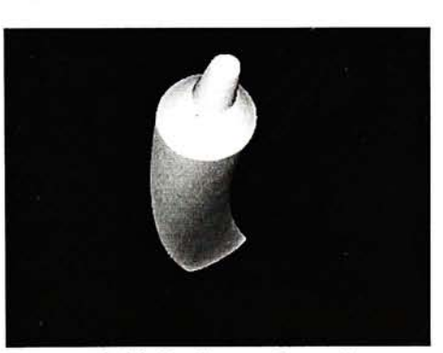


Figure 5.10 *Deformed toothpaste.*

Experiment 4: A cylinder modeled in CSG representation is converted into a cylinder in CSR. The cylinder is then deformed. Finally, a square hole is subtracted from the cylinder. Two sided gaps is not filled for simplicity.

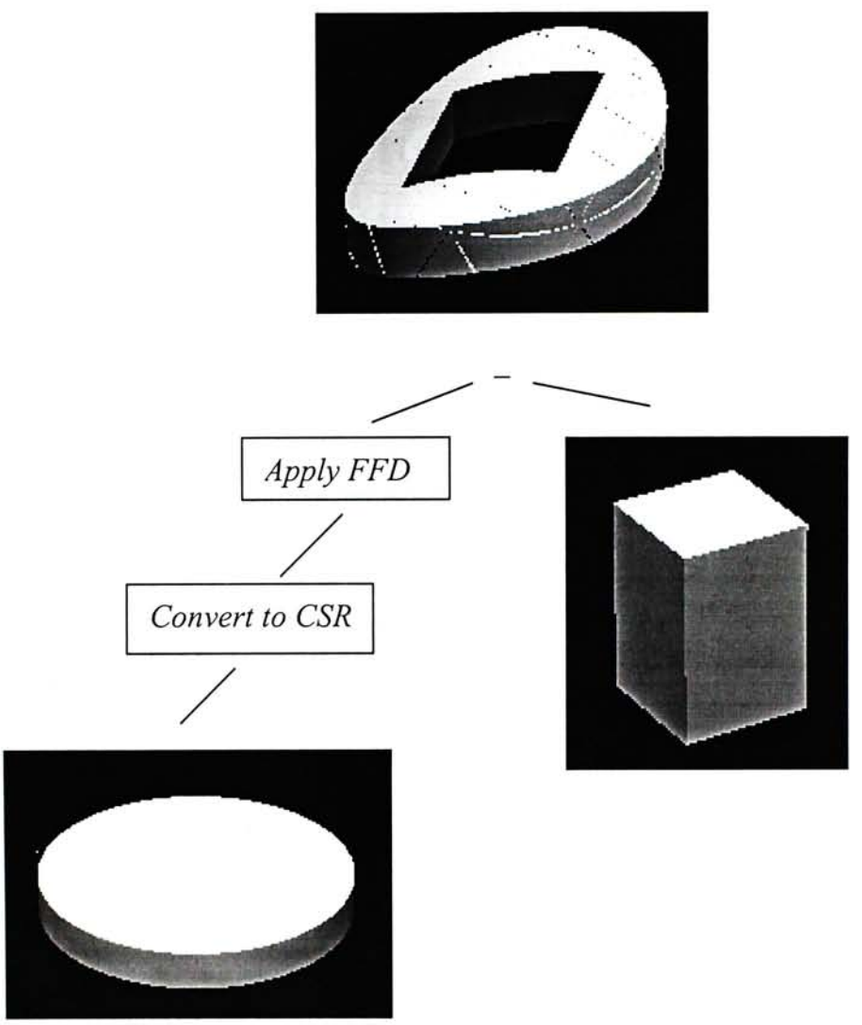


Figure 5.11 *Deformed short cylinder with square hole subtracted.*

Experiment 5: Deforming the handle of a dumbbell. This experiment demonstrates that the deformation can be applied to a specific element of a CSG solid.

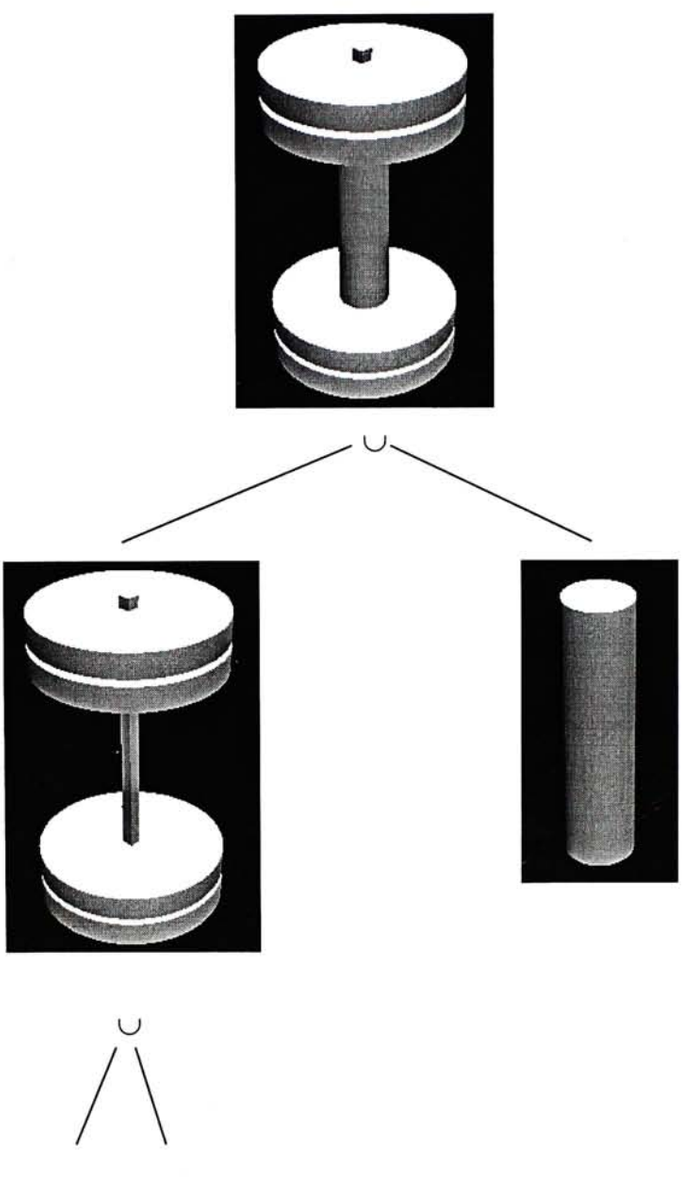


Figure 5.12 *CSG tree of the undeformed dumbbell.*

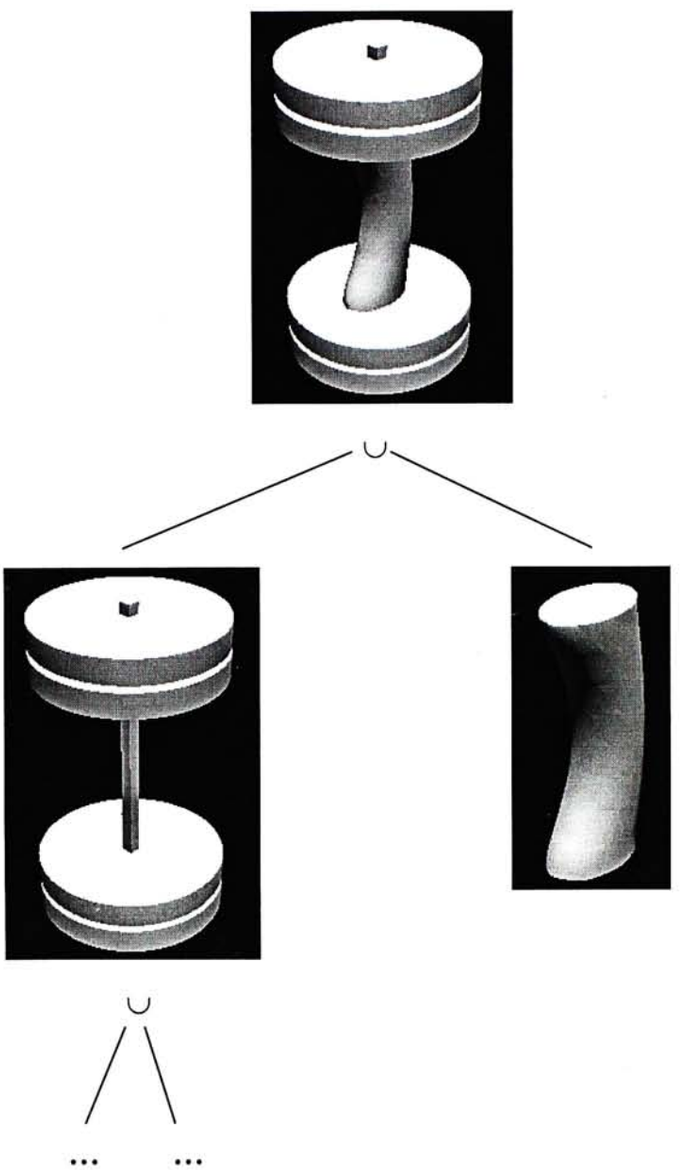


Figure 5.13 *CSG tree of the deformed dumbbell.*

Remark

Efficiency of Deformation:

The time taken to deform a model is determined by two factors:

- Number of control points on the lattice. (i.e. $(l+1) \times (m+1) \times (n+1)$).
- Number of patches.

The following table shows statistics on the time taken to deform 128 patches with lattice composed of different number of control points.

Number of control points	12	16	18	20	24	28	30	40	50
Time taken (sec)	0.200	0.221	0.230	0.24	0.267	0.29	0.30	0.363	0.421

The following table shows statistics on the time taken to deform a model with different patches by a 12-control point lattice.

Number of patches	128	160	192	224	246	286	320
Time taken (sec)	0.200	0.250	0.300	0.350	0.400	0.450	0.500

Remark: The experiment is done in a SGI power challenge multiprocessor supercomputer.

CHAPTER 6

CONCLUSION AND SUGGESTIONS FOR FURTHER WORK

6.1 Conclusion

In this research, several approaches for applying FFD on CSR is analyzed. The surface fitting approach is found to be the most satisfactory in fulfilling the objective. A detailed study on this approach is performed. Both algorithm and experiment results are given. CSR is used to represent a deformed solid model, so that further operation such as Boolean operation can be applied to the deformed model. Several ways to apply FFD on CSR objects have been studied, including the deformation of normals at the polyhedron core vertices, the deformation of neighborhood points around vertices of the polyhedron core, and the Surface fitting approach. This thesis also explain how the FFD lattice should be set according to the given CSR solid model.

The surface fitting approach is proposed to apply Free-Form Deformation on a solid object in CSR. A CSR solid model is formed by the union and subtraction of trunctets from an input polyhedron core. The deformation is affected by deforming trunctets and the polyhedron core.

The polyhedron core is deformed by applying FFD to the polyhedron vertices. The deformation of a trunctet is performed by deforming six evenly

distributed surface points on the quadric patch of a trunctet using FFD. Finally, by evaluating the weights on the trunctet, a deformed trunctet is constructed. A trunctet can be convex, concave or saddle shape.

In the process of constructing trunctets, upper, lower or double tetrahedron should be used according to the convexity of the quadric patch and whether the patch intersect the facet. Rules are given to determine which tetrahedron should be used. The decision is made based on an analysis of the weights of a trunctet. A protrusion trunctet formed inside an upper tetrahedron is used to model convex surface. A depression trunctet formed inside a lower tetrahedron is used to model concave surface. A saddle shape surface can be modeled by a protrusion or a depression trunctet. Double tetrahedron, as well as both protrusion and depression trunctets is used when a surface associated with a facet changes from concave to convex.

After deformation, if a trunctet is not able to enclose its surface patch, surface subdivision is required on that patch so that more trunctets is used to model the patch.

This deformation technique is extended for CSG objects by converting the CSG object into a CSR object. Experiment results were presented to show local and global deformation, the use of CSG and CSR objects in deformation, the conversion from CSG to CSR, deformation specific element in a CSG tree, and further operations after deformation.

A limitation of the proposed approach is that the best location of the lattice control points for deformation is not precisely controlled. Since FFD is used for the deformation, exact location of tan object point in the deformed solid cannot be controlled..

6.2 Suggestions for Furtherwork

The performance of the FFD system may be greatly enhanced if there is new stand alone modeler that specifically designed for the modelling and display of trunctets. By developing a special trunctet display algorithm for *"SvLis"*, the performance of the whole system can be improved. The algorithm of ploygonization of implicit surface patches developed by K. C. Hui and Z. H. Jiang [17] can be adopted for display propose. In their approach, a patch is adaptively subdivided into smaller tetrahedrons according to certain criteria. The result of polygonization is a set of triangular facets approximating the surface that can be used for visualization analysis.

In addition, the deformation flexibility may be pushed to an upper level if other deformation technique such as direct FFD method described in Section 2.1.2 is used. Moreover, cubic patches (instead of quadric patches used now) may also be used. Since cubic patch is capable of representing more complex surface, the number of patches used to model a given object can be reduced. Comparison and further analysis can be performed.

Since applying FFD on CSR objects is the main focus of this research, most of the original CSR models and CSG models are not made automatically. A more complete system can be made if all CSR object can be formed automatically by simply specifying a few parameter of a solid object. An advanced method to convert CSG object into CSR object can also be studied. J. P. Menon[1] also described how CSG representations of free form solids

bounded by algebraic patches can be computed as Boolean compositions of trunctets, shells, cores and input polyhedrons.

Finally, the interface between the system and end-users can be further improved so as to provide a user friendly environment for the operation.

REFERENCES

[1] J.P. Menon, *Constructive Shell Representations*, Technical Report CPA92-5, The Sibley School of Mechanical and Aerospace Engineering, Cornell University, 1992.

[2] Baining Guo, *Modeling Arbitrary Smooth Objects with Algebraic Surfaces*, Ph.D Thesis, Department of Computer Science, Cornell University, 1991.

[3] Thomas W. Sederberg, Scott R. Parry, *Free-Form Deformation of Solid Geometric Models*, SIGGRAPH'86, ACM Comp. Graph., 20(4), 151-160, 1986.

[4] S. Coquillart, *Extended free-form deformation: A sculpturing tool for 3D geometric modeling.* SIGGRAPH'90, ACM Comp. Graph. 24(4), 187-196, 1990.

[5] P. Kalra, A. Mangili, N. M. Thalmann and D. Thalmann., *Simulation of facial muscle actions based on rational free-form deformation*, EUROGRAPHICS'92, Computer Graphic Forum 2(3), C59-C69, 1992.

[6] F. Lazarus, S. Coquillart and P. Jancene, *Deformations axiales interactives,* GROPLAN'92, 117-124, 1992.

[7] P. Borrel and D. Bechmann, *Deformation of N-dimensional objects,* Symposium on Solid Modeling Foundations and CAD/CAM applications, ACM Press, Int. J. Computational Geometry Appl., 1(4), 427-453, 1991.

[8] W. M. Hsu, J. F. Hughes and H. Katufman, *Direct manipulation on free-form deformation,* SIGGRAPH'92, ACM Comp. Graph., 26(2), 177-184, 1992.

[9] Eric Bardinet, Laurent D. Cohen and Nicholas Ayache, *Fitting 3-D Data Using Superquadrics and Free-Form Deformations,* 1994IEEE.

[10] Thomas J. True, John F. Hughes, *Volume Warping*, 1992IEEE.

[11] Hidefumi Wakamatsu, Shinichi Hirai, and Kazuaki Iwata, *Modeling of Linear Objects Considering Bend, Twist, and Extensional Deformation*, IEEE Int. Conf. On Robotics and Automation, 433-438, 1995.

[12] Petros Faloutsos, Michiel van de Panne, Demetri Terzopoulos, *Dynamic Free-Form Deformations for Animation Synthesis*, IEEE Tran. On Visualization and CG, Vol. 3, No. 3, 201-214, 1997.

[13] James C Cavendish, *Integrating feature-based surface design with freeform deformation*, Computer-Aided Design, Vol. 27, No. 9, 703-711, 1995

[14] Adrian Bowyer, *SvLis – Introduction and User Manual.*

[15] Dahmen, W., "*Smooth piecewise quadric surfaces*", Mathematical Methods in CAGD, eds. T. Lyche and L. Schumaker, Academic Press Inc., pp. 181-193, 1989.

[16] Lodha, S., and Warren, J., "*Bezier representation for quadric surface patches*", Computer Aided Design, vol. 22, no. 9, pp. 574-579, November 1990.

[17] K.C. Hui and Z. H. Jiang, "Tetrahedra based adaptive polygonization of triangular implicit surface patches". Computer Graphics Forum, vol. 18, no. 1, pp. 57-68, March 1999.

[18] Henry J. Lamousin and Warren N. Waggenspack, "NURBS-Based Free-Form Deformations". Computer Graphics and Applications, pp. 59-65, 1994