

ShapeWright:
Finite Element Based Free-Form Shape Design

by

George Celniker

S.M., Mechanical Engineering
Massachusetts Institute of Technology
January, 1981

B.S., Mechanical Engineering
Massachusetts Institute of Technology
May, 1979

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy
in Mechanical Engineering

at the

Massachusetts Institute of Technology

September, 1990

© Massachusetts Institute of Technology, 1990, All rights reserved.

Signature of Author _____
Department of Mechanical Engineering
August 3, 1990

Certified by _____
Professor David C. Gossard
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Department Committee

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

NOV 08 1990

LIBRARIES

ShapeWright:
Finite Element Based Free-Form Shape Design
by
George Celniker

Submitted to the Department of Mechanical Engineering on
September, 1990 in partial fulfillment of the requirements for the
degree of Doctor of Philosophy in Mechanical Engineering.

Abstract

The mathematical foundations and an implementation of a new free-form shape design paradigm are developed. The finite element method is applied to generate shapes that minimize a surface energy function subject to user-specified geometric constraints while responding to user-specified forces. Because of the energy minimization these surfaces opportunistically seek globally fair shapes. It is proposed that shape fairness is a global and not a local property. Very fair shapes with C^1 continuity are designed.

Two finite elements, a curve segment and a triangular surface element, are developed and used as the geometric primitives of the approach. The curve segments yield piecewise C^1 Hermite cubic polynomials. The surface element edges have cubic variations in position and parabolic variations in surface normal and can be combined to yield piecewise C^1 surfaces. These are called deformable curves and surfaces because they respond to user inputs much like elastic beams and membranes.

The properties of these geometric primitives have been designed to enable a three phase interactive approach to defining very fair free-form shapes. The shape's character lines are created with deformable curve segments. These character lines are then "skinned" with a deformable surface. The final shape is sculpted interactively by applying loads to the surface to control the surface shape between character lines. Very fair free-form shapes have such applications as the design of automobiles, ships, ceramics, and air-supported buildings as well as the modeling of natural forms.

Thesis Advisor:
Thesis Committee:

Prof. David Gossard
Prof. Tony Patera
Prof. Alex Pentland

Dedication

To Matthew Benjamin and to those who follow.

Acknowledgements

My motivations and drive to explore that have resulted in this effort, are in no small part due to my associations with my family and friends. Before all others, I thank my wife, Nancy Stevens, for encouraging me to make the changes to do the things that are important. I thank my parents and sisters for their unquestioning love and for having made the world a valuable place for me. I thank all my friends in the extended M.I.T. community that have made this such an interesting and enjoyable time in my life.

The direction and quality of this work greatly benefited from the input of the members of my committee, Sandy Pentland and Tony Patera, and in particular from the advice and guidance of my thesis advisor, David Gossard. I also thank David Gossard for the support he has given in furthering my career.

The time and opportunity to pursue this study were provided by the people of the Schlumberger Technology Corporation. I thank them for having supported me throughout this endeavor and for showing their confidence and belief in my abilities.

This research was made possible in part by grants from the Ford Motor Company, and the Eastman Kodak Company.

Table of Contents

Abstract	3
Dedication.....	4
Acknowledgements.....	4
Table of Contents	5
List of Figures.....	7
Nomenclature.....	9
1 Introduction	13
1.1 Computer Aided Geometric Design	13
1.2 The ShapeWright Design Paradigm	17
1.3 ShapeWright Design Issues	18
1.4 Scope of Work.....	20
2 Previous Work.....	22
2.1 Precursors to Energy-Based Shape Design.....	22
2.2 The Foundations of Computer Aided Geometric Design	25
2.3 Shape Building and the User Load Problem	26
2.4 Parametric Shape Technical Issues	28
2.5 Topology	34
3 Deformable Models.....	39
3.1 Parametric Description of Shape.....	39
3.2 Modeling of an Elastically Deformable Surface.....	48
3.3 The Energy Functionals.....	49
3.4 Properties of the Energy-Based Deformable Models.....	50
3.5 Minimize a Functional or Solve a Differential Equation.....	56
3.6 Time, Mass and Dynamics	59
3.7 Forces.....	61
3.8 Soap Films, Plane Strain, Plates and Shells.....	64
3.9 Chapter 3 Summary	69
4 Curve and Surface Geometric Primitives	71
4.1 Finite Difference Solutions	72
4.2 The Finite Element Method.....	77
4.3 Mass, Damping and Integrating Through Time	84
4.4 A Deformable Surface Primitive	88
4.5 A Deformable Curve Primitive.....	114
4.6 Chapter 4 Summary	118
5 Survey of Constraint Methods	120
5.1 Reduced Transformation Equations	121
5.2 Penalty Methods	123
5.3 Lagrange Multipliers	125
5.4 Stabilized Constraints	129

5.5 Stabilized Parametric Constraints	133
5.6 Chapter 5 Summary	137
6 Implementations	139
6.1 A Finite Difference Program for Deformable Surface Design, GGT	139
6.2 A Finite Element Program for Deformable Surfaces, ECS	151
6.3 A Finite Element Program for Deformable Curves, CUBIC	162
6.4 Rendering	167
7 Conclusions and Future Work	171
7.1 Thesis Summary	171
7.2 Conclusions	173
7.3 Future Directions	176
References	179

List of Figures

Figure 1.1)	Current free-form design paradigms.....	14
Figure 1.2)	The ShapeWright design paradigm	18
Figure 1.3)	Geometric constraints needed for skinning	19
Figure 2.1)	Trimmed and constrained surface patches for complicated shapes.....	38
Figure 3.1)	The parametric tangents and normal to a regular surface	42
Figure 3.2)	Curves in the parametric plane map to curves fixed in the surface	43
Figure 3.3)	The curvature and normal for a curve fixed in a surface.....	45
Figure 3.4)	Solutions for avoiding an obstacle	61
Figure 3.5)	External forces, a survey.....	62
Figure 4.1)	Discretizing the uv domain for a finite difference solution	73
Figure 4.2)	Template shape for surface finite difference equations	75
Figure 4.3)	Surface template coefficients for Figure 4.2	75
Figure 4.4)	Barycentric coordinates in 2 dimensions	89
Figure 4.5)	Edge tangents and normals of a Barycentric mapping triangle.....	93
Figure 4.6)	Triangular element with 3 nodes and 9 degrees of freedom.....	95
Figure 4.7)	The 'e' and 'f' functions used to build the shape functions.....	96
Figure 4.8)	Adjoining triangles in a finite element mesh	99
Figure 4.9)	Triangular element with 12 degrees of freedom	99
Figure 4.10)	Partial mesh of aligned and orthogonal edges	102
Figure 4.11)	Alternative coordinate systems for vertex nodal tangents	104
Figure 4.12)	Partial mesh for intersecting constrained edges.....	107
Figure 4.13)	Node constraint state due to edge constraints (see Figure 4.12).....	108
Figure 4.14)	Numerical integration formulae for triangles [Zienkiewicz_67]	111
Figure 4.15)	Intermediate matrix sizes and dependencies.....	112
Figure 4.16)	Hermite polynomials.....	115
Figure 4.17)	Curve element, 4 degrees of freedom.....	115
Figure 5.1)	The geometry of Lagrangian constraints	127
Figure 6.1)	Program structure to support interactive simulations	140
Figure 6.2)	Goblet shape after skinning initial character line set	144
Figure 6.3)	Goblet shape after skinning augmented character line set.....	144
Figure 6.4)	Sculpting the body of the goblet.....	146
Figure 6.5)	Sculpting the stem.....	146
Figure 6.6)	A point force dents the goblet.....	147
Figure 6.7)	An outer body panel for an automobile.....	147
Figure 6.8)	Design of the matching inner body panel.....	149
Figure 6.9)	A surface avoids an object when sculpted with increasing pressure	150

Figure 6.10)	ShapeWright deformable surface data structures.....	152
Figure 6.11)	User leverage through parameterization of shape	155
Figure 6.12)	A 5 element ShapeWright surface constrained to interpolate 5 point locations.....	157
Figure 6.13)	Views of a shape	158
Figure 6.14)	Variations on a theme of shape	159
Figure 6.15)	3 corners in uv shape map to 3 corners in 3D shape.....	160
Figure 6.16)	The Ford door: interfacing shape and analytic modeling	161
Figure 6.17)	Curve deforming to avoid an obstacle	164
Figure 6.18)	Fairing a curve.....	166
Figure 6.19)	Sample 3-Draw sketches (by Andrew Roberts, MIT CADlab).....	167
Figure 6.20)	Different precisions for a deformable surface rendering grid.....	168
Figure 7.1)	The ShapeWright design paradigm	172
Figure 7.2)	User leverage through parameterization of shape	174

Nomenclature

Bold lower case symbols are used to represent vectors or functions with values which are vectors in R^2 and R^3 space. Symbols which are not bold represent scalar quantities or functions with values which are scalars. Uppercase bold symbols represent matrices or column vectors from matrix equations. These rules hold except where noted otherwise.

Shape definitions

\mathbf{w}	$\mathbf{w} = \mathbf{w}(\mathbf{u})$, shape in 3 space, $\mathbf{w}^T = [x \ y \ z]$
\mathbf{w}^h	The finite element approximation of the shape \mathbf{w}
\mathbf{x}_i	unknown gains in the approximate shape \mathbf{w}^h
ϕ_i	known shape functions of the approximate shape
$\mathbf{w}[i], \mathbf{w}[i,j]$	a particular discrete point in the shape \mathbf{w} in 2 and 3 dimensions
\mathbf{u}	parametric shape vector, either $\mathbf{u} = [u]$ or $\mathbf{u} = [u,v]$ for a curve or surface
\mathbf{a}	region of the parametric plane that is the domain of the deformable shape
\mathbf{x}	rest shape in 3 space $\mathbf{x} = [x \ y \ z]$
\mathbf{y}	deformed shape in 3 space, $\mathbf{y} = \mathbf{x} + \mathbf{w}$
\mathbf{w}_u	The partial of \mathbf{w} with respect to u , $\partial \mathbf{w} / \partial u$
\mathbf{w}_{uu}	The 2nd partial of \mathbf{w} with respect to u , $\partial^2 \mathbf{w} / \partial u^2$

Variational and dynamic geometry

$I(\mathbf{w})$	A minimum principle of \mathbf{w}
E_{curve}	ShapeWright curve energy minimum principle
E_{surface}	ShapeWright surface energy minimum principle
E_{film}	Energy functional for a soap film
E_{plate}	Energy functional for a plate
E_{shell}	Energy functional for a shell structure
\mathbf{f}	forcing vector applied to a curve or surface, $\mathbf{f} = \mathbf{f}(\mathbf{u})$
\mathbf{f}_s	force of a spring
\mathbf{f}_g	force of gravity
\mathbf{f}_p	force of pressure
\mathbf{e}	a deviation from \mathbf{w} in 3 space $\mathbf{e} = [x \ y \ z]$
\mathbf{e}^h	a deviation in the space spanned by the shape functions in Φ
ε	A small arbitrary value scalar weighting deviations from \mathbf{w}
α	The stretch resistance weight for a curve
$\underline{\beta}$	the bending resistance weight for a curve
$\underline{\alpha}$	matrix of stretch resistance weights for a surface
$\underline{\beta}$	matrix of bending resistance weights for a surface
ε_{ij}	strain
σ_{ij}	stress

f	a general minimization functional, $f = f(u)$ in $I(\mathbf{w}) = \int f \, du$
g	a general constraint function, $g = g(u)$
h	a combined minimization function, $h = f - g$
λ	a Lagrange multiplier, $\lambda = \lambda(u)$
λ^h	Ritz approximation to the λ Lagrange multiplier function
ρ	mass density for a curve or surface
μ	damping for a curve or surface

Co-ordinate space and transformation definitions

R^2, R^3	2 and 3 dimension vector spaces
e_1, e_2, e_3	unit orthonormal vectors in R^3
$\Delta u, \Delta v$	Finite difference spacing in the parametric plane
Δt	time step size for dynamic system integration
\mathcal{H}^0	Hilbert 0 space, the set of functions with finite energy
\mathcal{H}^2	Hilbert 2 space, the set of functions with 2nd derivatives of finite energy
\mathcal{H}^4	Hilbert 4 space, the set of functions with 4th derivatives of finite energy
$ \mathbf{w} $	the norm of \mathbf{w} , always a scalar
$ \mathbf{w} _0$	Norm of \mathcal{H}^0 space
$ \mathbf{w} _2$	Norm of \mathcal{H}^2 space
$ \mathbf{w} _4$	Norm of \mathcal{H}^4 space
z	distance in the e_3 direction
\mathbf{R}	Chapter 3, rotation matrix for angle θ
\mathcal{L}	The barycentric mapping matrix for a triangular element
u_1, v_1	1st vertex of a triangular element
u_2, v_2	2nd vertex of a triangular element
u_3, v_3	3rd vertex of a triangular element
L_1, L_2, L_3	coordinates of a barycentric point
\mathbf{t}_{ij}	tangent direction between nodes i and j
\mathbf{n}_{ij}	normal direction between nodes i and j
γ_{ij}	angle of edge between nodes i and j
2Δ	Twice the area of a triangular element
\mathbf{J}_1^{-1}	Jacobian between 1st order barycentric and parametric partial derivatives
\mathbf{J}_2^{-1}	Jacobian between 2nd order barycentric and parametric partial derivatives
$\mathbf{J}\gamma_{ij}$	rotation matrix for angle γ_{ij}
\mathbf{T}_{tn}	transformation matrix from parametric to edge tangent and normal partials

Numerical operators

$a(\mathbf{w}, \mathbf{w})$	The energy inner product associated with a minimum principle, $I(\mathbf{w}) = a(\mathbf{w}, \mathbf{w}) - 2(\mathbf{f}, \mathbf{w})$
∇	the gradient operator in parametric space

∇_L	gradient operator in barycentric coordinates
L_w	a general linear differential operator applied to w
L_c	the Euler differential operator for a ShapeWright curve
L_σ	the Euler differential operator for a ShapeWright surface
(g,h)	vector inner product, $(g,h) = g \cdot h$ when $g,h \in \mathbb{R}^3$, $(g,h) = \int g(u) h(u) du$ when $g,h \in \mathcal{H}^n$

Differential geometry properties

n	the normal vector to a curve or surface
m	vector in the direction of the center of an osculating circle of a curve in a surface
s	curve arc length, often used to parameterize a curve
A	area of a surface
κ	curvature
κ_0	normal curvature
κ_1	maximum principle curvature
κ_2	minimum principle curvature
κ_g	Gaussian curvature
κ_m	mean curvature
G	Chapter 3, the first fundamental matrix of a surface
B	Chapter 3, the second fundamental matrix of a surface
C^0, C^1, C^2, \dots	degree of continuity: continuous function, derivative, 2nd derivative,...

Numerical matrix equations

M	a mass matrix
C	a damping matrix
K	a system stiffness matrix
K_d	a finite difference stiffness matrix
K_σ	a finite element stiffness matrix
K_b	stiffness matrix terms due to bending
K_s	stiffness matrix terms due to stretch
K_λ	stiffness matrix terms due to Lagrange multiplier constraints
C_λ	Jacobian matrix of non-linear constraint equations
X	column vector of all degrees of freedom for a system of equations
\hat{X}	column vector of all combined degrees of freedom for a finite element model
F	column vector of all degree of freedom loads for a system of equations
Λ	column vector of the Lagrange multiplier Ritz approximation weights
A	a system matrix
B	chapter 4 and 5, a system forcing matrix
L, U, D	lower and upper triangular and diagonal matrices used in Gaussian elimination
Y	column vector of the degrees of freedom remaining after constraining a set of system equations

D_0, D_1	components constraint matrices defining the class of linear constraints on the dofs of the system
Γ	diagonal matrix of penalty matrix gains
D_{penalty}	Matrix of linear penalty constraint functions

Finite element stiffness matrix components

a^e	column vector of the dofs of the 9 dof triangle element
$\widehat{a^e}$	column vector of the dofs of the 12 dof triangle element
$\overline{a^e}$	column vector of constraint transformed dofs of the 12 dof triangle element
w_i, w_{ui}, w_{vi}	shape location, and u and v partials at node i
w_{nij}	shape partial normal to edge ij
w_{tij}	shape partial tangent to edge ij
Φ	row vector of all shape function, ϕ_i
Φ_b	column vector of the 2nd parametric derivatives of Φ
Φ_s	column vector of the 1st parametric derivatives of Φ
Φ_λ	row vector of the Lagrange multiplier shape functions
W_b	column vector of the 2nd partial derivatives of w
W_s	column vector of the 1st partial derivatives of w
N^9	row vector of the 9 dof shape functions
N^{12}	row vector of the 12 dof shape functions
$N_{L_1}^{12}$	matrix of the 1st partials with respect to L_i of the 12 dof shape functions
$N_{L_2}^{12}$	matrix of the 2nd partials with respect to L_i of the 12 dof shape functions
$\widehat{N}_{\text{element}i}$	row vector of all combined shape functions for a finite element model
f_i, f_{ij}, e_{ij}	basis functions for the triangle shape functions
Z	matrix of N^9 functions evaluated at the mid-edge locations

1 Introduction

The objective of this work is to develop an improved free-form design methodology capable of interactively defining “natural” shapes with a minimal amount of input from a user. The thesis is that this can be accomplished using “energy-based” deformable models that mimic real surface behavior. The bulk of this investigation was dedicated to developing the mathematical foundations for an implementation of such a system.

Free-form surfaces are shapes that are doubly-curved, smoothly varying and aesthetically pleasing. The need to define free-form surfaces arises in the modeling of molded and stamped shapes in such applications as the design of automobiles, ships, ceramics, air-supported buildings, and general industrial design. The shapes are characterized by loose tolerancing and dimensioning constraints, but strict demands on surface quality and aesthetics. These shapes usually depend more on a designer’s aesthetic judgement than on calculated performance.

The two major applications for mathematically complete descriptions of free-form surfaces have been the lightly scattered data problem and the design of free-form shapes from scratch. The lightly scattered data problem is, given a set of measured surface locations generate an interpolation that completely describes the surface. The free-form design problem is less constrained. Given a means to represent a free-form shape, define one that meets the desires of the user.

The need for mathematically complete descriptions of surfaces comes from the need to compute over the shape of an object. This information is needed for manufacturing to automatically generate NC tool paths in the making of dies and stamps, for computer graphics in rendering an object, and for analysis in the definition of finite element models. This thesis concentrates on the free-form shape design problem but suggests some advances for the lightly scattered data problem as well.

1.1 Computer Aided Geometric Design

Since the 1960s researchers in the Computer Aided Geometric Design field, primarily based on the early work of Coons, Bezier and de Casteljau, have established the utility of parametric surfaces for these applications. A definitive description of this technology

complete with its historical development is given by Farin in his book, "Curves and Surfaces for Computer Aided Geometric Design" [Farin_88].

In a parametric scheme a curve or surface is defined over a simply-connected region in parametric space. The parametric region is mapped to three-space to define the set of Cartesian coordinates that define the free-form curve or surface. Mappings described by polynomial functions have been popular because of ease of computation. Much effort has been devoted to developing polynomial functions that have shape stability and coefficients with geometric interpretations so that the user can modify the surface shape in a straightforward manner. To develop more user control over the free-form shape, piecewise continuous mappings have been used. These enable the user to modify small sections of the shape sequentially until a desired affect is achieved.

The parametric surface community has generated three major schemes for the interactive design of free-form surfaces as represented in Figure 1.1. These are the control point, trans-finite and skinning methods.

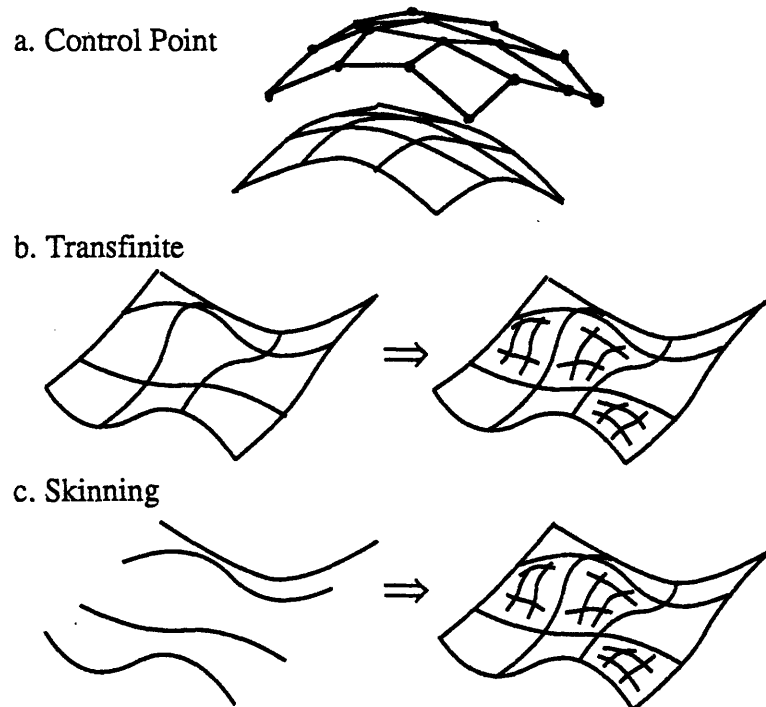


Figure 1.1) Current free-form design paradigms

1.1.a) Control Point Methods

The control point schemes, typified by the Bezier [Bezier_62], B-spline [de Boor_78] and recently the rational B-spline curves [Tiller_83, Piegl_84], are reminiscent of the lightly scattered data problem. The actual surface is defined by the locations of a set of control-points. By moving the control points the shape of the surface can be modified. These schemes do not try to interpolate the control point positions. Instead the control points act to generally define the position of the surface. This compromise is made to increase the user's control of the surface while editing its shape.

One of the nice geometric properties that have been engineered into these shapes is that the surface geometry depends only on the location of the control-points in the immediate neighborhood. This local control property facilitates interactive shape design. This enables the progressive definition of a surface, i.e. completing one section at a time. Modifying the control points of a particular section will not alter the shape of other previously completed sections.

The local control property is a two edged sword. Giving complete interactive control of a surface to the user makes an orderly development of a free-form surface possible but it also makes it very difficult to control the surface's global geometric properties. Modifying a surface while preserving its global convexity requires the user to manipulate several control-points simultaneously. This task has proven to be very difficult. Andersson [Andersson_88] reports that 170 hours were spent in modifying the control point locations on a 700 series Volvo car hood to achieve global geometric convexity and smoothness.

The control point scheme also makes it difficult to enforce geometric constraints since the surface's boundaries and points through which it passes are not explicitly represented. Constraints on such geometric properties require the additional consideration of constraint equations. Many proposals have been made to extend this scheme to the enforcement of geometric constraints [Barsky_80, Barsky_82].

1.1.b) Trans-Finite Methods

The trans-finite schemes define surfaces in the manner of a boundary value or Dirichlet problem. They generate a surface that is completely bounded by a set of arbitrarily shaped curves [Coons_67, Gordon_72]. In general this scheme enables the user to enforce

tangent as well as higher order differential constraints on the boundaries. Control of the interior region of the surface is achieved by dividing the original boundary into a net of curves. The original surface is then represented as a set of patches that join to form the original surface. Each patch can then be treated as its own boundary problem. Each patch is constrained to meet its neighbors to a specified degree of continuity so that the aggregate has acceptable geometric properties. The interior of the surface is modified by changing the shape of an interior curve.

The strength of the trans-finite method is that it directly enforces geometric boundary constraints. Its weakness is that controlling the interior of the surface patch is laborious. For instance, if the desire is to have a generally shaped surface constrained to go through a target point somewhere in its interior then the trans-finite surface has to be divided into a set of patches such that the patch boundaries pass through the target point. The user is now obligated to control the interior curves when originally the user was interested in controlling only a single point location. Once the net has more than a very few curves, controlling global geometric properties will require the manipulation of a large number of curves in the net and will be difficult for a user to accomplish.

1.1.c) Skinning Methods

Like the trans-finite scheme, skinning defines the surface as an interpolation through a set of curves. However, instead of interpolating a net of curves, skinning interpolates a series of curves defined in a set of parallel planes [Tiller_83]. The surface is generated by interpolating the curves through the space between planes. This basic method has been extended to allow non-planar curves with arbitrary orientation [Woodward_88, Coquillart_87] and can be viewed as a special case of generalized cylinders in which the surface is defined by a single curve which can change its shape as it is swept through space along a prescribed trajectory.

When compared to the trans-finite method this technique is a compromise between control and ease of use. The user is freed from specifying about half the geometric boundaries while still retaining a high degree of geometric control. This approach has been very effective at defining ship hull shapes where the geometry is mostly determined as a series of bulkhead profiles. As a general free-form modeling package this scheme still has the same limitations on controlling shape as the trans-finite system. Internal shape is specified

by internal contours and once the number of contours grows large, controlling global geometric properties becomes difficult.

The success of the current parametric surface technology is that it provides a means for developing a wide range of free-form shapes interactively. A major disadvantage of this technology is that the large range of shapes requires the user to control a large amount of information, i.e. the Cartesian locations of the control points. This limitation makes it difficult to build large-scale shape features due to the need to simultaneously coordinate changes to many control points or curves. Implementors of this technology, such as Mori et. al. at Nissan Motor Co. [Mori_86], commonly site the need to "create and modify (free-form) designs on the graphic terminal screen with much greater ease".

1.2 The ShapeWright Design Paradigm

This thesis proposes a new paradigm for the generation of free-form shapes that shares many of the strengths of the existing techniques but avoids their difficulties called the ShapeWright paradigm. The approach is based on the observation that people have a rich background with deformable surfaces. People find it easy to imagine folding a piece of paper or expanding a soap bubble with an internal pressure. Historically, such physical surfaces have commonly been employed as the basis for free-form design. Thin wooden splines were used to loft the shape of ship hulls and aircraft wings. Actual soap films were used to design the roof of the Olympic stadium in Munich, Germany.

The excellent geometric properties of deformable membranes can be exploited to build surfaces that opportunistically seek "fair" or "smooth" shapes while satisfying their geometric constraints thereby freeing the user from this responsibility. Deformable surfaces seek shapes that minimize an internal energy. For a soap film the energy is proportional to its area. For a thin elastic beam the energy of an infinitesimal section is proportional to its curvature. A geometric primitive can be generated with an artificial energy functional designed so that it naturally has desirable properties for geometric design.

The ShapeWright paradigm is a three step procedure for generating objects as shown in Figure 1.2. First, the essence of the object is defined as a set of three dimensional character lines. These lines define "hard" geometric constraints for the shape such as edges and creases. These do not include lines typically used to characterize the interior shape of a

surface such as silhouette lines. Then the object is skinned. Imagine dipping the wire frame object into a bucket of soap and pulling it out slowly so that over every face there is now a deformable soap film like surface. Finally, the object shape is completed by interactively sculpting the surfaces with forces. Modifying the shape can be achieved by changing the character lines or the sculpting forces.

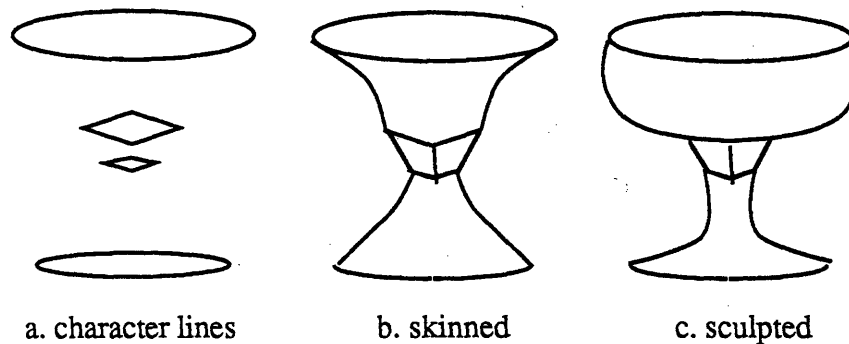


Figure 1.2) The ShapeWright design paradigm

The ShapeWright paradigm has been crafted keeping the philosophy suggested by Pentland [Pentland_86] in mind. He suggests an ideal CAD system should work with shape in a manner paralleling the way people think about shapes. Many researchers have suggested that people are only capable of making qualitative judgements of surface shape, such as the sign of the Gaussian curvature [Stevens_81, Brady_84, Pentland_87]. The ShapeWright paradigm seeks to build surface shape as a combination of explicit and implicit geometry. Character lines are explicit while interior surface region shapes are implicit. Once the specific geometric boundaries of the shape are defined with character lines, the shape of the interior can be modified by the general sculpting operators to achieve global shape characteristics such as general size, convexity and concavity.

1.3 ShapeWright Design Issues

The challenges addressed in this thesis are to select appropriate deformable curve and surface models and to define a solution strategy that supports the ShapeWright modeling paradigm. Additionally, these choices have to be made in such a way that the resulting system can be used interactively. More specifically, this investigation address three major

problem areas that have come to be loosely termed the "continuity", "constraint", and "topology" problems.

1.3.a) The Continuity Problem

The manufacturing processes used to make free-form surfaces require a complete description of a surface in the sense that enough information be available to generate, for example, the NC tool paths of an automatic milling machine. To do this a complete mathematical model yielding the continuous shape and normals of the surface are needed.

Additionally, the kinds of surfaces to be developed by ShapeWright are to be smooth and fair. At a minimum, the final shapes defined need to have continuous tangent planes. As will be seen, the energy algorithm to be used to define the surface naturally distributes curvature over large regions of the surface. As a result, very fair shapes are generated although only C^1 continuity is enforced by the underlying shape representation.

1.3.b) The Constraint Problem

The means to control shape explicitly in the ShapeWright paradigm is to fix a surface to a character line during the skinning step of the process. The system developed must support a set of geometric constraints rich enough to accomplish this skinning step. A set of constraints deemed sufficient for skinning include the "pinned", "hinged" and "fixed" edges, which are represented schematically in Figure 1.3.

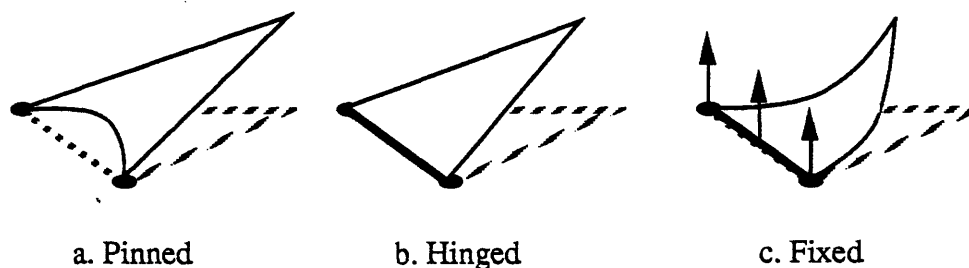


Figure 1.3) Geometric constraints needed for skinning

The "pinned" edge allows any point or set of points in the surface to be set to fixed locations in space. The position of a "hinged" edge is completely specified by the constraint, but the normal vector direction along the edge is free to vary. Hinged edges can

be used to model discontinuities in the interior of the surface. At every such character line, two surfaces may be stitched together with the hinged constraint, to make a C^0 discontinuity in the surface.

The "fixed" edge constraint specifies both the shape and the normal vector direction of a surface along the length of an edge. The effect of the fixed edge is to model a cantilevered beam, rather like a diving board at a pool. The internal surface is free to bend as long as the shape and orientation of the surface along the constrained edge remain fixed.

The fixed edge constraint is needed for local control. Any surface region isolated by a set of edges which are fixed can be deformed independently of the rest of the surface. For example, this feature can be used to add a pocket for a door handle into the shape of a car door. The region of the pocket needs to be identified by adding edges that surround it. Those edges are then fixed to isolate the geometry of the door from the geometry of the pocket. The pocket is then manipulated to make room for the door handle.

1.3.c) The Topology Problem

The range of topological structures that the system can model must be large enough to be of practical use in modeling objects to be built. The character line wire frame models of the ShapeWright paradigm will define the boundaries of a set of faces that will be used to define the final free-form surface. In general, these character line models will have vertices where any number of faces may meet, will have faces which may have any number of boundaries and may even have edges where more than two faces can meet. The topological range for surfaces that was selected as necessary for the ShapeWright paradigm included those shapes which are needed to skin such a character line wire frame model.

1.4 Scope of Work

In this thesis primitives are developed for both curves and surfaces to support the ShapeWright design paradigm. In Chapter 3, the curve and surface energy functionals are given. They consist of two terms: one that depends on a measure of local stretching and the other that depends on a measure of local bending. The term that depends on stretching makes the surface act like a soap film, causing it to seek to minimize its area. This term is very effective at preventing the surfaces from folding like a curtain on a curtain rod while

the bounding geometry is being modified. The term that depends on bending causes the surface to distribute high levels of curvature over large regions. This term causes the surface to deform in a naturally graceful manner.

Shape is achieved by a numerical simulation in time. The deformable shapes continuously seek shapes that minimize their energy while meeting the imposed geometric constraints. The user can apply forces, change the geometric constraints, or modify the weights in the energy functional while the simulation is running to sculpt the final free-form shape.

The choice of the energy functional and the implementation technologies are influenced by the desire to make the system highly interactive. For this reason the actual energy functional used is a quadratic function of the first and second order terms of the surface's local deformations as defined by its Taylor Series expansion resulting in a very well behaved set of linear equations. These equations can be expressed as symmetric, positive-definite, banded matrices and as such lend themselves to very fast solving methods.

In Chapter 4, the actual geometric primitive formulations based on the Ritz finite element method are given. The resulting curves and surfaces are represented parametrically and are suited for applications including the automatic generation of NC tool paths. They have the capacity to model a large topological range of shapes while meeting very general geometric constraints. Due to the energy minimization paradigm these surfaces tend to be very well behaved and enable the interactive sculpting of surfaces.

An implementation based on this scheme, and some of the related implementation issues are discussed in Chapter 6. Chapter 5 contains a discussion of some of the problems encountered in the implementation of geometric constraints. Some of the contributions of this work and problems of the Computer Aided Geometric Design field as well as a review of previous "energy-based" shape applications are given in Chapter 2.

The ShapeWright technique depends on the artful engineering of well-behaved geometric primitives for free-form design that can act to decouple the tasks of overall shape definition and surface fairness. The approach presented here relegates the fairness preservation to the geometric primitives freeing the user to concentrate on shape definition alone.

2 Previous Work

In this chapter the relatively brief history of applying energy minimization algorithms to the generation of shape geometry is presented. Following this section is a much broader description of current and past work related to all forms of free-form geometric shape modeling.

2.1 Precursors to Energy-Based Shape Design

Energy-based shape modeling is the combination of parametrically described geometry with an energy minimization algorithm that gives the geometry natural properties. This supplies a means to control shape behavior during an interactive modeling process that can greatly reduce the amount of input effort required to build or modify shape. The savings in input effort is obtained because the energy minimization algorithm directly controls the simultaneous manipulation of the huge number of degrees of freedom needed to parametrically represent free-form shapes while the user indirectly modifies shape with global operators described as forces, material properties and geometric constraints.

In contrast, the Computer Aided Geometric Design (CAGD) field has typically relied on building the properties of the surface directly into the parametric shape functions and relied on the user to manipulate the resulting degrees of freedom to build shape. The range of shapes that can be modeled by energy-based methods and the more established CAGD approaches are exactly the same. In fact, the energy-based scheme can be applied to any parametric representation.

One exciting aspect of the paradigm is that the energy functional to be minimized can be designed. In this manner it is possible to engineer surfaces with different “natural and desirable” properties. Surfaces which minimize their area and distribute their curvature smoothly have been built in this thesis to create geometry which is opportunistically fair-seeking.

Other researchers in the CAGD community have used energy-based techniques for interpolation and the fairing of existing shapes. Schweikert [Schweikert_66] introduced the notion of splines in tension to improve the interpolating behavior of the spline. Schweikert's splines are an analytic solution to the equation,

$$\frac{d^4 \mathbf{w}}{du^4} - \alpha^2 \frac{d^2 \mathbf{w}}{du^2} = 0 \quad 2.1$$

subject to the constraints of interpolating the data. Shape control of the spline is achieved by modifying the α values. Nielson [Nielson_74] noted that solving this differential problem for shape is equivalent to finding the one shape that minimizes the following integral,

$$I(\mathbf{w}) = \int_{\text{curve}} \left(\frac{d^2 \mathbf{w}}{du^2} \right)^2 + \alpha^2 \left(\frac{d\mathbf{w}}{du} \right)^2 du \quad 2.2$$

and developed a piecewise polynomial interpolant that approximated the minimum while interpolating the constraints. In both cases the final splines were found once the interpolation points and the α values for each span were given by solving a tri-diagonal set of equations.

Energy-based or minimization algorithms have also been used to help select the twist vectors in Coons-type patched surfaces. Several efforts have been published [Nowacki_83, Hagen_87, Kallay_90] that try to improve the surface shape by optimizing the twist vector without changing the position and tangent data of the surfaces. Similar approaches have been used in the energy-based fairing of surfaces [Kjellander_83, Farin_88, Sapidis_90, Lott_88] which are discussed more fully in the section 2.4.d.3. Here the constraint of preserving all the surface point locations and tangents is relaxed in favor of increasing surface fairness.

All of these energy-based twist vector selection and surface fairing techniques have been designed and implemented to be used as shape modification tools. Once the surface has been defined by the user these approaches help to finish the shape. In recent years researchers in the Computer Graphics and Vision area have also begun to consider the generation of deformable models by the use of energy-minimization techniques.

Terzopoulos, Witkin and Kass [Terzopoulos_87,Kass_87] have used deformable models for extracting shapes from video images and simplifying the generation of life-like animations. In this work the deformable model behavior is described by a minimization principle. The related Euler differential equations are solved using finite differences. To extract shape from video images, the video intensity map is treated as a force field that operates on the deformable object. In this manner the deformable objects are encouraged to extract different features of the video image. The resulting set of algebraic equations is stabilized by introducing damping and integrating through time. The introduction of time was exploited to extract sequences of life-like animations.

The deformable models in this early work have many of the elements of an interactive design package. But since the interest of these researchers was solving computer graphics problems, the resulting shape representations, range of topologies, and geometric constraints are inappropriate for CAD applications.

Some efforts have been made to consider the use of energy-based methods as a shape design package. Bloor and Wilson [Bloor_89] define blends as the solution to differential equations defined as boundary value problems. The differential equations are solved using finite difference approximations resulting in a surface model that consists of a set of discrete points. Interestingly, their basic elliptic differential equations can be shown to be equivalent to energy-minimization principles.

Perhaps the work in this thesis was most foreshadowed by the efforts of Antti Pramila [Pramila_78], who saw the opportunity to use finite element methods as a shape generation package for ship hull design. He proposed a variation on the trans-finite surface definition scheme that deforms a plane shape into a ship hull by considering deformations normal to the surface. His surfaces enforce geometric constraints while minimizing a fairness criterion, based on local curvatures, equivalent to linear plate bending. Grieger [Grieger_85] proposed the same idea of capturing shape defined over a set of constrained geometry. Although no implementation was discussed he mentioned that standard existing finite element packages could be used to define surfaces in such a manner.

In this thesis, energy minimization techniques will be used as the foundation for a distinctly new way to define free-form surfaces. The idea is to define surfaces with a minimal amount of information. Like the trans-finite methods of surface design, surfaces will be defined by skinning a set of geometric boundaries. But, the number of these character lines

can be limited to the set of character lines that exist in the actual object. The trans-finite method's need for artificial character lines to define the interior of a surface will be satisfied by sculpting the interpolating surface interactively with the use of applied surface forces.

2.2 The Foundations of Computer Aided Geometric Design

The early work in the Computer Aided Geometric Design (CAGD) field was based on the seminal work of Ferguson, Coons, Bezier and de Casteljau. Prior to these efforts, in the words of S.A. Coons [Coons_67], "the subject of surface mathematics had been investigated in analytical geometry and in differential geometry, from the standpoint of the analysis of geometric properties of surfaces that already exist, but very little literature had been produced on the subject of the creation of such surfaces." Using the tools of these fields, these researchers soon demonstrated the utility of parametric descriptions of curves and surfaces. Subsequent efforts in the CAGD field have concentrated on extending, formalizing, and enhancing the performance of free-form shape interpolation and design.

The original Bezier curve was defined in terms of a recursive algorithm developed by de Casteljau. The control point development of free-form shape was put on a formal basis when the Bezier curve was expressed explicitly as the linear sum of Bernstein polynomial basis functions. With this formalization, determining the many positive attributes of these curves became straightforward.

Spline curves, which are piecewise polynomial curves constrained to enforce a specified degree of parametric continuity at the boundaries, were introduced to support the modeling of very complicated shapes in a controlled interactive manner. Piecewise Bezier splines, or B-splines, popularized by Gordon and Riesenfeld [Gordon_74], differ from Bezier curves in that they exhibit local control and have the ability to model straight line segments without being everywhere linear. Rational B-splines were introduced so that the conic sections could be exactly represented. The history and the technical details of this series of developments are well described in Farin's book, "Curves and Surfaces for Computer Aided Geometric Design" [Farin_88].

Coon's work, which visualized a surface as the trans-finite interpolation of a network of bounding curves, was formalized when the interpolants were seen to be projector operators applied to the bounding constraint geometry [Gordon_72]. Later Barnhill and Gregory

[Gregory_74] augmented the Coons patch so that it could support non-singular cross derivatives at the corner boundaries and allow the interpolation of a boundary of both specified shape and tangent. Recent work of this same flavor continues with Bloor and Wilson who consider surfaces which are defined as the solution to differential equations [Bloor_89]. Interestingly, Gordon in 1972 also recognized the strong relationship between his blending functions and the solution to elliptic boundary value differential equation problems.

In applications the control point scheme and in particular the B-spline and the rational B-spline curves have been the most popular [Wu_77, Barsky_82, Fog_84, Douglas_86, Filip_89, Woodward_88]. This technology has been directly applied to the generation of surfaces with the use of tensor product surfaces. Tensor product surfaces, based on bi-linear interpolation, directly generalize any parametric curve scheme to a parametric surface scheme. Tensor product B-splines exhibit the same properties as do B-spline curves including local control.

2.3 Shape Building and the User Load Problem

In these systems the fundamental approach to refining shape is to take advantage of the B-spline local control property and manipulate the control points one at a time. This enables "rubber banding", where the user drags a control point through space while the shape is dynamically redrawn. A designer with a notion of desirable shape can iterate into a reasonable solution even though the surface may adopt quite unnatural shapes during the process.

The major difficulty with this approach is the large amount of user input required to accurately develop a surface. The local control property, necessary for the sequential building of shape, makes it difficult to build and maintain large scale features that depend on many control points. These features include the surface's basic shape, its global convexity and even its overall fairness. Consider trying to lower the height of a B-spline described dome while preserving its convexity. Every point in the surface must be moved. "Fair" or good looking surfaces require controlling the distribution of curvature throughout the surface. Once again the relationship between every point in the surface must be controlled.

Several approaches have been tried to reduce the user input required. Efforts have been made to improve the exercise of placing control points in three dimensions. At the University of Utah a mechanical "wand" was developed that actually manipulated control points in 3-D [Clark_76].

Several efforts have attempted to simplify the generation of the general shape of the surface. Most commonly, the user will be given a starting shape close to the desired shape which can then be modified into a final configuration one section at a time. For example, CADMUS, produced at the United States Naval Academy [Rogers_83], is an integral part of an automated cell for producing half hull hydrodynamic ship models. To limit the interface burden, the user is given an initial starting ship hull shape to modify.

Another approach for reducing the user load in building the general shape of an object is to generate an initial spline surface from other models. Thus a user can develop a general shape easily with a solid modeler and refine it in detail after it has been converted to a surface model. Beeker [Beeker_86] described an algorithm that smooths polyhedral models by converting them into surface models.

Efforts to simplify the modification of surface shape have concentrated on developing sophisticated operators that modify sets of control points simultaneously. Barr [Barr_84] has shown how to apply transformations to existing Spline surfaces to create tapers and twists. Coquillart [Coquillart_87] describes a technique for offsetting B-Splines that can be used to make a general purpose sweep operator. The user defines a path and a cross section using rational B-Splines and the system calculates an extruded surface shape.

Several systems have been proposed that combine the sweep shape generation paradigm with the control point representation. Initial shape can be generated by interpolating a set of user supplied profiles. The resulting surface can then be modified in the typical control point fashion [Tiller_83, Woodward_88, Rasna_89].

The method proposed in this thesis directly addresses the problem of user load in developing and modifying complicated free-form shapes. ShapeWright uses an energy minimization algorithm to directly manipulate the complete set of control points while the user acts on the surface indirectly with global operators described in terms of forces. So although the surface representation has the local shape property, the user is allowed to work at any size scale. For example, preserving the overall convexity of a dome while

raising or lowering its profile is achieved with a single command without resorting to direct user manipulation of the control points. The user applies a "pressure" to the dome causing it to swell like a balloon.

2.4 Parametric Shape Technical Issues

Some of the very attributes that make parametric surface modelers so powerful also cause many of the difficulties in using these systems. As discussed above, the large flexibility in modeling shape introduces a user load problem. The local control property used to allow the sequential building of shape makes it difficult to build or modify large scale features that depend on large numbers of control points. The piecewise approximation to a curve or surface introduces questions of "smoothness". How smoothly does one patch of the geometry connect to its neighbor? Parametric mappings, used to make the geometry independent of the coordinate system, have problems preserving the shape of a set of interpolated data. Interpolations tend to overshoot the data. Additionally, the domain of the parametric mapping directly affects the topology of shapes that can be supported. A rectangular patch can only model four-sided surfaces. What follows is a quick review of these problems and some suggestions made by researchers.

2.4.a) Shape Preservation

Shape preservation is the goal of data interpolation. The problem with the interpolation schemes that have been developed is that while they interpolate the data, they introduce shapes not necessarily suggested by the data. Higher order polynomial interpolations will often oscillate wildly between data points. Piecewise lower order polynomial schemes can add extra points of inflection, indulge in considerable overshoot and appear to be much too flat in sections. Additionally the schemes are very sensitive to any noise in the data.

Schweikert [Schweikert_66] introduced the notion that by adding a term to an interpolating spline that behaved like the tension in an elastic beam, a set of data could be interpolated in a convex fashion when the tension was made large enough. Nielson [Nielson_74] introduced the v_spline , showing that the spline in tension could be related to a minimal principle and proved that the resulting spline had well behaved geometric properties without enforcing high orders of parametric continuity.

Subsequent work has been spent on introducing a family of such tension interpolants and developing automated means of selecting the tension values. For example, Wever [Wever_88] suggests a non-linear optimization strategy for the automatic selection of the tension value over each piece of the interpolation. This selection is based on minimizing the bending energy of a beam while preserving the C^2 property of the interpolant.

Fletcher and McAllister [Fletcher_87] generalized the tension methods by introducing a single Hermite interpolating space that was general enough to include most of the tensioned interpolants. They analyzed the effectiveness of several tensioned interpolants at preserving the convexity of the original data and stated conditions that would allow the automatic selection of the tension to preserve convex interpolation.

The basic tension idea has been generalized to include any extra degree of freedom in a spline which can be used to control its shape without changing its interpolation or continuity properties. Some such systems include Piegl's [Piegl_87] effort to use rational B-splines for interpolation followed by the interactive manipulation of shape parameters for local shape control, and Harada's [Harada_84] introduction of the segmented Bezier curve with a fullness-control parameter called 'c' that is automatically selected based on estimates of the tangent magnitudes at each of the data points.

2.4.b) Shape Preservation, Surface and Twist

Shape preservation is also a problem for interpolating a set of data with a surface. Woodward [Woodward_88] proposed a typical surface interpolation approach. The data points are fixed onto a rectangular grid and interpolated in each direction with a family of curves. This network of curves is used to select the positions and tangents at each of the data points to be used in a tensor product interpolant scheme. If the curve interpolant used includes some tangent information, the tensor product surface will require the additional specification of the twist vector at each data point. The twist vector, unlike curvature or surface tangent, is not a geometric property of a surface. It is an artifact of the parameterization and as such is not a very intuitive parameter to specify. An excellent survey of the many efforts that have been made to select the twist vector values to improve the shape of the surface is given by Barnhill, Farin, Fayard and Hagen [Barnhill_87].

Optimization methods have been used to set the twist vector. Nowacki, Reese and Walter [Nowacki_83] selected the twists to minimize the quadratic sum of the principle curvatures

over the surface. For small deflections this captures the elastic energy stored in a thin elastic plate. This approach was applied to generalized Coons patches by Hagen and Schulze [Hagen_87]. Kallay and Ravani [Kallay_90] similarly selected twist vectors to minimize a quadratic function of the 2nd derivatives of surface shape.

The shape preserving tension methods, although motivated by the interpolation problem, are very interesting to the shape modeling community because they represent additional means to sculpt and manipulate surfaces. The twist selection problem is particularly important to the shape design community because these parameters are not geometrically invariant and therefore not intuitive.

2.4.c) Smoothness

Designers often desire to make very smooth and "fair" or good looking shapes. The introduction of piecewise parametric patch schemes makes it difficult to preserve smoothness across patch boundaries and curve knots. The most common definition of curve smoothness is the continuity of the parametric mapping and its derivatives, e.g. a C^1 curve is smoother than a C^0 curve.

The degree of surface continuity required between patches will depend on the application of the free-form surface. Finite element analysis often times only requires continuous surfaces, disregarding the continuity of higher order derivatives. Objects to be manufactured usually require much higher levels of smoothness. Automobile surfaces are designed so that the perceived reflection lines will themselves be smooth and well behaved [Kaufmann_87]. Kinematic surfaces used in cams are designed so that the third order derivatives, which determine the continuity of both acceleration and jerk, are continuous [MacCarthy_88].

For parametric surfaces, the easiest means to enforce smoothness between adjacent patches is to require continuity of the parametric partial derivatives. If derivatives up to order n are continuous, the segments or patches are said to meet with C^n , or n th order parametric continuity. One of the strengths of the Bernstein-Bezier curves is that the requirements for high order C^n continuity can be written down directly [Bohm_84]. The popular cubic B-spline is made by applying geometric constraints between curve segments to enforce C^2 continuity.

It has been shown that parametric continuity is sufficient, but not necessary, for geometric smoothness and thus it is believed that enforcing parametric continuity across patch boundaries is too stringent a requirement. Nielson, with his tensioned nu_splines, first established the necessary constraints needed for the geometric continuity of the curve's unit tangent and curvature.

Farin [Farin_82] introduced algorithms for curves and surfaces based on Bernstein polynomials that enforce geometric continuity. Barsky and Bette [Barsky_83] introduced the term geometric continuity to describe curves with continuous unit tangents and curvatures, and surfaces with continuous tangent planes and Dupin indicatrix. Deroose [DeRose_85] extended the definitions to any order of geometric continuity through the application of the chain rule to the parameterization being used. Lee and Ravani [Lee_90] developed a theory for any order of geometric contact independent of parameterization for piecewise parametric curves based on the differential geometry of evolutes.

Geometric continuity and not parametric continuity is now accepted as the true measure of curve and surface smoothness.

2.4.d) Fairness

2.4.d.1) Perceptual Methods of Fairing

In addition to smoothness, designer's often require that surfaces be fair. The smoothness of a surface is defined as the degree of differential continuity defined over the surface. A fair surface is a surface that looks good. Unfortunately, there is currently no single quantitative measure of fairness. Several different schemes of measuring fairness and producing fair surfaces have been proposed. These fairing approaches are incremental. The user first creates an initial surface capturing the general desired shape while meeting the geometric constraints. The fairness of the surface is then improved through small modifications of the original geometry.

The most simple of the fairing schemes are interactive. They depend on giving the user a graphical report of the surface from which the user can judge the fairness of the curve. The user is then free to modify the surface to improve its perceived overall fairness. Although these graphical reports depend directly on different geometric measures of the surface, they do not quantify the concept of fairness.

Beck, Farouki and Hinds [Beck_86] proposed to analyze surfaces to isolate peculiar features through the use of several different display techniques. They claim that the common practice of rendering a surface as a set of isoparametric lines is inadequate for this purpose. Instead they prefer plotting curvature maps, contours of constant height, and high resolution shaded images. The curvature maps are displayed by a color coding of the surface. Colors can be selected to display the Gaussian, mean and principle curvatures. Munchmeyer [Munchmeyer_87] gave an excellent case study of the imperfections in a surface's shape using all of the above mentioned graphical reports. In a similar vein Bonfiglioli [Bonfiglioli_86] suggested that surfaces can be well communicated with the use of silhouettes and gives a very simple algorithm for rendering such curves.

Poeschl [Poeschl_84], based on the practices of automobile surface designers, proposed measuring the fairness of a surface as the fairness of isophotes, lines of equal light intensity when seen under a point light source. Similarly, Kaufmann and Klass [Kaufmann_87] proposed using reflection lines as defined by the reflection in a surface of a set of parallel light sources. They allow the user to modify the reflection curves and automatically regenerate a fairer surface.

Bedi and Vickers [Bedi_89] proposed smoothing a surface interpolation through a set of sparse, irregular, inconsistent data by individually smoothing a set of skeletal lines. Skeletal lines are the profiles used in lofting or skinning a surface. Smoothing a skeletal line is accomplished by interactively eliminating the variation in the first divided differences in the control point locations. New skeletal lines are automatically generated by back substitution from the smoothed first divided difference lines. The final smoothed surface is made by lofting the smoothed skeletal lines.

2.4.d.2) Convexity Based Fairing Methods

Like the developers of the tensioned splines, many researchers feel that fairness problems introduced by splines are due to superfluous inflections in the part's shape. Several efforts have been made to identify and eliminate points of inflections on curves and points of zero Gaussian curvature on surfaces. Isolating and eliminating all of these points results in globally convex shapes.

Hoschek [Hoschek_84,85] proved that the polar image of a curve will have a singularity at every corresponding point of inflection on the original curve. This same measure of inflection can be applied to surfaces to find locations of zero Gaussian curvature. Chang and Feng [Chang_84] developed convexity criteria for Bernstein-Bezier surfaces over triangles.

Andersson et.al. [Andersson_87] have generated an automated system for modifying a general surface into a convex one. To do this, the system moved the surface's control points until the sign of the Gaussian curvature is made constant over the entire surface. The control points were moved in a direction normal to the surface and the range of the motion was limited by a specified tolerance. This approach was cast as a nonlinear problem which they solved using Karmarkar's algorithm for linear programming problems.

Ferguson, Frank and Jones [Ferguson_88] also considered making surfaces convex by solving a nonlinear constrained optimization problem. Their approach was based, not on making the entire surface convex, but in making the projection of each isoparameter line onto a plane convex. They claimed that improving the shape of the isoparameter lines in this fashion improves the surface's machinability and air flow properties. Their idea was implemented as an algorithm that moved control points in a direction normal to the surface. Their shape preserving idea was to limit the sum of the squared travel distances of all the control points that were moved.

2.4.d.3) Energy-based Fairing Methods

In the literature, the alternative to the perceptual-based methods of fairing are the energy-based methods of fairing. The first such proposal was made by Kjellander [Kjellander_83a] who, considering the work in tension by Schweikert and Nielson, argued that a cubic spline is a mathematical model of the draftsmen's wooden spline. He claimed the loss of fairness in a curve was directly related to the amount of internal energy that has been stored in the spline due to deformations. He suggested that curves can be made fairer by reducing the total internal energy of the spline. Different measures of a shape's internal energy lead to slightly different methods of fairing a curve.

Kjellander proposed an interactive smoothing approach in which curvature plots were used to identify the poorly behaved sections of a curve. Any identified points of the interpolation were repositioned to remove discontinuities in the third order parametric

derivatives. This is equivalent to removing a point load from a physical spline and thereby reduces the total stored internal energy. Kjellander [Kjellander_83b] also showed how this approach could be applied to surfaces. Farin [Farin_88, Sapidis_90] used this same technique to fair Bernstein-Bezier curves and surfaces. He automated the selection of the next point to be faired by choosing the point with the largest third order discontinuity.

Lott and Pullin [Lott_88] used energy minimization techniques to improve the fairness of a B-spline tensor product surface. Starting with the energy measure of a thin plate in bending they made several liberal approximating assumptions to reduce the numerical size of the problem. They linearized the energy due to bending, they limited the surface energy integral to a line integral through a surface, they constrained the control points to move in a direction only normal to the surface, and they limited the amount of motion by constraining a linearized measure of the change in surface volume. This technique was then satisfactorily applied to a set of ship hull designs.

The fairness and smoothness issues are very important to the shape design community. All of the above approaches used parameterizations that automatically enforced a specified degree of smoothness, but left large scale features such as convexity and fairness to be controlled by the user or added at a later time. As mentioned before this strategy for fair shape design can place a very large burden on the user. The proposal in this thesis is to develop a set of primitives and algorithms for free-form shape design that automatically enforce a degree of fairness while the user manipulates shape.

2.5 Topology

Many of the efforts to generate free-form surfaces are based on rectangular tensor product parametric patches that combine to give a large general shape. The advantages of such surfaces are that they are mathematically well understood, easy to implement, and are a direct extension of a related curve scheme. However, the range of surface shapes which can be modeled by a rectangular array of parametric patches is greatly limited.

All parametric surface schemes select a region in the parameter space which defines the domain of the mapping. The selection of this shape greatly affects the topological range of objects that can be modeled. Rectangle patches model areas bounded with four curves. They cannot model general n-sided areas without incorporating special singularity

functions. They can model topological cylinders and tori with the use of periodic boundary conditions. They cannot model a simple closed surface without introducing singularities in the parametric mapping.

Early work in the finite element area was spent on developing simple shape functions that could be used to support the finite element models. The shape of choice was the triangle since it could be used to approximate any general shape in the parameter domain.

Researchers attempting to solve the plate bending problems originally sought C^1 triangular elements.

Clough [Clough_65] and Zienkiewicz [Bazeley_65] both introduced refined 9 degree of freedom C^1 triangular interpolations to be used for plate bending. These elements allowed cubic variation in shape of the edges but only linear variation of the tangent normal to the edge direction. Zienkiewicz's method was based on finding the linear combination of a set of well behaved shape functions such that the positions and the tangent slopes at each of the triangle nodes could be set independently. These refined elements tend to produce flat interpolations when used on very curved surfaces and as a result behaved poorly as finite element shape functions.

The work in this thesis, as discussed in chapter 4, extends the approach taken by Zienkiewicz for shape function definition. Zienkiewicz used a refinement procedure so that a triangular element with just 9 degrees of freedom could be made to have shape that was C^1 continuous across adjacent elements. He felt the resulting loss in shape due to refinement was outweighed by the reduced complexity of implementation. In the work of this thesis the refinement restriction on shape proved to be unacceptable for aesthetic reasons. The models were too flat in the center of elements. The C^1 continuous triangular elements developed in this work were made by augmenting the triangular element degree of freedom count from 9 to 12.

Other researchers have proposed shape functions for C^1 continuous triangular elements without using refinement. These differ from the proposed element in that they are significantly more complicated. Efforts by Irons [Irons_69] and Bell [Bell_69] both produced 18 degree of freedom C^1 interpolations that supported quartic deviations in edge shape and full cubic variations in the tangent normal to the edge direction. Iron used the position, the tangents, and the three 2nd parametric derivatives at each of the triangle vertices as degrees of freedom. Bell used only the positions and tangents at the corner

vertices but considered the position, the tangent, and the 2nd parametric derivative in the direction normal to the edge at the three mid-edge points for his degrees of freedom.

Triangular elements have always been important to the FEM community because the range of topological shapes that can be modeled is much larger than can be accomplished with rectangular elements. Any polygonal shape in the plane can be modeled with arbitrarily complex meshes where any number of edges can meet at a node in any pattern of angles.

The triangular patch has also been investigated in the CAGD community if not as well received. When de Casteljau first extended the Bezier curve idea to surfaces he did so using regular triangulations of the plane. Later researchers have continued to investigate the generation of Bernstein-Bezier surfaces over irregular meshes. Catmull and Clark [Catmull_78] published a recursive algorithm for generating B-spline surfaces over arbitrary topological plane meshes that guaranteed C^2 continuity except at a set of extraordinary points limited to the vertices of the final mesh. Farin [Farin_82] developed Bezier triangular patches for regular and irregular triangular meshes of the parametric plane. Farin [Farin_86] exhaustively covered the triangular Bernstein-Bezier surfaces in a special issue of Computer Aided Geometric Design. Whelan [Whelan_86] pointed out that the simplest piecewise polynomial C^2 interpolant will be of order 9 requiring 55 degrees of freedom. He went so far as to develop a refined C^2 Bernstein triangular interpolation that only requires vertex data up through fourth order derivatives.

The trans-finite scheme has been extended to the triangle. Gregory and Charrot [Gregory_80] made a C^1 triangulation that was compatible with a rectangular Coons grid. This work was then extended to support n-sided patches with C^2 continuity. The vertices in this method's meshes are restricted to existing at the simple intersections of the bounding curve mesh. Each vertex is expected to have only four properly aligned edges. Such triangular and n-sided patches are needed to model the rounded corners of boxes and the like. Gregory [Gregory_85] then formalized the trans-finite scheme for triangles by showing how blends could be made over a simplex of any order. Interestingly, after the CAGD community received so many fecund ideas in shape modeling from the original finite element community, the flow of technology is now being reversed. Ho-Le [Ho-Le_89] developed triangular mid-edge finite elements to be used to change the mesh density in a finite element model based on the projector mathematics popularized by Gordon and Gregory.

Jones [Jones_88], by ignoring the complexities of having several edges meet a single vertex, showed how patches with any number of bounding edges can be decomposed into a set of four sided patches. He built C^2 interpolants for the four sided edges based on blending functions. Ball and Storry [Storry_89] similarly treated the problem of filling in patches with more than four sides with a set of rectangles. Instead of considering blending functions they use an extension of the B-spline approach.

Bloor and Wilson [Bloor_89a], pursuing the notion that a surface can be defined as the solution to a differential equation, developed convolution mappings that turned n -sided holes into squares. They then generated the final patch shape by applying the inverse convolution to the solution of the four sided boundary value problem. Their numerical solution of this problem was based on finite-difference schemes and therefore resulted in a surface represented as a discrete set of points.

Herron [Herron_85] pointed out a major limitation of discrete interpolants which involved cross boundary derivatives to form a C^1 surface. Requiring C^1 continuity between patches makes sense only if the patch domains are adjacent in the domain space. This makes it impossible to form C^1 closed surfaces. Herron proposed a G^1 triangle interpolant to be used to make well behaved closed surfaces.

An alternative for modeling shapes described by non-rectangular parametric domains is the trimmed surface patch. Here shape is defined over a rectangular region of the parameter plane and the final boundaries of the shape are made by trimming sections of the domain away. This description of arbitrary domain shape is well suited to the solid modeling community where intersections between objects define new boundaries. An example of a solid modeling scheme depending on trimmed surface patches is described by Casale [Casale_87]. A schematic of the trimmed surface patch concept and how it contrasts from a single continuous mapping of shape is shown in Figure 2.1.

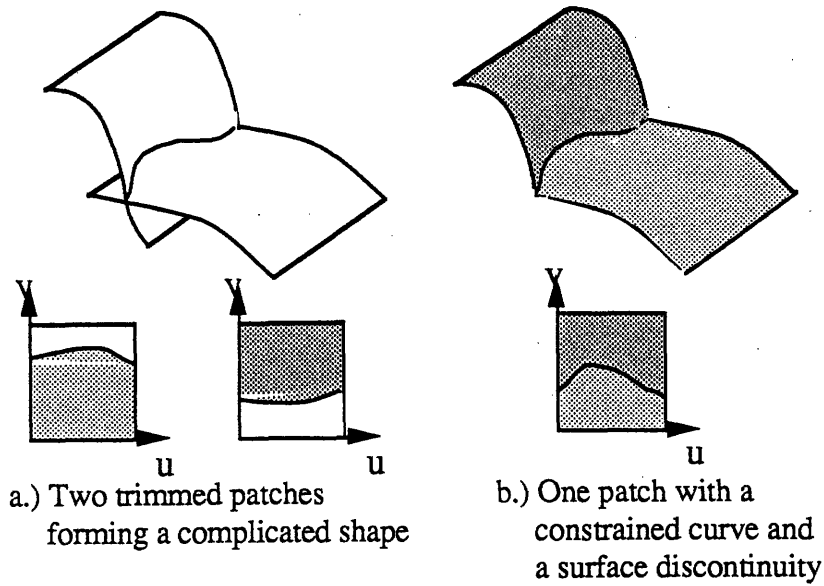


Figure 2.1) Trimmed and constrained surface patches for complicated shapes

The different techniques discussed above help to extend the topological range of the shapes that can be modeled. Using triangular patches enables the modeling of n -sided surfaces. Herron's G^2 triangular patch adds the ability to model simply closed surfaces. The blending functions of Coons and Gordon can be used to generate surfaces over boundaries without corners, in contrast to the triangulated parametric domains which can only approximate such surfaces. In this work the topological range supported by triangular patches is adopted.

3 Deformable Models

This chapter discusses the development of deformable surface and curve models appropriate for the ShapeWright design paradigm. The desire is to define a model which mimics the behavior of real physical surfaces while meeting the needs of the ShapeWright design paradigm.

- o The generation of naturally fair shapes
- o Easy and intuitive manipulation of the resulting faces
- o Interactive computational speeds

In this chapter the energy functionals, which define the properties of the deformable geometry are developed. The notions of time, dynamics and external forces are introduced so that the system enables the interactive manipulation of a deformable model. The physical properties of the deformable models are discussed and compared to those of real physical surface models to emphasize the similarities and differences. The chapter includes a brief review of differential geometry sufficient to define the geometric quantities of stretch, bending and curvature for curves and surfaces. The stretch and bending terms are used to define the energy functionals. The curvature behavior of shape is used to help define measures of fairness.

3.1 Parametric Description of Shape

A parametric description of a curve or surface describes each Cartesian coordinate as an independent function of parametric variables. A curve is described with one parameter as;

$$\mathbf{w} = \mathbf{w}(u) = [x(u), y(u), z(u)]^T \in \mathbb{R}^3 \quad 3.1$$

$$\text{with } \mathbf{u} = [u] \in \mathbb{R}^1$$

and a surface is described with two parameters as;

$$\mathbf{w} = \mathbf{w}(\mathbf{u}) = \mathbf{w}(u, v) = [x(u, v), y(u, v), z(u, v)]^T \in \mathbb{R}^3 \quad 3.2$$

$$\text{with } \mathbf{u} = [u, v] \in \mathbb{R}^2$$

The functions $x(a)$, $y(a)$ and $z(a)$ represent a mapping from the parametric space of $[u]$ or $[u,v]$ to the three-space of $[x,y,z]$. In general the range of $[u,v]$ will be restricted to some subregion of the parametric space. In this paper, that region will be called the parametric shape represented by a . Thus the set of all points in the parametric shape is written as;

$$u \in a \quad 3.3$$

The mapping of the parametric shape into R^3 space will yield the displacement shape. The curves that result by setting either u or v constant in the surface mapping are called isoparametric lines. By giving u or v a set of different values two different sets of isoparametric lines can be generated. Often times these curve families are considered as the basis of a surface, or material coordinate system defined in R^3 space but limited to the set of points in the surface.

3.1.a) Shape in the Local Neighborhood of a Point

Consider parametric functions, $x(u)$, $y(u)$ and $z(u)$ which are differentiable at least twice with respect to the parametric variables. The Taylor series expansion for the shape of the curve in the neighborhood of a point is given as;

$$w(u+du) = w(u) + w_u du + \frac{1}{2} w_{uu} du^2 + \dots \quad 3.4$$

And for a surface the local shape is given as

$$w(u+du) = w(u) + (w_u du + w_v dv) + \frac{1}{2} (w_{uu} du^2 + 2w_{uv} dudv + w_{vv} dv^2) + \dots \quad 3.5$$

$$\text{where } w_u = \frac{\partial w}{\partial u} \text{ etc.}$$

3.1.b) Geometric Properties of the Curve

The first order derivatives completely define the state of stretch for the deformed shape.

The arc length, s , of a curve is defined by the integral relation;

$$s(u) = \int_c \mathbf{w}_u \, du = \int_c \sqrt{x_u^2 + y_u^2 + z_u^2} \, du \quad 3.6$$

The curvature, or state of bending of a shape is defined by the first and second order derivatives. The curvature, κ , of a curve given in terms of parametric variables can be defined as;

$$\kappa = \frac{|\mathbf{w}_u \times \mathbf{w}_{uu}|}{|\mathbf{w}_u|^3} \quad 3.7$$

Curvature and arc-length are differential geometric properties of the curve. They are called differential because they depend on the derivatives found in the Taylor series expansion. They are called geometric because they are properties of the curve independent of the parameterization used to describe the curve.

3.1.c) The Surface Normal

Like a curve, the state of stretch and bending for a surface can be defined in terms of the first and second derivatives of the parametric mapping. Consider, a mapping $\mathbf{w}(u,v) = [x(u,v), y(u,v), z(u,v)]$ which is twice differentiable. When

$$\mathbf{w}_u \times \mathbf{w}_v \neq 0 \quad \text{for all } u \in a \quad 3.8$$

the surface normal is uniquely defined. This also means that nowhere in the parametric shape of the object are the isoparameter lines in R^3 space tangent. Such a parameterization is called regular.

The vectors \mathbf{w}_u and \mathbf{w}_v are both tangent to the surface at the point \mathbf{w} . For a regular mapping the local surface normal at that point will be;

$$\mathbf{n} = \frac{\mathbf{w}_u \times \mathbf{w}_v}{|\mathbf{w}_u \times \mathbf{w}_v|} \quad 3.9$$

The relationship of these vectors is shown in Figure 3.1. For a regular mapping the vectors \mathbf{w}_u and \mathbf{w}_v span the tangent plane to the surface. These vectors, or some appropriate linear combinations of them can be used as a coordinate system for the tangent plane.

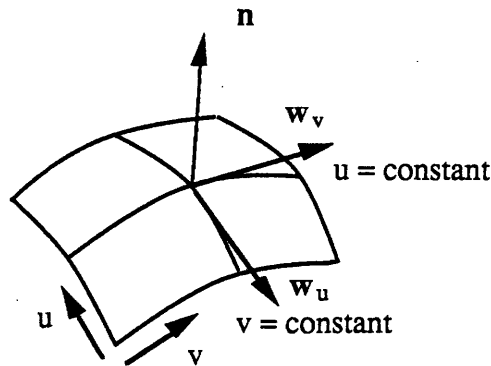


Figure 3.1) The parametric tangents and normal to a regular surface

3.1.d) Arc Lengths and The First Fundamental Form of a Surface

The state of stretch of a surface can be viewed by considering the arc length of a curve restricted to lie in the displaced surface. Such a curve can always be described by a curve in the parameter space which is then mapped to \mathbb{R}^3 space, as shown in Figure 3.2.

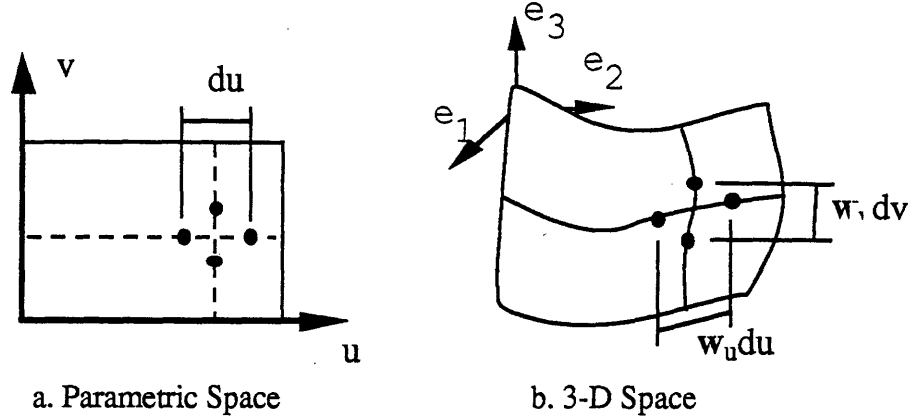


Figure 3.2) Curves in the parametric plane map to curves fixed in the surface

A curve in R^2 and its associated R^3 curve will be given as

$$\mathbf{u} = \mathbf{u}(r) = [u(r), v(r)] \text{ and} \quad 3.10$$

$$\mathbf{w}(r) = [x(\mathbf{u}(r)), y(\mathbf{u}(r)), z(\mathbf{u}(r))] \quad 3.11$$

The arc length squared of this curve, will be;

$$ds^2 = |\mathbf{w}_r|^2 dr^2 = \begin{bmatrix} u_r & v_r \end{bmatrix} \begin{bmatrix} \mathbf{w}_u \cdot \mathbf{w}_u & \mathbf{w}_u \cdot \mathbf{w}_v \\ \mathbf{w}_v \cdot \mathbf{w}_u & \mathbf{w}_v \cdot \mathbf{w}_v \end{bmatrix} \begin{bmatrix} u_r \\ v_r \end{bmatrix} dr^2 = \begin{bmatrix} du & dv \end{bmatrix} \mathbf{G} \begin{bmatrix} du \\ dv \end{bmatrix} \quad 3.12$$

This equation is known as the first fundamental form and defines the matrix \mathbf{G} . The matrix \mathbf{G} , a function of \mathbf{w}_u and \mathbf{w}_v only, completely determines the state of stretch for the parametric surface. This fact can be used to help calculate the area of the displaced shape by taking an integral in the parametric domain.

The area of a small element in the displacement shape for a regular mapping will be given by;

$$\begin{aligned} dA &= |\mathbf{w}_u du \times \mathbf{w}_v dv| = \sqrt{\mathbf{w}_u^2 \mathbf{w}_v^2 - (\mathbf{w}_u \cdot \mathbf{w}_v)^2} du dv \\ &= \sqrt{\det(\mathbf{G})} du dv \end{aligned} \quad 3.13$$

and the total area can be found from the first fundamental form by;

$$A = \int_{\sigma} \sqrt{\det(\mathbf{G})} \, du dv \quad 3.14$$

3.1.e) Arc Curvature and The Second Fundamental Form

Now consider the curvature of a curve which is restricted to lie in the surface. A curve's curvature is defined as the magnitude of the 2nd derivative of a curve with respect to arc length.

$$\kappa \mathbf{m} = \mathbf{w}_{ss} \quad 3.15$$

where \mathbf{m} = unit vector in the direction of the center of the osculating circle and
 κ = curvature

applying the chain rule to the parametric definition of the curve yields;

$$\kappa \mathbf{m} = \mathbf{w}_{uu}u_f^2 + 2\mathbf{w}_{uv}u_f v_f + \mathbf{w}_{vv}v_f^2 + \mathbf{w}_u u_{ff} + \mathbf{w}_v v_{ff} \quad 3.16$$

Notice that although the surface normal will be orthogonal to the surface tangents \mathbf{w}_u and \mathbf{w}_v , it will not generally be aligned with the curve's normal vector, \mathbf{m} . Taking the inner product of $\kappa \mathbf{m}$ with the surface normal \mathbf{n} , using $\mathbf{n} \cdot \mathbf{w}_u = \mathbf{n} \cdot \mathbf{w}_v = 0$, yields;

$$\kappa \mathbf{m} \cdot \mathbf{n} = [u_f \ v_f] \begin{bmatrix} \mathbf{n} \cdot \mathbf{w}_{uu} & \mathbf{n} \cdot \mathbf{w}_{uv} \\ \mathbf{n} \cdot \mathbf{w}_{vu} & \mathbf{n} \cdot \mathbf{w}_{vv} \end{bmatrix} \begin{bmatrix} u_f \\ v_f \end{bmatrix} \quad 3.17$$

which implies;

$$\kappa \mathbf{m} \cdot \mathbf{n} \, ds^2 = [du \ dv] \begin{bmatrix} \mathbf{n} \cdot \mathbf{w}_{uu} & \mathbf{n} \cdot \mathbf{w}_{uv} \\ \mathbf{n} \cdot \mathbf{w}_{vu} & \mathbf{n} \cdot \mathbf{w}_{vv} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix} = [du \ dv] \mathbf{B} \begin{bmatrix} du \\ dv \end{bmatrix} \quad 3.18$$

The vector relationships of the curvature properties of a curve lying in a parametric surface are shown in Figure 3.3. The above relation defining the matrix \mathbf{B} , a function of the first and second derivatives of the surface is called the second fundamental form of the surface.

Together with the first fundamental form, G , (needed to compute ds^2) the second fundamental form can yield the curvature for any curve in the surface with some given curve normal direction, \mathbf{m} .

The shape of a surface is completely defined by the \mathbf{B} and \mathbf{G} matrices. Any two parametric mappings that have \mathbf{B} and \mathbf{G} everywhere identical over the domain, \mathbf{a} , will have the same shape up to rigid body displacements.

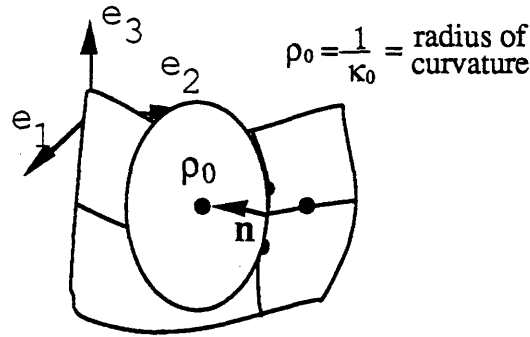


Figure 3.3) The curvature and normal for a curve fixed in a surface

3.1.f) Surface Curvatures

The normal curvature of a curve constrained to lie in the surface is given for the special direction where \mathbf{m} and \mathbf{n} are aligned and $\mathbf{m} \cdot \mathbf{n} = 1$, yielding;

$$\kappa_0 = \frac{[\mathbf{du} \ \mathbf{dv}] \ \mathbf{B} \ [\mathbf{du} \ \mathbf{dv}]^T}{[\mathbf{du} \ \mathbf{dv}] \ \mathbf{G} \ [\mathbf{du} \ \mathbf{dv}]^T} \quad 3.19$$

As the curve direction in the parametric space defined by the vector, $\mathbf{du} = [\mathbf{du} \ \mathbf{dv}]$ is varied, the magnitude of κ_0 will change. The maximum and minimum values of κ_0 are called the directions of principal curvature, κ_1 and κ_2 .

$$\begin{aligned} \kappa_1 &= \max(\kappa_0(\mathbf{du})) \\ \kappa_2 &= \min(\kappa_0(\mathbf{du})) \end{aligned} \quad 3.20$$

The principal curvatures completely define the state of curvature at a point on a surface. The normal curvature of any curve lying in the surface can be found as a function κ_1 , κ_2 and the angle between the curve and the direction of principal curvature, Ψ .

The state of surface curvature can be described alternatively in terms of Gaussian and mean curvatures. Gaussian curvature, κ_g , is defined as the product of κ_1 and κ_2 .

$$\kappa_g = \kappa_1 \kappa_2 = \frac{\det(\mathbf{B})}{\det(\mathbf{G})} \quad 3.21$$

The mean curvature is defined by $\kappa_m = 1/2 (\kappa_1 + \kappa_2)$ and is calculated as;

$$\kappa_m = 1/2 (\kappa_1 + \kappa_2) = \frac{(\mathbf{n} \cdot \mathbf{w}_{vv})(\mathbf{w}_u \cdot \mathbf{w}_u) - 2(\mathbf{n} \cdot \mathbf{w}_{uv})(\mathbf{w}_u \cdot \mathbf{w}_v) + (\mathbf{n} \cdot \mathbf{w}_{uu})(\mathbf{w}_v \cdot \mathbf{w}_v)}{\det(\mathbf{G})} \quad 3.22$$

And conversely the principal curvatures can be calculated from the Gaussian and mean curvatures as;

$$\begin{aligned} \kappa_1 &= \kappa_m + \sqrt{\kappa_m^2 - \kappa_g} \\ \kappa_2 &= \kappa_m - \sqrt{\kappa_m^2 - \kappa_g} \end{aligned} \quad 3.23$$

Positive Gaussian curvature indicates that both the directions of principal curvature are of the same sign. This happens for elliptic surfaces like spheres. A negative Gaussian curvature is indicative of an hyperbolic surface such as a saddle point where the surface is convex in one direction and concave in another. When the Gaussian curvature is zero the surface shape is called parabolic and one of the surface's principal curvatures has become zero. This is typical of cylinders and ruled surfaces where the surface is only singly curved. Surfaces of zero Gaussian curvature are called developable. They can always be transformed into a plane without stretching the surface, just unrolling it. Between every elliptic and hyperbolic region on the surface will be a set of parabolic points. Gaussian curvature is calculated as;

3.1.g) Measures of Surface Fairness

A surface is said to be fair when its shape is pleasing to the eye. Unfortunately, although many surfaces designed are required to be fair, there is no quantitative measure of fairness. It is therefore difficult to build a surface numerically that guarantees surface fairness.

Many efforts have been made to help quantify the fairness concept, or at least physically describe the attributes of a fair surface. For example, Farin and Sapidis, [Farin_88] have proposed the following definition of fairness,

"A curve is characterized as fair if the corresponding curvature plot is continuous and it is as close as possible to a piecewise monotone function with as few as possible monotone pieces."

Accordingly, fair surfaces tend to minimize the number of inflection points, and avoid rapid changes in curvature. Most suggested measures of fairness are based on some property of the curvature of the surface. Automatic control of the fairness of the curve may be achieved by controlling the curvature properties. So, reporting the curvature properties of the surface to the user will be a major means of verifying its fairness. In this work, curvature reports of a surface are made by color coding a surface based on its Gaussian, mean, maximum or minimum principal curvatures. These reports tend to identify regions of poor behavior by highlighting regions of discontinuity and rapid change in curvature.

Whatever the actual attributes of a fair surface, people have very little problem in perceiving fair surfaces. As such, one of the most effective things an interactive design system can do is to present the surface graphically to the user in such a way as to enhance its perception. In this work surfaces can be viewed as shaded light models, sets of contour lines, and sets of isoparameter lines.

We have found that the shaded light models combined with contour lines give the best impression of shape as long as the user can dynamically manipulate the orientation of the surface with respect to the light source and the viewing position. Contour lines give a superior understanding of the surface than do isoparameter lines which are seemingly presumed to be contour lines and therefore tend to mislead surface shape interpretation.

The system built also has the capacity to render the lighted surfaces in 3-d using a Stereo Graphics rendering system. Although the images generated are very dramatic, such renderings don't seem to add much information to the perception of surface fairness.

3.2 Modeling of an Elastically Deformable Surface

The main premise of the ShapeWright paradigm is that naturally fair surfaces can be easily defined in a medium that mimics real deformable surfaces. To model an elastically deformable surface we consider its strain energy expressed as a functional. In the words of Timoshenko, [Timoshenko_34], "Under the action of external forces an elastic body undergoes a deformation during which the forces do a certain amount of work. This work is transformed into strain energy, or the potential energy of deformation of the body."

The equilibrium configuration of a deformable body subject to constraints and forces can be found using the principle of virtual work. This principle states that the work done by the applied forces for any kinematically permissible variation about the equilibrium configuration will be zero. This statement is equivalent to defining the equilibrium configuration of the system as the one shape out of all possible shapes which minimizes the energy functional.

We seek to define the properties of the deformable models by defining their energy functionals. Actual equilibrium shapes will be found at a later time by satisfying the requirements of the virtual work principle. The energy functionals actually used in ShapeWright were motivated by the elastic behavior of soap films, thin plates, and shell elements with the additional desire for interactive solution times.

Soap films act to minimize their area. A soap film will never fold in on itself as a curtain might when pushed along a curtain rod. Area minimization is a desirable attribute for a free-form shape modeler. As the bounding character lines are modified, the user should not have to be concerned about having the surface fold, exhibit cusp-type singularities, or behave in a generally poor fashion all in an attempt to preserve its initial area. We therefore seek primitives that tend to minimize their areas. This will be done by including terms in the energy functional that add energy for increased areas.

Thin beams and plates such as the wooden spline deform by bending. The energy of deformation is proportional to the bending of the beam as measured by its local curvature. Wooden splines have been used for shape design because they produce very fair curves. One desire for the ShapeWright primitives is that they naturally produce fair shapes as the user manipulates them for design purposes. Terms that tend to weight the bending of the shape will be added to the energy functional to improve the fairness of the surface.

3.3 The Energy Functionals

The energy of deformation for a ShapeWright primitive consists of the sum of two independent terms; the energy due to stretch plus the energy due to bending. The energy functionals for these primitives have the form;

$$E_{\text{deformation}} = \int_{\sigma} (\alpha \text{ stretch} + \beta \text{ bending}) d\sigma \quad 3.24$$

The α and β terms act to weight the impact of stretching and bending on the overall energy of the shape. Larger α values increase the amount of energy due to stretch while β weights the energy due to bending. Changing the values of α and β will change the deformable surface's shape accordingly. Increasing the α values causes the surface to increasingly resist stretch and to seek solutions of smaller and smaller areas. Larger β values will cause the surface to distribute its curvature over large regions acting to make it seek flatter and flatter shapes.

The energy functional used for a ShapeWright curve is given by:

$$E_{\text{curve}} = \int_{\text{curve}} (\alpha(u) w_u^2 + \beta(u) w_{uu}^2 - 2 f w) du \quad 3.25$$

The functional for a ShapeWright surface is;

$$E_{\text{surface}} = \int_{\sigma} \left[\left(\alpha_{11} w_u^2 + 2\alpha_{12} w_u w_v + \alpha_{22} w_v^2 \right) + \left(\beta_{11} w_{uu}^2 + 2\beta_{12} w_{uv}^2 + \beta_{22} w_{vv}^2 \right) \right] - 2 \mathbf{f} \cdot \mathbf{w} \, dudv \quad 3.26$$

where $\mathbf{f} = \mathbf{f}(\mathbf{w}, t)$ and denotes the applied forces used to sculpt the surface.

The curve energy functional is very nearly equivalent to the energy functional used as the basis of Schweikert's interpolating v-splines, [Schweikert_66, Nielson_74] exclusive of the forcing terms. The surface energy functional was originally published by Terzopoulos et.al., [Terzopoulos_87] in a paper which defined surfaces with finite differences defined over the unit square.

3.4 Properties of the Energy-Based Deformable Models

The energy functionals completely characterize the behavior of the deformable primitive. In this section some of the properties of the above functionals that make these models interesting for shape modeling will be discussed.

3.4.a) Shape Definition

A major difference between these models and the theory of elasticity is the definition of shape. In these models deformable shape is defined as;

$$\text{Deformable Shape} = \mathbf{w}(u, v) \quad 3.27$$

While elasticity theory defines shape as;

$$\text{Elastic Shape} = \mathbf{x}(u, v) + \mathbf{w}(u, v) \quad 3.28$$

where $\mathbf{x}(u, v)$ is the rest shape of the material.

In elasticity theory the unloaded body has a minimum of strain energy when it is in its rest shape $\mathbf{x}(u, v)$. In this model of deformable shape there is no rest shape. Strain energy

continually decreases as the bending and stretching terms go to zero. An unconstrained, unforced deformable shape will become flat to reduce bending energies and reduce its area to zero to reduce stretch energy. Non-point shapes can only be made with some combination of constraints and forces.

3.4.b) Stretch Resistance, α

The α variable directly weights the amount of energy due to stretch. In a curve, the α weighted term is identical to the arc length. Stretch is a local measure of the overall length of a curve and α weights the resistance to it. A curve with a large α value is said to be stiff.

The relationship between α and a surface is not much different. The terms of the surface energy functional weighted by α are just the sum of the terms in the first fundamental form matrix, G . A large value of α in the energy functional will make the surface deform to minimize the terms of the G matrix and thereby weight the resistance to stretch of any curve forced to lie in the surface.

The α variable also acts to minimize the area of a surface, just as the curve's α variable acts to minimize the overall length of the curve. As mentioned previously the total area of a surface can be found as an integral of the determinant of G ;

$$A = \int_{\sigma} \sqrt{\mathbf{w}_u^2 \mathbf{w}_v^2 - (\mathbf{w}_u \cdot \mathbf{w}_v)^2} \, du dv \quad 3.29$$

Although the α weighted term does not directly weight the area of the surface, it does act to minimize the terms of G which tends to minimize the area of the surface.

Because G is a second order tensor, non-isotropic effects can be introduced by weighting the terms of the G matrix independently. The α_{11} term resists the stretch of fibers in the surface in the direction of the u isoparameter line. The α_{22} term resists stretch for fibers in the direction of the v isoparameter line. By adjusting the α_{11} and α_{22} terms independently a surface can be made that is very stiff in one direction and not in the other. The α_{12} term can be used to rotate the material non-isometric axis. Consider a pair of orthogonal

directions in the uv parameter plane that are rotated from the u and v directions by some angle θ . These directions are related by;

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} \quad 3.30$$

Now consider the shape function $\mathbf{w} = \mathbf{w}(u(\tilde{u}, \tilde{v}), v(\tilde{u}, \tilde{v}))$ and its derivatives found with the chain rule as;

$$\begin{bmatrix} \mathbf{w}_{\tilde{u}} \\ \mathbf{w}_{\tilde{v}} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} \mathbf{w}_u \\ \mathbf{w}_v \end{bmatrix} \quad 3.31$$

The α weighted energy functional can be written in terms of the rotated directions;

$$E_\alpha = \int_\sigma \begin{bmatrix} \mathbf{w}_{\tilde{u}} & \mathbf{w}_{\tilde{v}} \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{\tilde{u}} \\ \mathbf{w}_{\tilde{v}} \end{bmatrix} dudv \quad 3.32$$

Substituting the relations between the rotated and original directional partial differentials into the α weighted energy functional yields;

$$E_\alpha = \int_\sigma \begin{bmatrix} \mathbf{w}_u & \mathbf{w}_v \end{bmatrix} \mathbf{R} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{bmatrix} \mathbf{R}^T \begin{bmatrix} \mathbf{w}_u \\ \mathbf{w}_v \end{bmatrix} dudv \quad 3.33$$

So changing the weights of the α matrix so that;

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \alpha_{11} & 0 \\ 0 & \alpha_{22} \end{bmatrix} \mathbf{R}^T \quad 3.34$$

Is equivalent to weighting the stretch of fibers aligned in the \tilde{u} and \tilde{v} directions.

3.4.c) Bending Resistance, β

It would be conveniently intuitive if the β weighted terms were identical with curvature. However, the parametric representation of curvature is a rational non-linear function of \mathbf{w}_u and \mathbf{w}_{uu} . Minimizing this actual quantity would be computationally prohibitive.

Fortunately, the β variable does weight the terms that dominate the actual curvature for

small deflections. This can be seen by considering the Taylor series expansion for the curve and surface in the neighborhood of a point as;

$$w(u+du) = w(u) + w_u du + \frac{1}{2} w_{uu} du^2 + \dots \quad 3.35$$

And for a surface the local shape is given as

$$w(u+du) = w(u) + (w_u du + w_v dv) + \frac{1}{2} (w_{uu} du^2 + 2w_{uv} dudv + w_{vv} dv^2) + \dots \quad 3.36$$

The β term weights the second order behavior of the curve and surface and in this way has the effect of increasing the energy due to bending. Larger and larger curvatures will continue to increase the energy although the linear approximation to curvature will no longer be valid. Additionally, the bending energy weighted by β can only be zero when the surface is flat. So although the β variable does not actually weight a curvature measure it does act to resist surface bending.

3.4.d) Quadratic and Adjoint Energy Functionals

All the internal terms of the energy functional have been constrained to be quadratic and adjoint. Linear variational problems are posed in terms of quadratic functionals, $I(w)$. In our case we are minimizing a potential energy that can be written in the form,

$$I(w) = a(w, w) - 2(f, w) \quad 3.37$$

where $a(w, w)$ is the energy and consist of the quadratic terms of the functional.

The notation (g, h) indicates the inner product between two vectors. If g and h are three space vectors the inner product is the familiar vector dot product. When g and h are continuous functions, such that $g = g(u)$ and $h = h(u)$, the inner product is given by;

$$(h(u), g(u)) = \int_{\text{range of } u} h(u)g(u) du \quad 3.38$$

The second-degree term is the strain energy, and it is associated with an energy inner product,

$$a(\mathbf{w}, \mathbf{g}) = 1/4(a(\mathbf{w}+\mathbf{g}, \mathbf{w}+\mathbf{g}) - a(\mathbf{w}-\mathbf{g}, \mathbf{w}-\mathbf{g})) \quad 3.39$$

The energy inner product is symmetric $a(\mathbf{w}, \mathbf{g}) = a(\mathbf{g}, \mathbf{w})$ the associated energy functional is adjoint. The minimum of an adjoint energy functional can be approximated numerically by the Ritz method resulting in a symmetric matrix equation problem [Strang_73]. Limiting the functional to quadratic terms guarantees that those system matrix equations will be positive definite. The combination of positive definiteness and symmetry in the matrix equation ensures the very rapid solution these problems for shape which will help in supporting interactivity.

3.4.e) Decoupled X,Y,Z Equations

The energy functionals have been additionally constrained so that the related Euler equations will be independent in x,y, and z. This was done so that the system equations need to be built only once, then used for all three directions, reducing the problem size by a factor of three. Since the algorithms which are applied to solving the final matrix equations tend to run $O(n^2)$, where n is the number of degrees of freedom for the problem. This results in an order of magnitude decrease in solution times. The limitation is that constraints which require coupling in the x, y and z directions will not be able to be enforced directly.

The effect on the form of the energy functional is to exclude terms which combine the coordinate functions such as $(x_u(u,v) * y_u(u,v))$. This limitation is automatically satisfied when the energy functional is a function of the inner product of the shape vector and its derivatives. For instance;

$$\mathbf{w}_u^2 = \mathbf{w}_u \cdot \mathbf{w}_u = (x_u^2 + y_u^2 + z_u^2) \quad 3.40$$

The resulting Euler equations are given in section 3.5 and are seen to be independent in the x, y, and z directions.

3.4.f) Elliptic Energy Functionals are Fair Seeking

An additional insight into the fair shape generation property of the ShapeWright primitives can be found in the theory of elliptical differential equations. A detailed discussion of this theory will not be developed here but is discussed by Strang, [Strang_73]. At this time, we are only interested in the implications of this theory for determining the properties of the energy functionals for the ShapeWright primitives.

The theory of elliptical differential equations states that if the boundary conditions and the data, f , are smooth then the resulting solution, w , will also be smooth. Taking advantage of this property the energy functionals have further been constrained so that the resulting energy functionals will be elliptical.

The solvability of the fundamental variational equation, $I(w) = a(w, w) - 2(f, w)$, is guaranteed if the form is elliptic, where the energy of the system is equivalent to the norm of the function in Hilbert space (see section 4.3). For our curve primitives,

$$\begin{aligned} a(w, w) &= \int_C (\alpha w_u^2 + \beta w_{uu}^2) du \quad \text{and} \\ (f, w) &= \int_C f \cdot w \, du \end{aligned} \tag{3.41}$$

A space is a set of functions that all have finite size as measured by the norm of that space. A common norm used to measure the magnitudes of functions in Hilbert space is called the energy norm. Different Hilbert spaces can be defined by including different number of derivatives in the energy norm. For the Hilbert space of all functions which have finite energy through their second derivatives the norm is given by;

$$\begin{aligned} \|w\|_2^2 &= \int_C (w_{uu}^2 + w_u^2 + w^2) du \quad \text{and for surfaces} \\ \|w\|_2^2 &= \int_\sigma (w_{uu}^2 + w_{uv}^2 + w_{vv}^2 + w_u^2 + w_v^2 + w^2) du \end{aligned} \tag{3.42}$$

For the ShapeWright energy to be equivalent to the norm of the Hilbert space it must satisfy the relation;

$$a(\mathbf{w}, \mathbf{w}) \leq k \|\mathbf{w}\|_2^2 \quad 3.43$$

where k is some constant.

Comparing the ShapeWright energy functionals with the Hilbert space norms shows that the ShapeWright energy functionals are just partial weighted norms of their respective Hilbert spaces. They lack the term that adds energy for the position of the body. What this means is that as long as a boundary condition is applied to the ShapeWright surface that fixes its position in space a solution is guaranteed that is equivalent to the solution of an elliptical value boundary problem. And consequently, the guaranteed smoothness of an elliptical boundary value problem solution will be exhibited by the ShapeWright geometry.

3.4.g) No Poisson's Effects

The ShapeWright deformable models can be compared to models of simple plane strain and simple bending. The ShapeWright deformable models can be forced to act without Poisson's effects by enforcing the following constraints on the terms of the α and β matrices.

$$\alpha_{11} = \alpha_{22} = \alpha_{12} \quad \text{and} \quad \beta_{11} = \beta_{22} = 1/2\beta_{12} \quad 3.44$$

Poisson type effects were eliminated to enhance the user intuition in manipulating the surfaces. The user should be able to stretch a general shape without drastically modifying the dimensions orthogonal to the stretch.

3.5 Minimize a Functional or Solve a Differential Equation

The ShapeWright models of deformation have been expressed in the intuitively simple form of an energy functional. These surfaces have greater strain energies when they are stretched or bent. In this section the equivalence between minimizing a quadratic energy functional and solving a related linear differential equation, known as the Euler equation, will be shown.

We seek to minimize the energy functional,

$$I(\mathbf{w}) = a(\mathbf{w}, \mathbf{w}) - 2(\mathbf{f}, \mathbf{w}) \quad 3.45$$

Subject to certain geometric constraints. We assume that the actual minimum is \mathbf{w} , still an unknown at this time. Then any other function which meets the boundary conditions of the problem can be written as

$$\tilde{\mathbf{w}} = \mathbf{w} + \epsilon \mathbf{e} \quad 3.46$$

where ϵ is a scalar of arbitrary sign and \mathbf{e} is a function that when added to \mathbf{w} does not violate the geometric boundary conditions.

Since $I(\mathbf{w})$ is a minimum it must be equal to or less than $I(\mathbf{w} + \epsilon \mathbf{e})$,

$$I(\mathbf{w}) \leq I(\mathbf{w} + \epsilon \mathbf{e}) = I(\mathbf{w}) + 2\epsilon(a(\mathbf{w}, \mathbf{e}) - (\mathbf{f}, \mathbf{e})) + \epsilon^2 a(\mathbf{e}, \mathbf{e}) \quad 3.47$$

The last term weights $a(\mathbf{e}, \mathbf{e})$ with ϵ^2 . The energy, $a(\mathbf{e}, \mathbf{e})$, will always be positive as will ϵ^2 and the addition of the last term will not violate the inequality. However, the second term is weighted by an arbitrarily signed ϵ . To preserve the inequality the relation

$$a(\mathbf{w}, \mathbf{e}) = (\mathbf{f}, \mathbf{e}) \quad 3.48$$

must be preserved for any function \mathbf{e} . Integration by parts will change the symmetric $a(\mathbf{w}, \mathbf{e})$ into an inner product between a general differential operator acting on \mathbf{w} and the function \mathbf{e} with some additional boundary conditions as;

$$(L\mathbf{w}, \mathbf{e}) = (\mathbf{f}, \mathbf{e}) \quad 3.49$$

This relation is known as the weak form of the variation principle. The notation $L\mathbf{w}$ stands for any differential operator applied to the function \mathbf{w} . As an example, the Laplacian

operator ∇^2 is defined as $\nabla^2 = \frac{\partial}{\partial u} + \frac{\partial}{\partial v}$. If $L = \nabla^2$ then $L\mathbf{w} = \nabla^2 \mathbf{w} = \frac{\partial \mathbf{w}}{\partial u} + \frac{\partial \mathbf{w}}{\partial v}$.

The weak form can be rewritten as;

$$(L\mathbf{w}, \mathbf{e}) = (\mathbf{f}, \mathbf{e}) \Rightarrow (L\mathbf{w}, \mathbf{e}) - (\mathbf{f}, \mathbf{e}) = ((L\mathbf{w} - \mathbf{f}), \mathbf{e}) = 0 \quad 3.50$$

The weak form must hold for all functions \mathbf{e} . This will only be true when;

$$L\mathbf{w} = \mathbf{f} \quad 3.51$$

These are the Euler equations also known as the strong form of the original variational principle. A physical interpretation of the above equation is that the external forces, \mathbf{f} , are balanced against the internal forces represented by $L\mathbf{w}$. This is an equation of equilibrium.

Actual execution of the integration by parts steps yields the following syntactical relation for deriving the Euler equations for curves and surfaces. The Euler equations related to the minimized functional of a curve of the form;

$$I(\mathbf{w}) = \int_c f(\mathbf{w}, \mathbf{w}_u, \mathbf{w}_{uu}, u) du \quad \text{where } \mathbf{w} = [x(u), y(u), z(u)] \text{ are} \quad 3.52$$

$$\frac{d^2}{du^2} \left(\frac{\partial f}{\partial x_{uu}} \right) - \frac{d}{du} \left(\frac{\partial f}{\partial x_u} \right) + \frac{\partial f}{\partial x} = 0 ,$$

$$\frac{d^2}{du^2} \left(\frac{\partial f}{\partial y_{uu}} \right) - \frac{d}{du} \left(\frac{\partial f}{\partial y_u} \right) + \frac{\partial f}{\partial y} = 0 \quad \text{and}$$

$$\frac{d^2}{du^2} \left(\frac{\partial f}{\partial z_{uu}} \right) - \frac{d}{du} \left(\frac{\partial f}{\partial z_u} \right) + \frac{\partial f}{\partial z} = 0$$

The Euler equations related to the functional of a surface of the form;

$$I(\mathbf{w}) = \int_c f(\mathbf{w}, \mathbf{w}_u, \mathbf{w}_v, \mathbf{w}_{uu}, \mathbf{w}_{uv}, \mathbf{w}_{vv}, u, v) \, du dv \quad 3.53$$

where $\mathbf{w} = [x(u, v), y(u, v), z(u, v)]$ are

$$\begin{aligned} \frac{\partial^2}{\partial u^2} \left(\frac{\partial f}{\partial x_{uu}} \right) + 2 \frac{\partial^2}{\partial u \partial v} \left(\frac{\partial f}{\partial x_{uv}} \right) + \frac{\partial^2}{\partial v^2} \left(\frac{\partial f}{\partial x_{vv}} \right) - \frac{\partial}{\partial u} \left(\frac{\partial f}{\partial x_u} \right) - \frac{\partial}{\partial v} \left(\frac{\partial f}{\partial x_v} \right) + \frac{\partial f}{\partial x} &= 0, \\ \frac{\partial^2}{\partial u^2} \left(\frac{\partial f}{\partial y_{uu}} \right) + 2 \frac{\partial^2}{\partial u \partial v} \left(\frac{\partial f}{\partial y_{uv}} \right) + \frac{\partial^2}{\partial v^2} \left(\frac{\partial f}{\partial y_{vv}} \right) - \frac{\partial}{\partial u} \left(\frac{\partial f}{\partial y_u} \right) - \frac{\partial}{\partial v} \left(\frac{\partial f}{\partial y_v} \right) + \frac{\partial f}{\partial y} &= 0 \quad \text{and} \\ \frac{\partial^2}{\partial u^2} \left(\frac{\partial f}{\partial z_{uu}} \right) + 2 \frac{\partial^2}{\partial u \partial v} \left(\frac{\partial f}{\partial z_{uv}} \right) + \frac{\partial^2}{\partial v^2} \left(\frac{\partial f}{\partial z_{vv}} \right) - \frac{\partial}{\partial u} \left(\frac{\partial f}{\partial z_u} \right) - \frac{\partial}{\partial v} \left(\frac{\partial f}{\partial z_v} \right) + \frac{\partial f}{\partial z} &= 0 \end{aligned}$$

And for the ShapeWright minimum energy principles, the Euler equations are;

$$\begin{aligned} \frac{d^2(\beta \mathbf{w}_{uu})}{du^2} - \frac{d(\alpha \mathbf{w}_u)}{du} &= L_c \mathbf{w} = \mathbf{f} \quad \text{and for a surface} \\ \left(\frac{\partial^2(\beta_{11} \mathbf{w}_{uu})}{\partial u^2} + \frac{\partial^2(\beta_{12} \mathbf{w}_{uv})}{\partial u \partial v} + \frac{\partial^2(\beta_{22} \mathbf{w}_{vv})}{\partial v^2} \right) &= L_\sigma \mathbf{w} = \mathbf{f} \\ - \left(\frac{\partial(\alpha_{11} \mathbf{w}_u + \alpha_{12} \mathbf{w}_v)}{\partial u} + \frac{\partial(\alpha_{12} \mathbf{w}_u + \alpha_{22} \mathbf{w}_v)}{\partial v} \right) & \end{aligned} \quad 3.54$$

3.6 Time, Mass and Dynamics

The notion of time was added to the system to enhance interactivity. The user manipulates the shape through time by changing the forces, geometric constraints, and material constants as a simulation of the system is run. Mass and damping were added in addition to introducing time so that the solutions of the system would automatically appear to be animated in a natural way. This was meant to enhance the notion of manipulating real physical surfaces to design shape.

The equations of motion for the system can be written using d'Alembert's principle, treating the dynamic effects as forces that modify the equilibrium equation as;

$$\frac{\partial}{\partial t} \left(\rho \frac{\partial \mathbf{w}}{\partial t} \right) + \mu \frac{\partial \mathbf{w}}{\partial t} + L\mathbf{w} = \mathbf{f}(\mathbf{w}, t) \quad 3.55$$

The first term models the inertia of the system and the second term models viscous damping forces. The damping forces continually eliminate energy as long as the system is moving. Equilibrium is achieved when the system comes to rest and both the inertial and damping terms go to zero yielding the original equation set,

$$L\mathbf{w} = \mathbf{f}. \quad 3.56$$

Dynamics do not alter the final shape of the object, only the path taken to achieve it.

The strain energy of the deformable surface has been carefully constrained to guarantee that the operator $L\mathbf{w}$ is linear. However, no such constraint has been placed on the forcing functions, \mathbf{f} . When the forces are nonlinear functions of the shape, the positive definiteness of the whole system of equations is lost. There may be several local minima to the shape energy functional that could each act as a local solution to the shape problem. By introducing time, the user can control which of the local minima shapes will be found. As an example consider an obstacle modeled by a non-linear repulsive force field. The force acting on any point of the shape will be a function of the its distance and orientation to the object.

$$\mathbf{f}_{\text{obstacle}}(u, v) = -K_{\text{ob}} \frac{(\mathbf{w}(u, v) - \mathbf{w}_0)}{(\mathbf{w}(u, v) - \mathbf{w}_0)^2} \quad 3.57$$

Figure 3.4 shows two solutions for a curve that is forced to avoid an obstacle. The user can select between these solutions by temporarily applying additional forces to make the curve move to the correct side of the obstacle and then releasing it. Without time, this selection process would be difficult to accomplish on an incremental basis.

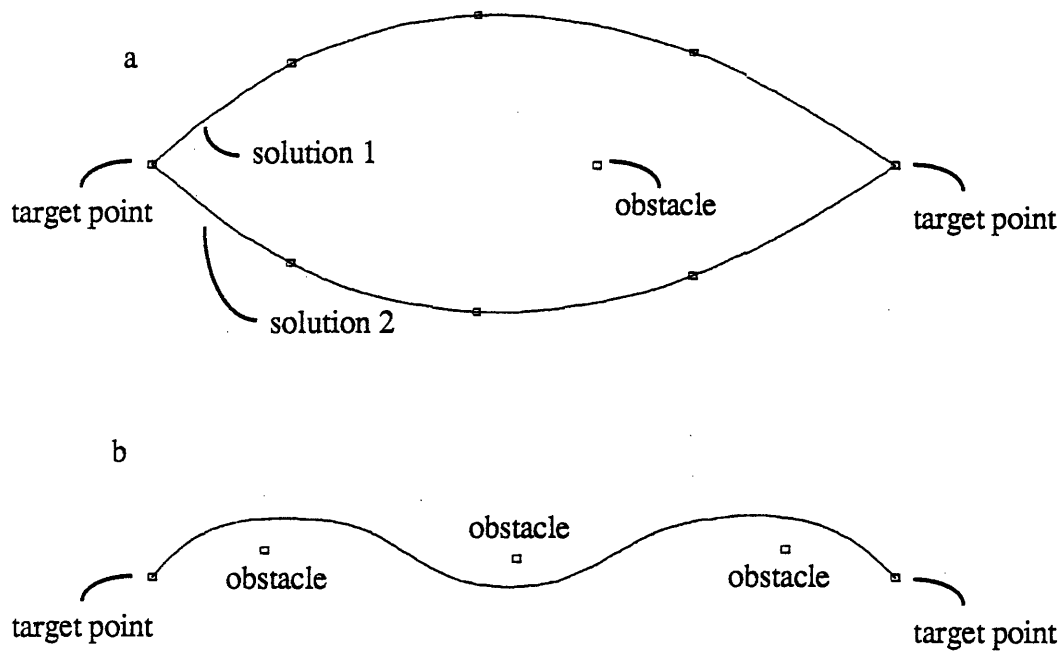


Figure 3.4) Solutions for avoiding an obstacle

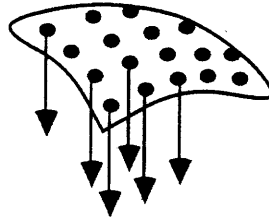
- a.) Two valid solutions for connecting two target points while avoiding one obstacle modeled as a force source
- b.) More obstacles create the potential for many solutions

3.7 Forces; Their Applications and a Survey

Forces are the mainstay for the interactive manipulation of deformable surfaces. Force fields, also called loads, can be modeled for the convenience of the user to simplify manipulating the shape or to increase the intuitive feel of sculpting. Examples of some simple force models are shown in Figure 3.5. These include pressure, springs, gravity, and the previously mentioned obstacle. “Springs” attract a particular point on the surface to

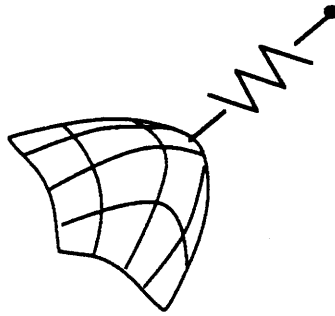
a particular point in space. “Obstacles” accomplish the reverse by pushing all points on the surface away. “Gravity” acts on all points of the surface to pull them in one direction. “Pressure” is a force applied to the surface acting in the direction of the surface normal. All of the above forces can be made to act on the entire surface or on some subregion of the surface.

o “Gravity” :



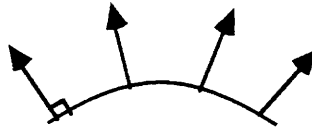
$$\mathbf{f}_g(\mathbf{w}, t) = \rho(\mathbf{w}) \mathbf{g}$$

o “Springs” :



$$\mathbf{f}_s(\mathbf{w}, t) = k(\mathbf{w} - \mathbf{w}_0)$$

o “Pressure” :



$$\mathbf{f}_p(\mathbf{w}, t) = p \mathbf{n}(\mathbf{w})$$

Figure 3.5) External forces, a survey

The most important fact about loads is that they can be applied individually to overlapping sections of the surface and the surface will respond in an intuitive manner. This is accomplished by treating the total applied force acting on a point of the surface as the sum of all the loads acting at that point.

$$\mathbf{f}_{\text{net}}(\mathbf{u}, \mathbf{v}, t) = \sum_i \mathbf{f}_i(\mathbf{u}, \mathbf{v}) \quad 3.58$$

An excellent way to specialize the interface for manipulating deformable surfaces can be achieved by custom designing force fields as needed. New loads can be designed for future situations as they become needed without modifying the general structure of the ShapeWright approach or an associated implementation. One measure of the ease of use in designing a shape is how well that shape can be parameterized. A circle in a plane is completely defined by specifying a point and a radius. By varying the radius an infinite number of circles can be designed all sharing the same center point. External forces applied to the ShapeWright geometric primitives can be used to achieve the same effect.

The implementation that was developed in this thesis has an interface that allows sets of loads to be applied to the surface. Each load is highly parameterized. A pressure is parameterized by the region of the surface that it acts on and its magnitude. A spring is parameterized by the region of the surface it acts on, the location in space of the anchor point and its magnitude. The user can explore an infinite number of shapes by applying a load and varying the load's parameters. The advantage of this approach is that, like the circle example, the family of shapes will be intuitively similar and easy to control. By applying a set of loads to the object the user can parameterize the free-form shape as it is designed in any fashion needed to control the shape.

This approach to sculpting reduces the number of parameters the user controls from hundreds and thousands to just a few. It additionally resolves the problem of how to coordinate the motion of several degrees of freedom simultaneously to achieve desired effects. It also lets each deformable shape be parameterized uniquely as a product of the steps that were taken to make it.

Another idea for specializing the interface with forces is the use of sculpting tools. A sculpting tool is a force field that acts locally around an arbitrarily shaped object. The force field can be characterized by the shortest distance between the object and the shape much like the point obstacle was in the preceding section. The geometry of the sculpting tool can be built using any convenient modeling scheme available.

Multiple deformable objects can interact with one another by through force fields. These can be designed to avoid intersecting one shape with another or to cause shapes to seek each other out. Applications that might benefit from this approach could be an automated wire router where the wires seek to group together to form wire harnesses or assembly shape modeling where shapes automatically deform to prevent interference.

3.8 Soap Films, Plane Strain, Plates and Shells

It is both interesting and informative to compare the ShapeWright deformable models with the models of real physical deformable surfaces. This comparison yields insights into how the ShapeWright models achieve fair seeking properties and how they respond to user inputs. Soap films and plane strain will be considered to develop an appreciation of systems that gain strain energy through stretch. Simple plate bending models will be examined as an example of systems that gain strain energy through bending. And a very brief glimpse of the kinematically exact treatment of the bending and stretching of free-form shapes known as shell theory will be given to show that the decoupling of strain and bending energies is a physical as well as a ShapeWright property.

3.8.a) Soap Films

A soap film has a constant surface tension that is a property of the materials that come in contact, for example soapy water and air. Such surfaces deform to minimize their area while being constrained to meet their geometric constraints. Consider a soap film on a child's bubble-blowing toy. From the section on parametric geometry we know that the energy of such a surface will be given as;

$$E_{\text{film}} = k \int_{\sigma} \mathbf{w}_u \times \mathbf{w}_v \, du dv = k \int_{\sigma} \sqrt{\det(\mathbf{G})} \, du dv \quad 3.59$$

Soap films build minimal surfaces. The properties of the minimal surface are such that the mean curvature is everywhere zero. The surfaces cannot support self-intersecting or slope-discontinuous singularities. Overall, soap films are very well behaved surfaces excellent for interpolating a data set. Their energy functionals are completely defined in terms of the first partial derivatives of the surface which are the terms weighted by α in the ShapeWright primitives.

$$E_{\text{film}} = E_{\text{film}}(\mathbf{w}_u, \mathbf{w}_v) \quad 3.60$$

The ShapeWright deformable primitives hope to achieve some of the excellent interpolation properties of a minimal surface by attempting to minimize their area through the use of α

weighted terms. By analogy, the α parameters are acting somewhat like a surface tension parameter. As discussed before, since the area is not minimized directly the ShapeWright primitives will not be making exactly minimal surfaces. Shapewright will only make surfaces with minimal surface-like properties.

3.8.b) Plane Strain

In linear elasticity theory, deformed shape, y , is defined as the sum of a rest shape, x , and a deformation, w as;

$$y = x + w \quad 3.61$$

where x, y , and w are vectors in three space spanned by e_1, e_2, e_3 .

The actual strains of a full three dimensional body are defined as;

$$\epsilon_{ij} = \frac{1}{2}(w_{i,j} + w_{j,i}) \quad 3.62$$

where the notation, $w_{i,j}$, represents the partial in the j th direction of the i th component of w . Both i and j vary from 1 to 3.

In elasticity theory, strain is related to stress by Hooke's law. Ignoring initial stresses this can be expressed as the linear symmetric relation;

$$\bar{\sigma} = C_{\text{hooke}} \bar{\epsilon} \quad 3.63$$

$$\text{where } \bar{\sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{bmatrix}, \quad C_{\text{hooke}} = \text{Material Property Matrix, and } \bar{\epsilon} = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \epsilon_{12} \\ \epsilon_{23} \\ \epsilon_{31} \end{bmatrix}$$

The strain energy of such a system is given by

$$E_{\text{elasticity}} = \int_{\gamma} \bar{\sigma}^T \bar{\varepsilon} = \int_{\gamma} \bar{\varepsilon}^T C_{\text{hooke}} \bar{\varepsilon} \quad 3.64$$

Plane strain is a special case where deformations out of the plane, in the direction of the e^3 axis are prohibited and the deformation is the same for all planes. In this case we can consider the two dimensional problem where the vectors x , y , and w are now constrained to the space spanned by the vectors e_1 , e_2 . The stress and strain vectors reduce to,

$$\bar{\sigma}_{\text{ps}} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix}, \quad \text{and} \quad \bar{\varepsilon}_{\text{ps}} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{bmatrix} \quad 3.65$$

The material law for an isotopic material with a Young's modulus, E , and a Poisson's ratio, ν , for plane stress will be of the form;

$$C_{\text{ps}} = k_{\text{ps}}(E, \nu) \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \quad 3.66$$

Placing these definitions into the strain energy functional and equating e_1 with u and e_2 with v yields;

$$E_{\text{plain strain}} = k_{\text{ps}} \int_{\sigma} [\mathbf{w}_u \ \mathbf{w}_v \ (\mathbf{w}_u + \mathbf{w}_v)] \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \begin{bmatrix} \mathbf{w}_u \\ \mathbf{w}_v \\ \mathbf{w}_u + \mathbf{w}_v \end{bmatrix} du dv \quad 3.67$$

The difference between this energy functional and the ShapeWright α weighted functional can be made by eliminating the Poisson's ration, $\nu = 0$, and ignoring the energy due to shear deformations. The ShapeWright energy functional ignores these terms to improve the intuitive response of the shape to input forces. By analogy, we can interpret the α terms that weight stretch to be akin to Young's modulus acting for a linear elastic material.

3.8.c) Plate Bending

The simplest plate theory is the classical small deflection theory developed by Lagrange. All deformations of the plate are due to bending, deformations due to stretch are excluded. Linear plate theory is built on the assumptions:

- i) points which lie on a normal to the mid-plane of the undeflected plate lie on a normal to the mid-plane of the deflected plate.
- ii) the stresses normal to the mid-plane of the plate are negligible compared with the stresses in the plane of the plate.
- iii) the slope of the deflected plate in any direction is small so that the local curvature can be approximated linearly.
- iv) the mid-plane of the plate is a 'neutral plane' where mid-plane stresses due to non-developable deformations may be ignored. The plate does not stretch.

In this theory the plate deformation is limited to the direction normal to the plane taken to be the \mathbf{e}_3 axis. The deformation reduces from a three space vector to a scalar field, $\mathbf{w} = w \mathbf{e}_3$. Thus the neutral plane deformed shape is given by the implicit relation,

$$\mathbf{y} = \mathbf{x} + w \mathbf{e}_3 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w(u,v) \end{bmatrix} \quad 3.68$$

The slopes of the mid-plane are w_u and w_v . For small angle slopes the deformation of the plate in a plane at some height z above the neutral plane will be given by;

$$\mathbf{w} = \begin{bmatrix} -z w_u \\ -z w_v \\ w \end{bmatrix} \quad 3.69$$

The strains in the elevated plane will be

$$\bar{\epsilon}_{pb} = \begin{bmatrix} \bar{\epsilon}_{11} \\ \bar{\epsilon}_{22} \\ \bar{\epsilon}_{12} \end{bmatrix} = \begin{bmatrix} -z w_{uu} \\ -z w_{vv} \\ -2z w_{uv} \end{bmatrix} \quad 3.70$$

Assuming a state of plane stress, the stress in the plate is given as;

$$\bar{\sigma}_{pb} = \bar{\sigma}_{pb}(z) = C_{ps} \bar{\epsilon}_{pb} \quad 3.71$$

The general state of stress can be used to find the moments that bend in u and v and twist by integrating over the thickness of the plate

$$M_{pb} = \begin{bmatrix} M_u \\ M_v \\ M_{uv} \end{bmatrix} = \int_{\sigma} z \bar{\sigma}_{pb} du dv = D_{pb} C_{ps} \begin{bmatrix} -w_{uu} \\ -w_{vv} \\ -2w_{uv} \end{bmatrix} \quad 3.72$$

with $D = Et^3/12(1-\nu^2)$; t = plate thickness

The equivalent 'strain' for this system is

$$\epsilon_{pb} \equiv \begin{bmatrix} -w_{uu} \\ -w_{vv} \\ -2w_{uv} \end{bmatrix} \quad 3.73$$

The terms w_{uu} and w_{vv} approximates curvature, $\kappa_{uu} = -w_{uu}/(1 + w_u^2)^{3/2}$ and $\kappa_{vv} = -w_{vv}/(1 + w_v^2)^{3/2}$ when w_u and w_v are smaller than 1.

The strain energy of a bending plate is given by;

$$E_{\text{plate bending}} = D_{pb} \int_{\sigma} [-w_{uu} -w_{vv} -2w_{uv}] \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \begin{bmatrix} -w_{uu} \\ -w_{vv} \\ -2w_{uv} \end{bmatrix} du dv \quad 3.74$$

By analogy between the bending plate and ShapeWright β weighted energy functional we can conclude that in the range of small deformations the β 's model the effects of Young's

modulus, E on bending. Also we see that all effects due to Poisson's ratio have once again been eliminated.

3.8.d) Shell Theory Results

We have looked at soap films and plane strain and discovered that the ShapeWright energy functionals are very similar to these models providing effects that minimize area and resist stretch. The examination of the plate bending model indicates by analogy that the β weighted terms of the ShapeWright energy functional act to resist bending. One major question remaining is how these effects couple together. In the ShapeWright primitives these were assumed to simply add together. This assumption garners some credibility from the results of shell theory.

Shell theory is a careful kinematic analysis of the elastic behavior of bending and stretching a fully free-form shape made of a thin sheet. Due to the highly coupled nature of the geometry and the stiffness very nonlinear relations result. However, for small deformations about the rest shape of a shell structure the energy is actually modeled as the sum of a term due to bending and a term due to stretch.

$$E_{\text{shell}} = \int_{\sigma} (\text{bending term}) + (\text{stretch term}) \, dudv \quad 3.75$$

In fact, the early Finite Element solutions to shell problems used elements whose stiffness were due to the decoupled effects of plate bending and plane strain.

$$\mathbf{K}_{\text{shell}} = \begin{bmatrix} \mathbf{K}_{\text{plain strain}} & 0 \\ 0 & \mathbf{K}_{\text{plate bending}} \end{bmatrix} \quad 3.76$$

Once again by analogy, we conclude that the effects of resisting bending and stretching can be combined effectively in the ShapeWright primitives by treating them as decoupled events.

3.9 Chapter 3 Summary

A free-form shape can be represented parametrically. These surface representations can be used to model deformable shapes in an energy functional. The actual shape assumed by a deformable primitive can be found as the one shape out of all possible shapes that meet the boundary conditions which minimizes the energy functional. An equivalent statement of the problem is to solve the related Euler differential equations.

ShapeWright energy functionals consist of the sum of a stretching and a bending term. These terms are weighted by α and β so that the behavior of the surface can be modified during run time. By resisting stretch the ShapeWright primitives hope to mimic the behavior of Soap films and produce shapes with minimal material. When the user manipulates the constraining character lines this property will act to prevent the surface from folding like a curtain on a curtain rod. Resistance to bending was added to increase the fairness of the produced shapes. Resistance to bending is a property common to simple beams and plates which are physical objects that have long been exploited to produce fair geometry. Deformation effects that couple changes in shape between orthogonal directions were eliminated. This action was equivalent to setting Poisson's ratio to zero in linear isotropic elasticity theory.

The actual ShapeWright energy functionals are designed to enhance the speed at which numerical solutions can be obtained. This is done to enable interactive design.

The notions of time, and dynamics were added to the ShapeWright equation set to enhance user interactivity and to allow for user controlled selection between alternative solutions. Forces are identified as the primary means to achieve interactive sculpting of deformable shapes. The notion of parameterized loads is introduced to simplify the design of very complicated shapes.

In the next chapter solution schemes and actual ShapeWright primitives are discussed.

4 Curve and Surface Geometric Primitives

This chapter presents solutions to the variational problem of finding the ShapeWright geometry given in Chapter 3. That problem is; find the shape, w , for a curve or surface which minimizes one of the following functionals;

$$E_{\text{curve}} = \int_{\text{curve}} \alpha(u) w_u^2 + \beta(u) w_{uu}^2 - 2 f w \, du \quad 4.1$$

$$E_{\text{surface}} = \int_{\sigma} \left(\begin{aligned} &(\alpha_{11} w_u^2 + 2\alpha_{12} w_u w_v + \alpha_{22} w_v^2) \\ &+ (\beta_{11} w_{uu}^2 + 2\beta_{12} w_{uv}^2 + \beta_{22} w_{vv}^2) \end{aligned} \right) - 2 f w \, dudv \quad 4.2$$

in such a way as to satisfy the following requirements;

- o Interactivity
- o Continuity
- o Constraints
- o Topology

needed to support the three step ShapeWright design paradigm as;

- 1.) Build Character Lines
- 2.) Skin Faces with Deformable Surfaces
- 3.) Sculpt Faces into Final Shape

Analytic solutions to the above problem are available over very simple domains. Since it is our ambition to model complicated topologies to be represented by arbitrarily shaped domains in the parameter space we consider numerical solutions.

The beginning of this chapter gives a finite difference solution as the simplest means to solve the ShapeWright problem. The finite element method is introduced to develop a shape representation capable of satisfying the above requirements. A technique for time integration is given appropriate for either the finite element or finite difference solution approaches. Constraint enforcement schemes are reviewed briefly in search of a scheme to implement the skinning step of the ShapeWright paradigm.

The remainder of the chapter is spent in developing the actual ShapeWright surface and curve primitives. Triangular finite elements are used to satisfy the topology requirement combined with a set of shape functions that can be used to satisfy the constraint and the continuity requirements. Barycentric coordinates are introduced for building functions defined over triangular domains. The details of developing the element stiffness matrices, forcing vectors and enforcing geometric constraints are presented. The same is done for a curve element based on Hermite polynomial shape functions.

4.1 Finite Difference Solutions

The finite difference method for solving the ShapeWright deformable model problem seeks solutions to the Euler differential equations;

$$\begin{aligned}
 \text{Curve:} \quad & \frac{d^2(\beta \mathbf{w}_{uu})}{du^2} - \frac{d(\alpha \mathbf{w}_u)}{du} = L_c \mathbf{w} = \mathbf{f} \quad 4.3 \\
 \text{Surface:} \quad & - \left(\frac{\partial^2(\beta_{11} \mathbf{w}_{uu})}{\partial u^2} + \frac{\partial^2(\beta_{12} \mathbf{w}_{uv})}{\partial u \partial v} + \frac{\partial^2(\beta_{22} \mathbf{w}_{vv})}{\partial v^2} \right) \\
 & - \left(\frac{\partial(\alpha_{11} \mathbf{w}_u + \alpha_{12} \mathbf{w}_v)}{\partial u} + \frac{\partial(\alpha_{12} \mathbf{w}_u + \alpha_{22} \mathbf{w}_v)}{\partial v} \right) = L_\sigma \mathbf{w} = \mathbf{f}
 \end{aligned}$$

by approximating the differential terms by differences of the actual function. These approximations derive from a Taylor series expansion of the function about a point. For a curve the Taylor series expansion is;

$$\mathbf{w}(u+du) = \mathbf{w}(u) + \mathbf{w}_u du + \frac{1}{2} \mathbf{w}_{uu} du^2 + O(du^3) \quad 4.4$$

Solving for \mathbf{w}_u yields a forward difference approximation to the first derivative of \mathbf{w} that is accurate to first order terms.

$$\mathbf{w}_u = \frac{\mathbf{w}(u+du) - \mathbf{w}(u)}{du} + \frac{1}{2} \mathbf{w}_{uu} du + O(du^2) \quad 4.5$$

More accuracy can be achieved by combining this result with another Taylor series expansion in the opposite direction;

$$w(u-du) = w(u) - w_u du + \frac{1}{2} w_{uu} du^2 + O(du^3) \quad 4.6$$

Subtracting the two Taylor series and solving for w_u yields a central difference approximation to w_u accurate to second order terms.

$$w_u = \frac{w(u+du) - w(u-du)}{2du} + O(du^2) \quad 4.7$$

Various combinations of Taylor series expansions yield a family of approximations for the different differentials in the above equations.

To solve the above Euler differential equations, the parametric domain is broken up into an array of discrete points taken at even intervals, as shown in Figure 4.1, appropriate for the finite difference approximations;

$$a(u) \approx a[i] = a(i \cdot \Delta u) \quad \text{and} \quad a(u,v) \approx a[i,j] = a(i \cdot \Delta u, j \cdot \Delta v) \quad 4.8$$

where i,j are integers that range from 1 to m and 1 to n respectively
 $\Delta u, \Delta v$ are the spacing intervals in the u and v directions.

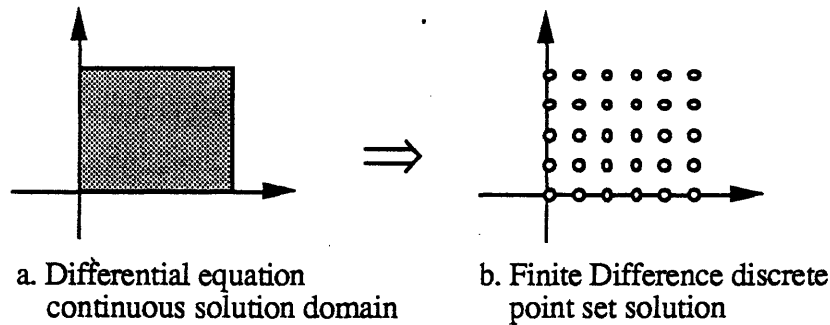


Figure 4.1) Discretizing the uv domain for a finite difference solution

Differential terms in the original Euler equations are approximated with the following differences;

$$w_u[i] = \frac{w[i+1] - w[i-1]}{2\Delta u} \quad \text{and} \quad w_{uu}[i] = \frac{w[i+1] - 2w[i] + w[i-1]}{\Delta u^2} \quad 4.9$$

Applying these definitions to themselves yields higher order differentials;

$$\frac{d^2\beta(u)w_{uu}(u)}{du^2} = \frac{\beta[i+1]w_{uu}[i+1] - 2\beta[i]w_{uu}[i] + \beta[i-1]w_{uu}[i-1]}{\Delta u^2} \quad 4.10$$

and for the surface we also need $\frac{d^2\beta_{12}(u,v)w(u,v)}{dudv}$ which can be found with the following;

$$\begin{aligned} \frac{d^2w(u,v)}{dudv} &= \frac{dw_v(u,v)}{du} = \frac{w_v[i+1,j] - w_v[i-1,j]}{2\Delta u} \\ &= \frac{w[i+1,j+1] - w[i+1,j-1] - w[i-1,j+1] + w[i-1,j-1]}{2\Delta u \, 2\Delta v} \end{aligned} \quad 4.11$$

Armed with the above definitions for the differential terms, the original Euler equations can be turned into a set of linear algebraic equations where the unknowns are the values of the shape, w , at the array of sampled points, $w[i,j]$. This is accomplished by requiring that the finite difference approximation to the differential equation hold simultaneously at all of the sampled points. Thus the original differential equation becomes $n*m$ (or n for the curve) linear algebraic equations each evaluated at one of the array points.

The beauty of the finite difference method is that all of the algebraic equations are of the same form. This regularity is best expressed by showing the template of the equation. The template is a map showing how the equation for one point location depends on its neighbors. The templates for the curve and surface equations are given in Figure 4.2. Reading the template tells how to generate the algebraic equation for a typical node in the array. First the template is centered on the node in question, then the algebraic equation for the node is written weighting each neighboring node with the coefficient in the template that aligns with that neighbor. Moving the template to every node in the array generates all of the required equations.

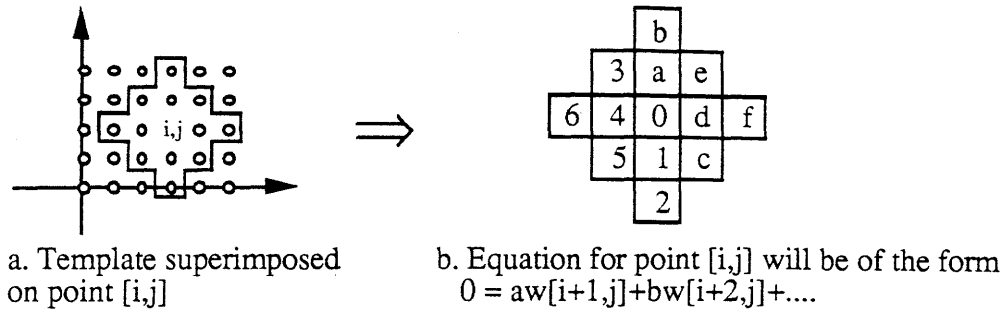


Figure 4.2) Template shape for surface finite difference equations

6: $\beta_{11}[i-1,j]$	5: $2\beta_{12}[i-1,j-1]$
4: $-\alpha_{11}[i-1,j] - \alpha_{12}[i-1,j]$ $-2\beta_{11}[i-1,j] - 2\beta_{11}[i,j] - 2\beta_{12}[i-1,j-1] - 2\beta_{12}[i-1,j]$	3: $\alpha_{12}[i-1,j]$ $2\beta_{12}[i-1,j]$
2: $\beta_{22}[i,j-1]$	1: $-\alpha_{22}[i,j-1] - \alpha_{12}[i,j-1]$ $-2\beta_{22}[i,j] - 2\beta_{22}[i,j-1] - 2\beta_{12}[i-1,j-1] - 2\beta_{12}[i,j-1]$
0: $\alpha_{11}[i,j] + \alpha_{11}[i-1,j] + \alpha_{22}[i,j] + \alpha_{22}[i,j-1] + 2\alpha_{12}[i,j]$ $\beta_{11}[i-1,j] + 4\beta_{11}[i,j] + \beta_{11}[i+1,j]$ $\beta_{22}[i,j-1] + 4\beta_{22}[i,j] + \beta_{22}[i,j+1]$ $2\beta_{12}[i,j] + 2\beta_{12}[i,j-1] + 2\beta_{12}[i-1,j] + 2\beta_{12}[i-1,j-1]$	a: $-\alpha_{22}[i,j] - \alpha_{12}[i,j]$ $-2\beta_{22}[i,j] - 2\beta_{22}[i,j+1] - 2\beta_{12}[i-1,j] - 2\beta_{12}[i,j]$
c: $\alpha_{12}[i,j-1]$ $2\beta_{12}[i,j-1]$	b: $\beta_{22}[i,j+1]$
e: $2\beta_{12}[i,j]$	d: $-\alpha_{11}[i,j] - \alpha_{12}[i,j]$ $-2\beta_{11}[i,j] - 2\beta_{11}[i+1,j] - 2\beta_{12}[i,j-1] - 2\beta_{12}[i,j]$
	f: $\beta_{11}[i+1,j]$

Figure 4.3) Surface template coefficients for Figure 4.2

The equation set is completed by approximating the forcing functions with finite differences as;

$$f(u,v) \approx f[i,j] = f(i*\Delta u, j*\Delta v) \quad 4.12$$

The regularity of the algebraic equations results in matrix equations that are symmetric and banded, and thus easy to solve. Ordering the unknowns and forces into discrete vectors as;

$$\mathbf{X} = \begin{bmatrix} \mathbf{w}[1,1] \\ \mathbf{w}[1,2] \\ \vdots \\ \mathbf{w}[n,m] \end{bmatrix} \quad \text{and} \quad \mathbf{F} = \begin{bmatrix} \mathbf{f}[1,1] \\ \mathbf{f}[1,2] \\ \vdots \\ \mathbf{f}[n,m] \end{bmatrix} \quad 4.13$$

yields the matrix equation approximation to the Euler equations as;

$$\mathbf{K}_d \mathbf{X} = \mathbf{F} \quad 4.14$$

Solving for the unknowns, \mathbf{X} , yields an approximate solution for the original shape, \mathbf{w} , at a set of discrete points in space. No shape information between the points is given by this method. Dealing with the time derivatives in the inertial and damping terms are discussed in section 4.3.

The above system of equations were used as the basis for a free-form modeling implementation that is further described in [Celniker_89] and is very much the basis for the solution techniques adopted by Terzopoulos et.al. for extracting shape from video images [Terzopoulos_87] and Bloor and Wilson for defining shape as a boundary value problem [Bloor_90]. This implementation revealed several facts. The positive conclusions were that the deformable energy paradigm for shape generation could be made to work. Very complicated shapes were generated with minimal input. Additionally, it was found that the system of equations could be solved quickly enough on current hardware platforms (Silicon Graphics 4D GT) to enable the interactive sculpting of surfaces of reasonable sized problems.

The shortcomings of this approach for Computer Aided Design were all due to the fact that the finite difference method requires that shape be represented as an array of discrete points. This limits the topology of the mapping to the unit square, violates the need for continuous descriptions of surfaces for manufacturing and makes it very difficult to enforce geometric constraints on the surface. For example, a discrete point in space has no tangents.

The idea of interpolating such a set of points to define a continuous surface was rejected due to the problems inherent in interpolations and the difficulty of enforcing geometric

constraints simultaneously on the finite element point locations and the interpolation. Instead the Ritz finite element method was applied to solve these problems while preserving the strengths of the deformable modeling approach.

4.2 The Finite Element Method

This section is divided into two parts. The first part presents the mechanics of solving the deformable model problems given geometric constraints and arbitrary forcing functions with the finite element technique. Following the procedure presented deformable finite element matrices can be defined once the shape functions are selected. The second part presents some of the theory of finite elements to show that the approximate solution found with the Ritz finite element is the best that can be obtained with any given set of shape functions.

The notation introduced in sections 3.4.d), 3.4.f) and 3.5 is used liberally in this section.

4.2.a) The Ritz Finite Element Method

The Ritz solution method for solving the ShapeWright deformable model problem starts with the variational statement of the problem: find the shape \mathbf{w} to minimize the energy functional;

$$E_{\text{curve}} = I_{\text{curve}}(\mathbf{w}) = \int_{\text{curve}} \alpha(u) \mathbf{w}_u^2 + \beta(u) \mathbf{w}_{uu}^2 - 2 \mathbf{f} \mathbf{w} \, du \quad 4.15$$

$$E_{\text{surface}} = I_{\text{surface}}(\mathbf{w}) = \int_{\sigma} \left(\begin{aligned} &(\alpha_{11} \mathbf{w}_u^2 + 2\alpha_{12} \mathbf{w}_u \mathbf{w}_v + \alpha_{22} \mathbf{w}_v^2) \\ &+ (\beta_{11} \mathbf{w}_{uu}^2 + 2\beta_{12} \mathbf{w}_{uv}^2 + \beta_{22} \mathbf{w}_{vv}^2) \end{aligned} \right) - 2 \mathbf{f} \mathbf{w} \, dudv \quad 4.16$$

Instead of approximating the equations as in the finite difference approach, the solution, \mathbf{w} , is approximated by \mathbf{w}^h , which is a weighted sum of continuous shape functions. The shape functions, ϕ_i , are fixed in advance and the weights, \mathbf{x}_i , are unknown.

$$\mathbf{w}(u,v) \approx \mathbf{w}^h(u,v) = \sum_i \mathbf{x}_i \phi_i(u,v) \quad 4.17$$

The Ritz method substitutes this approximation into the energy functional, changing and simplifying the nature of the problem. The unknown used to be $w(u,v)$, a continuous function in the parametric domain. Now the unknowns are the discrete set of weights in the Ritz approximation. The problem now is to find the set of weights, x_i , that will minimize the energy functional;

$$I(w^h) \tag{4.18}$$

In the finite difference method the forcing function, f , as well as the solution, w , is approximated as a set of discrete points in space. In the finite element method the function f , is a function defined over the whole parametric region. We need to consider what class of functions, f , will be allowed as data thus determining what class of functions, w^h , will have to be searched to find the minimum to the above variational problem.

The choice made in this work is to admit all functions f which have finite energy. This means that;

$$\int_a f(a)^2 da < \infty \tag{4.19}$$

and any piecewise smooth function f is allowed. This constraint is more stringent than required but is acceptable to the ShapeWright application. The set of functions satisfying the finite energy property also exhibit the property that the linear sum of any two such functions also satisfies the above property;

$$\text{if } f(a) = c_1 f_1(a) + c_2 f_2(a) \quad \text{when } \int_a f_1^2 da < \infty \quad \text{and} \quad \int_a f_2^2 da < \infty \tag{4.20}$$

$$\text{then } \int_a f^2 da < \infty$$

In the accepted terminology of mathematics, such a set of functions is called a space. The space of functions satisfying the above energy function requirement is denoted \mathcal{H}^0 . The superscript in this case, zero, defines how many derivatives are required to have finite energy. The norm of these spaces is taken to be the square root of the energy, thus for the spaces \mathcal{H}^0 , \mathcal{H}^2 , and \mathcal{H}^4 , the energies and norms are given as,

$$f \in \mathcal{H}^0 \Rightarrow \int_a f^2 da < \infty \quad \text{and} \quad \|f\|_0 = \left[\int_a f^2 da \right]^{1/2} \quad 4.21$$

$$f \in \mathcal{H}^2 \Rightarrow \int_a f''^2 + f'^2 + f^2 da < \infty \quad \text{and} \quad \|f\|_2 = \left[\int_a f''^2 + f'^2 + f^2 da \right]^{1/2}$$

$$f \in \mathcal{H}^4 \Rightarrow \int_a f''''^2 + f'''^2 + f''^2 + f'^2 + f^2 da < \infty$$

$$\text{and} \quad \|f\|_4 = \left[\int_a f''''^2 + f'''^2 + f''^2 + f'^2 + f^2 da \right]^{1/2}$$

A differential equation can be thought of as an operator that maps functions from one space to another. The Euler equation for a ShapeWright curve maps functions from \mathcal{H}^4 space to \mathcal{H}^0 space. Introducing operator notation, this equation can be written as;

$$L_{\text{curve}} w = f \quad 4.22$$

$$\text{where } L_{\text{curve}} = \beta \frac{d^4}{du^4} - \alpha \frac{d^2}{du^2}$$

We know that the equivalent problem to solving the Euler equation is minimizing the related variational principle. What class of functions should be considered in trying to find that minimum? For the curve, clearly we should include at least all \mathcal{H}^4 functions with proper boundary conditions. However, the energy minimization principle only requires the second derivatives of w and not the fourth. One of the great advantages of the finite element theory is that the space of functions need not be restricted to \mathcal{H}^4 functions but can be expanded to include all \mathcal{H}^2 functions that meet the boundary conditions that are a function of the position and tangents only.

Including this larger class of functions will not change the minimum value. That will indeed be a function in \mathcal{H}^4 space as indicated by the Euler equations, however it does simplify the problem considerably. Only functions from \mathcal{H}^2 space need to be examined in search of the minimum. All functions which are C^1 continuous will be in the \mathcal{H}^2 space. This simplifies the problem of defining the Φ_i shape functions needed to approximate the original shape. We do not need to consider functions which are C^3 continuous as the Euler equation would indicate but only functions which are C^1 continuous as needed by the variational problem.

This property of the variational problem has significant implications for shape smoothness. Although the shapes to be used to solve the variational problem will only guarantee C^1 continuity, the solutions found will tend to be C^3 continuous. The only time this will be violated is if the number of degrees of freedom in a curve is too few to allow the approximation procedure to find a good estimate of the actual solution.

Because the energy functional has been constrained to be quadratic and since the Φ_i shape functions are known, the above integral can be taken yielding a quadratic matrix equation to be minimized;

$$\min (\mathbf{X}^T \mathbf{K}_\sigma \mathbf{X} - \mathbf{F}_\sigma^T \mathbf{X}) \quad 4.23$$

where the unknowns and the shape functions have been ordered into column vectors as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \quad \text{and} \quad \Phi^T = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_n \end{bmatrix} \quad 4.24$$

and \mathbf{K}_σ and \mathbf{F}_σ define the stiffness matrix and forcing vector. These terms are given by;

$$\mathbf{K}_\sigma = \int_{\sigma} \Phi_b^T \bar{\beta} \Phi_b + \Phi_s^T \bar{\alpha} \Phi_s \, dudv \quad 4.25$$

$$\mathbf{F}_\sigma = \int_{\sigma} \Phi^T \mathbf{f} \, dudv \quad 4.26$$

$$\text{where } \Phi_b = \begin{bmatrix} \Phi_{uu} \\ \Phi_{vv} \\ 2\Phi_{uv} \end{bmatrix}, \quad \Phi_s = \begin{bmatrix} \Phi_u \\ \Phi_v \end{bmatrix},$$

$$\bar{\alpha} = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{bmatrix}, \quad \bar{\beta} = \begin{bmatrix} \beta_{11} & & \\ & \beta_{22} & \\ & & \beta_{12} \end{bmatrix}$$

Finding the minimum to the above quadratic matrix equation is equivalent to solving the matrix problem;

$$\mathbf{K}_\sigma \mathbf{X} = \mathbf{F}_\sigma \quad 4.27$$

The Ritz method becomes the finite element method when the shape functions are constrained to be zero everywhere except in the neighborhood of some node in the surface.

The matrix \mathbf{K}_σ is guaranteed to be symmetric and positive definite due to the limitations imposed on the energy functional. The finite element constraint on the shape functions will make \mathbf{K}_σ sparse and generally banded.

Both the finite difference and finite element methods result in solving a set of algebraic equations with convenient properties for rapid solution. The difference between the two methods is the semantics of the unknown variables. In finite differences the unknowns are a discrete set of points in space. In finite elements the unknowns define a unique continuous function appropriate for the ShapeWright deformable models.

The finite element method presented here can be applied to any set of shape parameters. As will be discussed in section 4.4 the shape functions used in ShapeWright were selected for their ability to enforce the set of geometric requirements needed for the ShapeWright skinning step. In general the deformable modeling approach can be applied to any discrete parametric representation scheme. Thus deformable B-splines, Bezier polynomials, Rational B-splines, Hermite polynomials, etc. are all possible.

4.2.b) The Finite Element Theory

The previous section discussed the mechanism to approximate the solution of a differential equation with the finite element method. In this section a very brief review of the theory of the finite element is presented to show how the approximation relates to the actual solution. The answer to that question is that the approximate solution is as close to the real solution as the degrees of freedom of the approximation will allow in that the energy of the error is a minimum.

The problem being solved can be stated as either a linear differential equation or a minimum principle as;

$$Lw = f \quad \text{and} \quad I(w) = a(w, w) - 2(f, w) \quad 4.28$$

The w that solves $Lw = f$ minimizes $I(w)$. The Ritz approximation is to find a minimum over some approximate space of functions, w^h . The Ritz theory states that the w^h which minimizes $I(w^h)$ is the best approximation to the w that minimizes $I(w)$ in the sense that the energy of the resulting error, $w - w^h$, is a minimum at w^h ;

$$\text{when } w^h \text{ minimizes } I(w^h) \text{ it also minimizes } a(w - w^h, w - w^h) \quad 4.29$$

The proof for this starts by considering variations about the minimum solution function. Since $I(w^h)$ is a minimum then arbitrarily small variations about w^h given by, $w^h + \epsilon e^h$, will have to increase the value of the minimum principle;

$$I(w^h) \leq I(w^h + \epsilon e^h) \quad 4.30$$

$$\leq I(w^h) + 2\epsilon[a(w^h, e^h) - (f, e^h)] + \epsilon^2 a(e^h, e^h)$$

where ϵ is an arbitrary scalar and

e^h is any function from the sub-space spanned by the shape functions, ϕ_i

Since ϵ can be of arbitrary sign the relation, $a(w^h, e^h) - (f, e^h)$ must equal zero for the above inequality to hold for any function e^h . We conclude;

$$a(w^h, e^h) = (f, e^h) \quad \text{for all functions } e^h. \quad 4.31$$

which is a statement of the virtual work principle. Now to find the energy of the error. We know from the original problem statement that $a(\mathbf{w}, \mathbf{e}) = (\mathbf{f}, \mathbf{e})$ is true for all functions \mathbf{e} in the admissible space of the solution \mathbf{w} . Since \mathbf{e}^h is a function which comes from the subspace containing \mathbf{e} then $a(\mathbf{w}, \mathbf{e}^h) = (\mathbf{f}, \mathbf{e}^h)$ is certainly true. Subtracting this result and the virtual work relation for the approximate space yields;

$$\begin{array}{rcl} a(\mathbf{w}, \mathbf{e}^h) & = & (\mathbf{f}, \mathbf{e}^h) \\ -a(\mathbf{w}^h, \mathbf{e}^h) & = & (\mathbf{f}, \mathbf{e}^h) \\ \hline a(\mathbf{w} - \mathbf{w}^h, \mathbf{e}^h) & = & 0 \end{array} \quad 4.32$$

The function \mathbf{e}^h can be any function from the space spanned by the shape functions ϕ_i . The energy inner product of the error, $\mathbf{w} - \mathbf{w}^h$, and the function \mathbf{e}^h is zero indicating that the error is orthogonal to the space spanned by the shape functions. This will only be true when the energy of the error is a minimum; $a(\mathbf{w} - \mathbf{w}^h, \mathbf{w} - \mathbf{w}^h)$;

$$a(\mathbf{w} - \mathbf{w}^h, \mathbf{w} - \mathbf{w}^h) \leq a(\mathbf{w} - (\mathbf{w}^h + \epsilon \mathbf{e}^h), \mathbf{w} - (\mathbf{w}^h + \epsilon \mathbf{e}^h)) \quad 4.33$$

Thus the Ritz method finds the one solution, \mathbf{w}^h , that minimizes the error between the approximate solution and the actual solution.

With this result it is now easy to show an additional property of the estimated solution. The energy of the estimated solution is always less than the energy of the actual solution. Using $a(\mathbf{w} - \mathbf{w}^h, \mathbf{w}^h) = 0$ to imply $a(\mathbf{w}, \mathbf{w}^h) - a(\mathbf{w}^h, \mathbf{w}^h) = 0$, we can write;

$$\begin{aligned} a(\mathbf{w} - \mathbf{w}^h, \mathbf{w} - \mathbf{w}^h) &= a(\mathbf{w}, \mathbf{w}) - 2a(\mathbf{w}, \mathbf{w}^h) + a(\mathbf{w}^h, \mathbf{w}^h) \\ &= a(\mathbf{w}, \mathbf{w}) - a(\mathbf{w}^h, \mathbf{w}^h) \end{aligned} \quad 4.34$$

The estimated energy $a(\mathbf{w}^h, \mathbf{w}^h)$ is always less than the actual energy of the solution $a(\mathbf{w}, \mathbf{w})$.

4.3 Mass, Damping and Integrating Through Time

Both the finite difference and the finite element methods as discussed, result in a static set of equilibrium equations of the form;

$$\mathbf{KX} = \mathbf{F} \quad 4.35$$

Inertial and damping effects are added to the system by augmenting this equilibrium equation as;

$$\mathbf{M} \frac{d^2\mathbf{X}}{dt^2} + \mathbf{C} \frac{d\mathbf{X}}{dt} + \mathbf{KX} = \mathbf{F} \quad 4.36$$

These equations are solved through time using an implicit integration scheme based on finite difference approximations of the time derivatives. The system of equations are satisfied at a set of discrete times separated by equal time intervals, Δt . The approach described in this section guarantees that a stable configuration will be found at each time step. The user can modify the time interval, Δt , to adjust the system speed of response to inputs. Very large Δt values will make the system jump directly from one equilibrium configuration to another eliminating the dynamics of the interface. Very small Δt values will exaggerate the dynamics and result in very oscillatory systems that take a long time to settle down.

The mass and stiffness matrices are taken to be diagonal for simplicity, $\mathbf{M} = m\mathbf{I}$ and $\mathbf{C} = c\mathbf{I}$ where m and c are scalar values for mass and damping and \mathbf{I} is the identity matrix. The actual values used for mass, m and damping, c are not very important as long as they can be adjusted to make the time constants of the system suitable for interactive use.

The system of equations is solved through time by approximating the time derivatives with finite differences as;

$$\begin{aligned} \frac{d^2\mathbf{X}(t)}{dt^2} &= \frac{\mathbf{X}(t+\Delta t) - 2\mathbf{X}(t) + \mathbf{X}(t-\Delta t)}{\Delta t^2} \\ \frac{d\mathbf{X}(t)}{dt} &= \frac{\mathbf{X}(t+\Delta t) - \mathbf{X}(t-\Delta t)}{2\Delta t} \end{aligned} \quad 4.37$$

The integration is made implicit by using the forward values of $\mathbf{X}(t+\Delta t)$ in the stiffness term \mathbf{KX} . Current values of $\mathbf{X}(t)$ are used in evaluating $\mathbf{F}(t)$.

The unknowns of the system are now $\mathbf{X}(t+\Delta t)$ and the knowns include $\mathbf{X}(t)$ and $\mathbf{X}(t-\Delta t)$ as well as $\mathbf{F}(t)$. Placing the unknowns on the left and the knowns on the right yields a new system of equations in the form

$$\mathbf{A} \mathbf{X}(t+\Delta t) = \mathbf{B}(\mathbf{F}(t), \mathbf{X}(t), \mathbf{X}(t-\Delta t)) \quad 4.38$$

$$\text{where } \mathbf{A} = \left[\frac{\mathbf{M}}{\Delta t^2} + \frac{\mathbf{C}}{2\Delta t} + \mathbf{K} \right] \text{ and } \mathbf{B} = \mathbf{F}(t) + \frac{2\mathbf{M}}{\Delta t^2} \mathbf{X}(t) + \left[\frac{\mathbf{C}}{2\Delta t} - \frac{\mathbf{M}}{\Delta t^2} \right] \mathbf{X}(t-\Delta t)$$

Because of the diagonal forms of the mass and damping matrices, the \mathbf{A} matrix is made by adding a constant value, $(c/2\Delta t + m/\Delta t^2)$, to each diagonal element of the \mathbf{K} matrix. The effect of this on the eigenvalues of the \mathbf{K} matrix is to shift all of them to the left in the complex plane by a constant amount. This tends to reduce the time constants of the system, guarantee that there are no zero value poles, and increase the damping of the system.

During the simulation, time proceeds in increments of Δt . At each time step the forcing vector, \mathbf{B} , is evaluated given the current values of $\mathbf{X}(t)$ and $\mathbf{X}(t-\Delta t)$ and $\mathbf{F}(t)$ calculated from the applied loads. As will be discussed in the next section the forcing vector \mathbf{B} , will also contain terms that enforce geometric constraints. This forcing vector is then used with the \mathbf{A} matrix to calculate the position at time $\mathbf{X}(t+\Delta t)$. At the end of each iteration the current and past positions are updated as;

$$\mathbf{X}(t) \quad \text{becomes the next } \mathbf{X}(t-\Delta t) \quad 4.39$$

$$\mathbf{X}(t+\Delta t) \quad \text{becomes the next } \mathbf{X}(t) \quad 4.40$$

and the process is repeated ad infinitum.

During the design session the above time integration proceeds continuously while the user input is taken as changes made between time steps.

The matrix \mathbf{A} is positive definite, symmetric and banded and as such is amenable to Gaussian elimination. Gaussian elimination is a direct and inexpensive way to solve the

$AX = B$ equations for X without directly inverting the A matrix. The Gaussian Elimination idea is to divide the system A matrix into $A = LU$ the product of a lower and an upper triangular matrix. The L and U matrices can then be used to solve the $AX = B$ problem in two sequential steps each requiring the solution of a triangular system of equations.

$$\begin{array}{ll} \text{With} & AX = B \text{ or } LUX = B \\ \text{then use} & UX = C \text{ and } LC = B \end{array} \quad 4.41$$

Triangular systems can be solved in $O(n(n+1)/2)$ steps where n is the size of the matrix L and U . Solving two sets of triangular systems can be accomplished in $O(n^2+n)$ steps. When the matrix is banded the solution time becomes $O(2*n*m)$ where m is the band width of the system. Narrow bandwidths greatly reduce the cost of computations. For the deformable curve elements the bandwidth of the matrix is 4. Given a reasonably complex curve with 50 elements (100 degrees of freedom) the difference in solution times with and without taking advantage of banding is the difference between $O(10,000)$ and $O(400)$ floating point operations for each time iteration.

The factorization of the A matrix is found by elimination where L contains the multipliers used in the elimination and 1s on the diagonal. Because A is originally symmetric the factorization can be rewritten in the form $A = LDL^T$ with $U = DL^T$ where D is a diagonal matrix. Since A is symmetric and DL^T is upper triangular only the diagonal and super-diagonal rows of this matrix need be stored.

An efficient implementation of Gaussian elimination separates the factorization and solution steps. The A matrix is factored and saved as LDL^T whenever the A matrix is modified by the user. This happens only when the user modifies the α or β material constants, adds or removes a constraint, or adds or moves any of the nodes or elements of the parametric geometry, a relatively infrequent event. Solving the $AX = B$ system of equations happens once every time step. First the B matrix is made from terms applied by the loads, constraints and dynamics of the system. Then B is transformed into C by solving $C = L^{-1}B$, and finally X is found by solving $X = (DL^T)^{-1}C$.

Because the elements of B and X are vectors in three space this process is done three times, once for x , y , and z , using the same A matrix in all cases. If the equations for x , y

and z were not identical then the size and the band width of the A matrix would have to be tripled. Solution times would become of the $O(18*n*m)$ an increase of almost an order of magnitude.

4.4 A Deformable Surface Primitive

The objective of this section is to develop the finite element primitive that is used to solve for surface shape in the ShapeWright program. The finite element method is selected over finite difference approach to support continuous representations of shape. The reduced transformation constraint technique will be used to enforce the geometric constraints needed to support the skinning, and sculpting phases of the ShapeWright modeling paradigm. The class of topological structures that will be considered at this time will be all shapes that can be approximated by arbitrarily meshed polygons in the parameter space. This will be accomplished by making the basic finite element surface primitive a triangle.

The decision to use the reduced transformation constraint technique to enforce geometric constraints is due to the fact that this approach is the numerically best behaved. Ideally, the more geometric constraints that a shape design package can support the better. For instance if a surface could be made to slide over a point in space like some kind of two dimensional pulley, then these surface could be applied to the lightly scattered data problem of interpolating a set of measured points in space. In this work the problem statement is limited to implementing the three steps of the ShapeWright shape design paradigm. The complete set of geometric constraints needed to support ShapeWright has been defined to be free, pinned, hinged, and fixed edges. Although a general constraint scheme that could define and enforce any kind of geometric constraint would be ideal, this work only considers schemes that can support all of the ShapeWright constraints.

The constraints on the form of the ShapeWright triangular shape functions are set by the above design decisions. These include,

- o C^1 continuity between elements
- o Geometric Constraints defined as linear functions of the shape function weights
- o Shape function weights defined to be geometric properties at a set of 'nodes'
- o Edge geometry completely defined by the weights of the nodes on that edge.

4.4.a) Barycentric Coordinates in 2 Dimensions

Barycentric coordinates in 2 dimensions are a natural choice for defining functions that apply to triangular domains. Barycentric coordinates are an alternative to Cartesian coordinates for defining locations in the plane. Barycentric coordinates are interesting because their geometric interpretation is based on the triangle. In fact, the mapping from Barycentric to Cartesian coordinates is defined by the locations of the vertices of a mapping triangle. The advantage of writing a function in terms of Barycentric coordinates is that the function can be mapped to any shaped triangle simply by changing the vertex locations of the mapping triangle. For the finite element to be developed in this section, it means that the stiffness matrix will be derived only once and be applicable to all triangle shapes.

Barycentric coordinates are defined by the mapping;

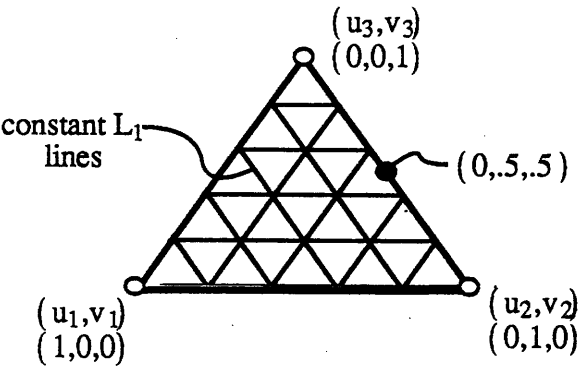
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix}$$


Figure 4.4) Barycentric coordinates in 2 dimensions

Where a point location in the uv plane is given as $[u,v]$ in Cartesian coordinates or $[L_1, L_2, L_3]$ in Barycentric Coordinates. The locations $[u_1, u_2]$, $[u_2, v_2]$, and $[u_3, v_3]$ define the vertex locations of the mapping triangle as shown in Figure 4.4. The Barycentric Coordinates, $[L_1, L_2, L_3]$ use three values to map locations over the entire plane. The extra degree of freedom is eliminated by the constraint, $1 = L_1 + L_2 + L_3$, which is the bottom row equation of the Barycentric mapping given above.

Remark 1: When $L_1 > 0$ and $L_2 > 0$ and $L_3 > 0$ then the equivalent point $[u,v]$ is inside the triangle.

Remark 2: A line of constant L_1 value will be parallel to the edge opposite the v_1 vertex. The line $L_1 = 0$ is coincident with the opposite edge. The same is true of lines of constant L_2 and L_3 .

Remark 3: The vertices of the mapping triangle given in Barycentric coordinates are $[1,0,0]$, $[0,1,0]$ and $[0,0,1]$.

Remark 4: Vectors are defined as the difference between points. The vector between two points, P_1 and P_0 will be given as;

$$P_1 - P_0 = \begin{bmatrix} L_{11} - L_{10} \\ L_{21} - L_{20} \\ L_{31} - L_{30} \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \text{note: } a + b + c = 0 \quad 4.42$$

The Barycentric mapping can always be inverted as long as the three vertices of the mapping triangle are not co-linear. This inverted relationship can be written as;

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \frac{1}{2\Delta} \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} 1 \\ u \\ v \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \equiv \mathcal{I} \quad 4.43$$

$$\text{and } a_i = u_j v_m - u_m v_j \Rightarrow \sum_i \frac{a_i}{2\Delta} = 1$$

$$b_i = u_j - u_m \Rightarrow \sum_i b_i = 0$$

$$c_i = v_m - v_j \Rightarrow \sum_i c_i = 0$$

All nine elements can be found by solving the above three equations three times, with $[i=1, j=2, m=3]$, $[i=2, j=3, m=1]$ and $[i=3, j=1, m=2]$.

Remark 1: The \mathcal{I} matrix completely characterizes the shape of the triangle up to rigid body displacements.

Remark 2: 2Δ = twice the area of the triangle and it can be found as $\sum_i a_i = 2\Delta$.

The ShapeWright energy minimum principle for shape, w is a function of its Cartesian partial derivatives.

$$I = I(w_u, w_v, w_{uu}, w_{uv}, w_{vv}) \quad 4.44$$

Fortunately, since the Barycentric mapping is a linear function, differentiating Barycentric functions in terms of Cartesian coordinates is very simple. The chain rule for first order derivatives yields;

$$\begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \begin{bmatrix} \frac{\partial L_1}{\partial u} & \frac{\partial L_2}{\partial u} & \frac{\partial L_3}{\partial u} \\ \frac{\partial L_1}{\partial v} & \frac{\partial L_2}{\partial v} & \frac{\partial L_3}{\partial v} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial L_1} \\ \frac{\partial}{\partial L_2} \\ \frac{\partial}{\partial L_3} \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} \frac{\partial L_1}{\partial u} & \frac{\partial L_2}{\partial u} & \frac{\partial L_3}{\partial u} \\ \frac{\partial L_1}{\partial v} & \frac{\partial L_2}{\partial v} & \frac{\partial L_3}{\partial v} \end{bmatrix} \equiv J_1^{-1} \quad 4.45$$

The inverse Jacobian matrix, J_1^{-1} , is constant because of the linear relationship between the Barycentric and Cartesian coordinates and is given by;

$$J_1^{-1} = \frac{1}{2\Delta} \begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & v_2 & v_3 \end{bmatrix} \quad \text{and} \quad \nabla_L = \begin{bmatrix} \frac{\partial}{\partial L_1} \\ \frac{\partial}{\partial L_2} \\ \frac{\partial}{\partial L_3} \end{bmatrix} \quad 4.46$$

Applying the chain rule a second time, as in $\frac{\partial^2}{\partial u^2} = \frac{\partial}{\partial u} \left(\frac{\partial}{\partial u} \right)$ yields the relationship for second order differentials;

$$\begin{aligned}
 \begin{bmatrix} \frac{\partial^2}{\partial u^2} \\ \frac{\partial^2}{\partial v^2} \\ \frac{2\partial^2}{\partial u \partial v} \end{bmatrix} &= \frac{1}{2\Delta^2} \begin{bmatrix} b_1b_1 & 2b_1b_2 & b_2b_2 & 2b_2b_3 & b_3b_3 & 2b_3b_1 \\ c_1c_1 & 2c_1c_2 & c_2c_2 & 2c_2c_3 & c_3c_3 & 2c_3c_1 \\ 2b_1c_1 & 2\left(\begin{smallmatrix} b_1c_2 \\ + \\ b_2c_1 \end{smallmatrix}\right) & 2b_2c_2 & 2\left(\begin{smallmatrix} b_2c_3 \\ + \\ b_3c_2 \end{smallmatrix}\right) & 2b_3c_3 & 2\left(\begin{smallmatrix} b_3c_1 \\ + \\ b_1c_3 \end{smallmatrix}\right) \end{bmatrix} \begin{bmatrix} \frac{\partial^2}{\partial L_1 \partial L_1} \\ \frac{\partial^2}{\partial L_1 \partial L_2} \\ \frac{\partial^2}{\partial L_2 \partial L_2} \\ \frac{\partial^2}{\partial L_2 \partial L_3} \\ \frac{\partial^2}{\partial L_3 \partial L_3} \\ \frac{\partial^2}{\partial L_3 \partial L_1} \end{bmatrix} \quad 4.47 \\
 &= \frac{J_2^{-1}}{2\Delta^2} \nabla_L^2
 \end{aligned}$$

In addition to the Cartesian derivatives, we will need derivatives of the Barycentric functions tangent and normal to the edges of the mapping triangle. The direction of the tangent vector, \mathbf{t} , to an edge is defined so that its associated normal, \mathbf{n} , has a positive v component. The normal is always orthogonal to the tangent vector and rotated in the counterclockwise direction so that the cross product, $\mathbf{t} \times \mathbf{v}$, is positive. The angle between the u axis and a tangent vector, γ_{ij} , for edge e_{ij} will always obey the relation,

$$0 \leq \gamma_{ij} < \frac{\pi}{2} \quad \text{or} \quad \frac{3\pi}{2} \leq \gamma_{ij} < 2\pi \quad 4.48$$

Figure 4.5 shows the relation between the mapping triangle, positive normals and the tangent angles.

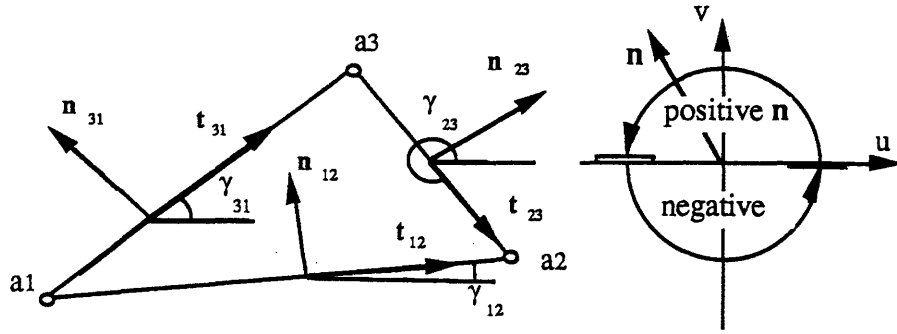


Figure 4.5)
and normals of a Barycentric mapping triangle

Edge tangents

The orthonormal transformation from the Cartesian coordinates to the tangent-normal coordinates is given by;

$$\begin{bmatrix} \mathbf{t}_{ij} \\ \mathbf{n}_{ij} \end{bmatrix} = \begin{bmatrix} \cos(\gamma_{ij}) & \sin(\gamma_{ij}) \\ -\sin(\gamma_{ij}) & \cos(\gamma_{ij}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad 4.49$$

Derivatives of the Barycentric coordinates in the direction of the tangents and the normals along edge_{ij} can be found by applying the chain rule, yet again as;

$$\begin{aligned} \begin{bmatrix} \frac{\partial}{\partial \mathbf{t}_{ij}} \\ \frac{\partial}{\partial \mathbf{n}_{ij}} \end{bmatrix} &= \begin{bmatrix} \frac{\partial \mathbf{t}_{ij}}{\partial u} & \frac{\partial \mathbf{t}_{ij}}{\partial v} \\ \frac{\partial \mathbf{n}_{ij}}{\partial u} & \frac{\partial \mathbf{n}_{ij}}{\partial v} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \begin{bmatrix} \cos(\gamma_{ij}) & \sin(\gamma_{ij}) \\ -\sin(\gamma_{ij}) & \cos(\gamma_{ij}) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} \\ &= \mathbf{J}_{\gamma_{ij}} \begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \mathbf{J}_{\gamma_{ij}} \mathbf{J}_1^{-1} \nabla_L \end{aligned} \quad 4.50$$

Due to the linear nature of the Barycentric mapping the matrix, $\mathbf{J}_{\gamma_{ij}} \mathbf{J}_1^{-1}$, is constant depending only on the values of the terms of the \mathbf{J} matrix. This matrix is given the name, $\mathbf{T}_m \equiv \mathbf{J}_{\gamma_{ij}} \mathbf{J}_1^{-1}$ so that;

$$\begin{bmatrix} \frac{\partial}{\partial \mathbf{t}_{ij}} \\ \frac{\partial}{\partial \mathbf{n}_{ij}} \end{bmatrix} = \mathbf{T}_m \nabla_L \quad 4.51$$

The advantage of using Barycentric coordinates for defining shape functions over triangular regions is best shown through the use of examples. Consider the three following simple functions and their properties as follows. The value w , is mapped on the z axis and the x and y axes span the L_1, L_2, L_3 plane

$$1.) w(L_1, L_2, L_3) = L_1 \quad 4.52$$

The shape, w , is a flat plane. Its value at vertex 1 is 1 and at vertices 2 and 3 is zero. The shape, $w(L_1=0)$, along edge₂₃ where $L_1 = 0$ is also zero.

$$2.) w(L_1, L_2, L_3) = L_2^2 L_3 \quad \text{and so} \quad \nabla_L w^T = \begin{bmatrix} 0 & 2L_2 L_3 & L_2^2 \end{bmatrix} \quad 4.53$$

The shape, w , and its normal and tangent derivatives are zero at all three vertices except for the non-zero normal derivative at vertex 2. The shape geometry along edge₃₁ and edge₁₂ are zero implying that the tangent derivatives along those edges are zero; $w_{31} = w_{12} = 0$. The value of $\nabla_L w(L_2=0) = 0$ along edge₃₁ implying that the normal derivative is zero along that entire edge as well.

$$3.) w(L_1, L_2, L_3) = L_1 L_2 L_3 \quad \text{and so} \quad \nabla_L w^T = \begin{bmatrix} L_2 L_3 & L_1 L_3 & L_1 L_2 \end{bmatrix} \quad 4.54$$

The shape, w , is zero along all three edges implying that all $w_{ij} = 0$ on all the edges. At the vertices $w_{nij} = 0$ but will vary along each edge. This function is known as the bubble function, since its addition to an existing shape will only change the interior geometry and will not effect the edge geometry or the surface tangents at the vertex locations.

4.4.b) Triangular Shape Functions - 9 Degrees of Freedom

In this section the shape approximation, $w^h = \sum_i x_i \Phi_i$, for the ShapeWright triangular shape function is given. The approach taken to define the Φ_i shape functions is to find a piecewise polynomial function that will guarantee C^1 continuity between triangles and will support the geometric constraints of pinned, hinged and fixed edges as a linear function of the x_i weights.

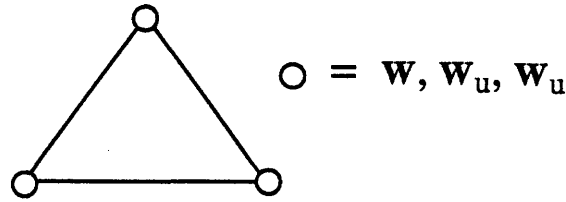


Figure 4.6) Triangular element with 3 nodes and 9 degrees of freedom

The finite element strategy is to assign nodes to the triangular element and to have the x_i weights be equal to geometric properties at that node. As a first try, consider the 9 degree of freedom triangle shown in Figure 4.6. The triangle is given three nodes, one at each vertex. Three degrees of freedom are assigned to each node; its position and tangents in the u and v directions. Zienkiewicz first published shape functions for such a triangle in 1969 as the preferred element for solving linear plate bending problems. Generally speaking, since the position and tangents are vectors in three space, this triangle really has 3×9 or 27 degrees of freedom. However, since the energy functional has been constrained so that the equations for the x , y , and z directions are identical and independent, the triangle can be developed as a 9 degree of freedom element to be applied once in each of the three directions. The degrees of freedom for one triangle element are ordered as;

$$\mathbf{a}^e{}^T = [w_1 \quad w_{u1} \quad w_{v1} \quad w_2 \quad w_{u2} \quad w_{v2} \quad w_3 \quad w_{u3} \quad w_{v3}] \quad 4.55$$

where $w_1 = w(u_1, v_1)$ and $w_{u1} = w_u(u_1, v_1)$ etc.

The advantage of defining the degrees of freedom of the system as geometric properties is that it simplifies the generation of the associated set of shape functions. We require of each shape function, Φ_i , that it be 1 for the geometric property at the node corresponding to the i th degree of freedom for the triangle and be zero for all other geometric properties. For example, the shape function Φ_1 corresponding to the shape position at the first vertex,

needs to be 1 at vertex 1 and 0 at vertices 2 and 3. Additionally the value of $\Phi_{1u} = \frac{\partial \Phi_1}{\partial u}$ and

$\Phi_{1v} = \frac{\partial \Phi_1}{\partial v}$ at the three vertices needs to be zero.

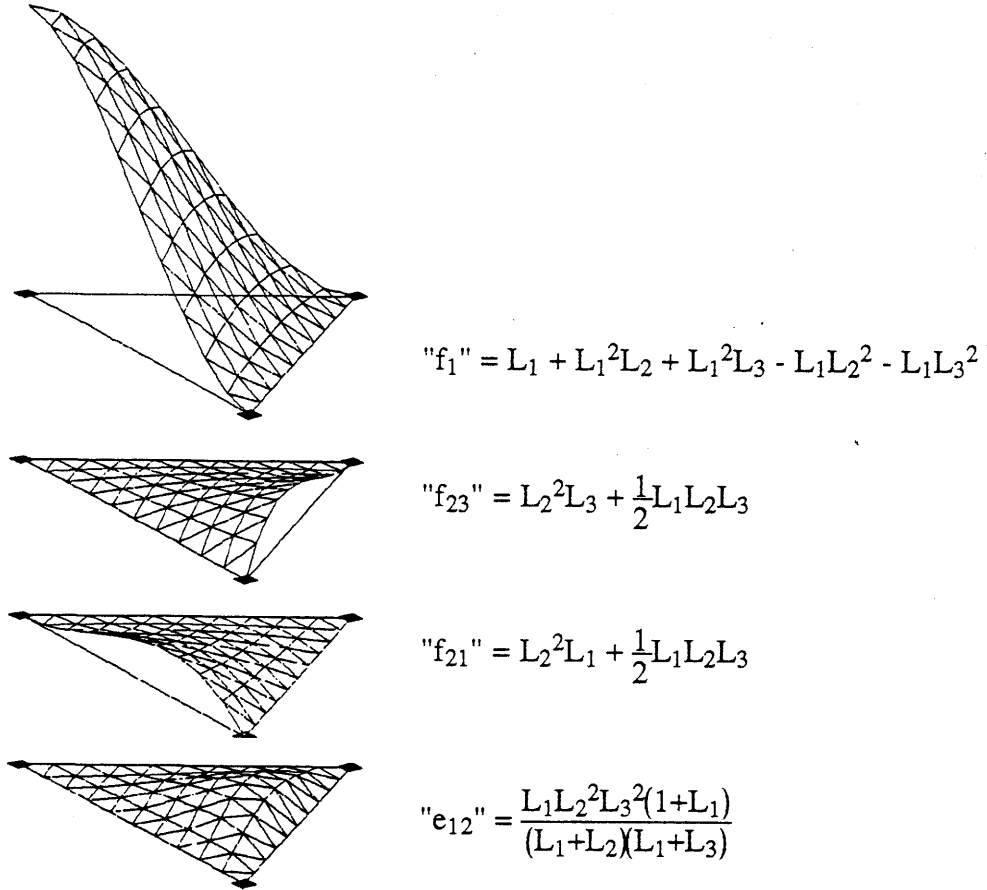


Figure 4.7) The 'e' and 'f' functions used to build the shape functions

The shape functions for the 9 degree of freedom triangle developed by Zienkiewicz are based on the 'f' and 'e' functions shown in Figure 4.7 and are as expressed as;

$$\mathbf{w}^h = \mathbf{N}^9(L_1, L_2, L_3, \mathbf{f}) \mathbf{a}^e \quad 4.56$$

$$\text{where } \mathbf{N}^9 = \begin{bmatrix} \mathbf{N}_1^9 & \mathbf{N}_2^9 & \mathbf{N}_3^9 \end{bmatrix}$$

$$\mathbf{N}_1^{9T} = \begin{bmatrix} L_1 + L_1^2L_2 + L_1^2L_3 - L_1L_2^2 - L_1L_3^2 \\ c_3(L_1^2L_2 + .5L_1L_2L_3) - c_2(L_1^2L_3 + .5L_1L_2L_3) \\ -b_3(L_1^2L_2 + .5L_1L_2L_3) + b_2(L_1^2L_3 + .5L_1L_2L_3) \end{bmatrix}$$

$$= \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ c_3 f_{12} - c_2 f_{13} \\ -b_3 f_{12} + b_2 f_{13} \end{bmatrix}$$

The symmetry of the triangle in Barycentric coordinates can be exploited to define the shape functions for the 2nd and 3rd nodes in terms of the first node shape functions. To generate N_2^9 use the above equations but add a 1 to each index, so that 1 \Rightarrow 2 and 2 \Rightarrow 3 and the special case 3 \Rightarrow 1. Adding another 1 to each index generates the shape functions for N_3^9 .

Along the edges the shape varies cubically. For example, along edge₂₃, let $L_1 = 0$, and $L_2 = (1 - L_3)$. Substituting these values into the shape relation yields a cubic equation in L_3 that only depends on the nodal data at nodes₂ and nodes₃.

The $\nabla_L \mathbf{w}^h$ functions, $\nabla_L \mathbf{w}^h = [\mathbf{w}_{L_1}^h \quad \mathbf{w}_{L_2}^h \quad \mathbf{w}_{L_3}^h]$ are defined in pieces by;

$$\begin{aligned}
 N_{1L_1}^9{}^T &= \begin{bmatrix} 1 + 2L_1L_2 + 2L_1L_3 - L_2^2 - L_3^2 \\ c_3(2L_1L_2 + .5L_2L_3) - c_2(2L_1L_3 + .5L_2L_3) \\ -b_3(2L_1L_2 + .5L_2L_3) + b_2(2L_1L_3 + .5L_2L_3) \end{bmatrix} \\
 N_{1L_2}^9{}^T &= \begin{bmatrix} L_1^2 - 2L_1L_2 \\ c_3(L_1^2 + .5L_1L_3) - c_2(.5L_1L_3) \\ -b_3(L_1^2 + .5L_1L_3) + b_2(.5L_1L_3) \end{bmatrix} \\
 N_{1L_3}^9{}^T &= \begin{bmatrix} L_1^2 - 2L_1L_3 \\ c_3(.5L_1L_2) - c_2(L_1^2 + .5L_1L_2) \\ -b_3(.5L_1L_2) + b_2(L_1^2 + .5L_1L_2) \end{bmatrix}
 \end{aligned} \tag{4.57}$$

where symmetry once again builds the partials of the N_2^9 and N_3^9 functions by incrementing the subscripts. Be sure to increment subscripts on both sides of the equal sign so that $N_{1L_1}^9$ goes to $N_{2L_2}^9$.

The normal and tangent derivatives are calculated as linear sums of $\mathbf{w}_{L_1}^h$, $\mathbf{w}_{L_2}^h$ and $\mathbf{w}_{L_3}^h$. Checking the geometry along edge₂₃, by setting $L_1 = 0$ and $L_2 = (1 - L_3)$, reveals the derivatives to vary parabolically depending on the values at all three nodes, nodes₁, nodes₂ and nodes₃.

The properties of the N^9 functions can be examined by looking only at the N_1^9 functions and trusting symmetry to extend the results to the two other nodes. The major requirement

on the shape functions is that they can set their associated degrees of freedom independently. To check this we evaluate the following;

$$\begin{aligned} \phi_1(1,0,0) &= 1 & \frac{\partial \phi_1(1,0,0)}{\partial u} &= b_1 + b_2 + b_3 = 0 \\ \phi_1(0,1,0) &= 0 \quad \text{and} \\ \phi_1(0,0,1) &= 0 & \frac{\partial \phi_1(0,1,0)}{\partial u} &= \frac{\partial \phi_1(0,0,1)}{\partial u} = 0 \end{aligned} \quad 4.58$$

Evaluating the values of w_u^h and w_v^h at node₁ we find;

$$\begin{aligned} \begin{bmatrix} w_u^h \\ w_v^h \end{bmatrix}_{(1,0,0)} &= \frac{1}{2\Delta} \begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} w_{L_1}^h \\ w_{L_2}^h \\ w_{L_3}^h \end{bmatrix}_{(1,0,0)} \\ &= \frac{1}{2\Delta} \begin{bmatrix} (b_2c_3 - b_3c_2) w_{u1} \\ (c_3b_2 - c_2b_3) w_{v1} \end{bmatrix} = \begin{bmatrix} w_{u1} \\ w_{v1} \end{bmatrix} \end{aligned} \quad 4.59$$

And so these functions act to directly set the 9 degrees of freedom of the triangle independently as required. However, they do not enforce C^1 continuity between triangles. To see this, examine the tangent and normal derivatives at the mid-edge locations between nodes. The values of $\nabla_L w^h(.5, .5, 0)$ are ;

$$\nabla_L w^h(.5, .5, 0) = \begin{bmatrix} 1.25 & .5c_3 & -.5c_3 & -.25 & -.25c_3 & .25b_3 & 0 & 0 & 0 \\ -.25 & .25c_3 & -.25b_3 & 1.25 & -.5c_3 & .5b_3 & 0 & 0 & 0 \\ .25 & \frac{1}{8}(c_3-c_2) & \frac{1}{8}(b_2-b_3) & .25 & \frac{1}{8}(c_1-c_3) & \frac{1}{8}(b_3-b_1) & .5 & \frac{1}{8}(c_2-c_1) & \frac{1}{8}(b_1-b_2) \\ & -.25c_2 & +.25b_2 & & +.25c_1 & -.25b_1 & & & \end{bmatrix} \mathbf{a}^e \quad 4.60$$

The tangent and normal derivatives at the mid-edge point are given by;

$$\begin{bmatrix} w_{t12}^h \\ w_{n12}^h \end{bmatrix}_{(.5, .5, 0)} = \mathbf{T}_{tn} \nabla_L w^h(.5, .5, 0) \quad 4.61$$

The terms of the matrix T_{tn} are all constants depending only on the Barycentric transformation defined by the matrix \mathcal{B} . The normal derivative, w_{n12}^h , will be a function of w_3 , w_{u3} and w_{v3} , as well as the nodal data at nodes $node_1$ and $node_2$. This shows that these functions can not be C^1 continuous. The reasoning is thus, consider two triangles that are adjoined along the edge₁₂ as shown in Figure 4.8. Assume for the time being that they are currently C^1 continuous. The normal derivative along edge₁₂ in triangle 1 depends on the data of node₃ as shown above, but node₃ does not effect the normal derivative of edge₁₂ in element 2. Changing the value of the data at node₃ while fixing the data of node₄ will cause the normal derivative in element 1 to change making the normal derivative across edge₁₂ discontinuous. In general, any property that must be continuous across a finite element boundary must be computed only by the data of the nodes on that boundary.

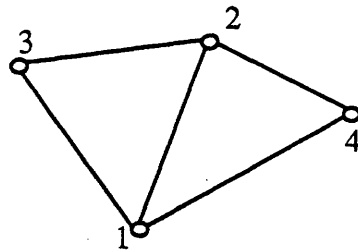


Figure 4.8) Adjoining triangles in a finite element mesh

4.4.c) Triangular Shape Functions - 12 Degrees of Freedom

In this section the shape functions for a C^1 continuous 12 degree of freedom triangle are developed as an extension to the N^9 shape functions. The 12 degree of freedom triangle is defined by adding three degrees of freedom, taken as the normal derivative at the mid-side of each edge, to the 9 degree of freedom triangle as shown in Figure 4.9. In this manner it becomes possible to set both the shape and the normal derivative geometry along an edge as a function of the information in the nodes on that edge.

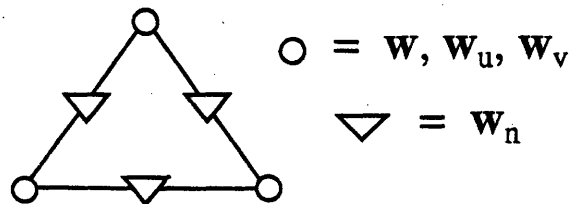


Figure 4.9) Triangular element with 12 degrees of freedom

Along each edge of the 9 degree of freedom triangle, shape varies cubically and the normal derivative varies parabolically. It requires four degrees of freedom to define the shape of a cubic function and three degrees of freedom to fix the shape of a parabolic function. It will require at least 7 degrees of freedom to be able to define both the shape and the normal derivative geometry along an edge. The 9 degree of freedom triangle has only six degrees of freedom on an edge and was incapable of supporting C^1 continuity.

The 'e' function introduced by Zienkiewicz has the excellent properties that its shape value along all three edges is zero and that the only non-zero shape edge derivative is in the normal direction of one of the three edges. Along that edge, the normal derivative varies parabolically with a maximum of 1/4 at the center of the edge. This 'e' function can be used to define the N^{12} shape functions of the 12 degree of freedom triangle.

Conceptually, the N^{12} shape functions can be built in two steps. First adjust the equations so that the old 9 degrees of freedom act to leave the normal derivatives at the mid-nodes zero while they vary. The second step is to add in weighted 'e' functions so that the degrees of freedom at the new nodes directly set the mid-edge normal derivatives.

The effect of the current 9 degrees of freedom on the normal derivative at the mid-edges can be found by evaluating the functions at the new nodes as;

$$\begin{bmatrix} w_{n12}^h(.5,.5,0) \\ w_{n23}^h(0,.5,.5) \\ w_{n31}^h(.5,0,.5) \end{bmatrix} \equiv \mathbf{Z} \mathbf{a}^e \quad 4.62$$

The matrix \mathbf{Z} is constant depending only on the terms of the \mathbf{L} matrix defining the Barycentric mapping. A 9 degree of freedom set of shape functions can be written that independently set their 9 degrees of freedom while leaving the mid-side normal derivatives at zero as;

$$\mathbf{w}_{\text{zero mid-edge normal derivatives}}^h = (\mathbf{N}^9 - 4 [\mathbf{e}_{12} \mathbf{e}_{23} \mathbf{e}_{31}] \mathbf{Z}) \mathbf{a}^e \quad 4.63$$

The 12 degree of freedom shape functions can now be generated by adding the appropriately weighted 'e' functions to set the mid-side normal derivatives to equal the values of the mid-side nodal degrees of freedom.

$$\mathbf{w}^h = \begin{bmatrix} \mathbf{N}^9 & | & 0 & 0 & 0 \end{bmatrix} + 4 \begin{bmatrix} \mathbf{e}_{12} & \mathbf{e}_{23} & \mathbf{e}_{31} \end{bmatrix} \begin{bmatrix} \mathbf{Z} & | & 1 & 0 & 0 \\ & | & 0 & 1 & 0 \\ & | & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}^e \\ \mathbf{w}_{n12} \\ \mathbf{w}_{n23} \\ \mathbf{w}_{n31} \end{bmatrix} \quad 4.64$$

The new set of shape functions can now be written as;

$$\mathbf{w}^h = \mathbf{N}^{12} \hat{\mathbf{a}}^e \quad \text{defining} \quad 4.65$$

$$\hat{\mathbf{a}}^e \equiv \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_{u1} & \mathbf{w}_{v1} & \mathbf{w}_2 & \mathbf{w}_{u2} & \mathbf{w}_{v2} & \mathbf{w}_3 & \mathbf{w}_{u3} & \mathbf{w}_{v3} & \mathbf{w}_{n12} & \mathbf{w}_{n23} & \mathbf{w}_{n31} \end{bmatrix}$$

This builds a set of shape functions that has the required 7 degrees of freedom per edge so that both the shape and the normal derivative can be set independently in terms of the nodes on that edge. These triangles support C^1 continuity between adjoining elements. The triangle shape varies cubically and the normal derivative varies parabolically along each edge.

Interestingly, Zienkiewicz did not use the 'e' functions in this manner. He felt implementing a triangle that had different degrees of freedom at each node would be too cumbersome. Instead he used the 'e' functions to restrict the normal derivative to vary linearly along each edge. Now only the 6 degrees of freedom on each edge of the 9 degree of freedom triangle need be required to fix the edge and normal derivative geometry to support C^1 continuity. Unfortunately, he found that plate bending finite elements based on this constrained geometry approach worked poorly and ended up concluding that C^1 continuity was too excessive a restriction. He went on to explore the use of C^0 elements on the plate problem and started a research path that ended in developing the patch test to prove convergence for non-conforming elements. After experimentation we also found that this constrained 9 degree of freedom triangle made a poor shape function. It tended to be too flat in the center of the element and when highly curve would actually calculate surface normals in the wrong directions.

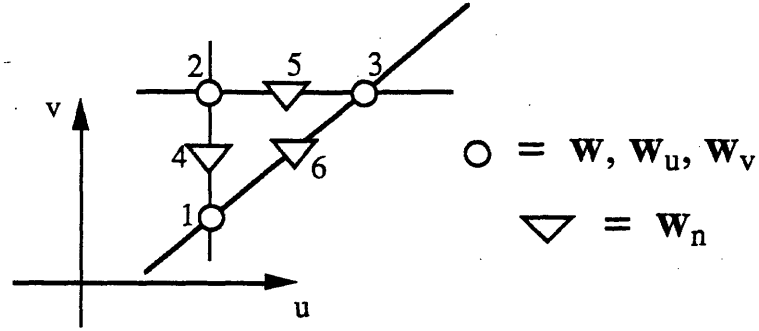


Figure 4.10) Partial mesh of aligned and orthogonal edges

The 'e' functions are rational polynomials and have a singularity in the evaluation of the cross derivative w_{uv} at the triangle vertices. This singularity is required of a triangle that enforces C^1 continuity without using 2nd order derivatives at the corner nodes. The reason for this has to do with the requirement that the geometry along an edge be defined in terms of the information of the nodes on that edge. Consider the partial mesh shown in Figure 4.10. The edges of the mesh are aligned with the u and v coordinate axes. The shape on edge₁₂, w_{12}^h , is completely defined by the information at node₁, node₂ and node₄.

Likewise, the shape of edge₂₃, w_{23}^h , is defined by node₂, node₃ and node₅. The derivative

values of shape along the edges, $\frac{\partial w_{12}^h}{\partial v}$ and $\frac{\partial w_{23}^h}{\partial u}$, will similarly only depend on the data from the same nodes.

$$w_{12}^h = w_{12}^h(\text{node}_1, \text{node}_2, \text{node}_4) \Rightarrow \frac{\partial w_{12}^h}{\partial v} = \frac{\partial w_{12}^h}{\partial v}(\text{node}_1, \text{node}_2, \text{node}_4) \quad 4.66$$

$$w_{23}^h = w_{23}^h(\text{node}_2, \text{node}_3, \text{node}_5) \Rightarrow \frac{\partial w_{23}^h}{\partial u} = \frac{\partial w_{23}^h}{\partial u}(\text{node}_2, \text{node}_3, \text{node}_5)$$

A C^1 surface that has no singularities in the second order derivatives will have the property

that at any point $\frac{\partial^2 \mathbf{w}_{12}^h}{\partial v \partial u} = \frac{\partial^2 \mathbf{w}_{23}^h}{\partial u \partial v}$. Calculate the cross derivatives for both of the above expressions and evaluate them at the common vertex at node₂.

$$\frac{\partial^2 \mathbf{w}_{12}^h(0,1,0)}{\partial v \partial u} = \frac{\partial \mathbf{w}_{12}^h(0,1,0)}{\partial v}(\text{node}_1, \text{node}_2, \text{node}_4) \quad 4.67$$

$$\frac{\partial^2 \mathbf{w}_{23}^h(0,1,0)}{\partial u \partial v} = \frac{\partial \mathbf{w}_{23}^h(0,1,0)}{\partial u}(\text{node}_2, \text{node}_3, \text{node}_5)$$

In general, these two cross derivatives calculated at the same point will not be the same. If these two cross derivatives happen to be the same, then the information at node₂ and node₄ could always be changed while fixing the values at node₂, node₃ and node₅ to make the two terms unequal. As long as the cross derivatives are calculated and not set by a unique degree of freedom at each vertex this will be a problem. Although the derivatives are defined and are continuous everywhere on the surface, the twist derivatives will not be defined at the vertices.

To build C^1 surfaces two approaches can be taken, include higher order derivatives as degrees of freedom at the vertices or introduce functions which support discontinuities in the higher order derivatives. In the CAGD literature it is common to include the twist vector at each vertex. This resolves the problem as long as the angles at all the edge intersections are the same such as in rectangular or parallel lined grids. But for arbitrarily meshed regions where any number of edges can meet in any number of angles at a point the entire second order derivative information of the curve will be required, \mathbf{w}_{uu} , \mathbf{w}_{uv} , and \mathbf{w}_{vv} . The alternative of including singular functions, like the 'e' function, to make continuous surfaces without explicitly including higher order derivatives was introduced into the CAGD community by Gregory and Barnhill as an extension to the Coons patch [Gregory_74].

It is felt that supporting arbitrary meshes is essential and that adding three extra degrees of freedom at each vertex would be excessive and not intuitive and thus the above shape functions are used as the basis of the ShapeWright surface finite element. In this work the singularity of the twist vector has presented no problem. The actual surfaces that have been generated are continuous and appear well behaved in the neighborhood of the element vertices.

4.4.d) Geometric Constraint Set with a 12 Degree of Freedom Element

The shape functions are required to be able to support the ShapeWright geometric constraint set as a linear function of the explicit degrees of freedom of the 12 degree of freedom triangle. This set of constraints consists of free, pinned, hinged, and fixed edges. The free and pinned constraints are supported directly because a free edge is unconstrained, and a pinned edge can be made by fixing the position degrees of freedom of a triangular element edge. All that remains is to show that the hinged and fixed edge conditions can be enforced with the current shape functions.

The hinged edge constraint is defined as the case where the edge geometry is fixed and the normal tangent along the edge is free to vary. The edge geometry of the 12 degree of freedom shape functions varies cubically along an edge. The geometry of a cubic curve can be completely specified by four parameters. When a cubic function is represented with Hermite polynomials those four parameters are the positions and the tangents at the ends of a curve segment. Hermite polynomials for cubic curves are discussed in detail in section 4.6.

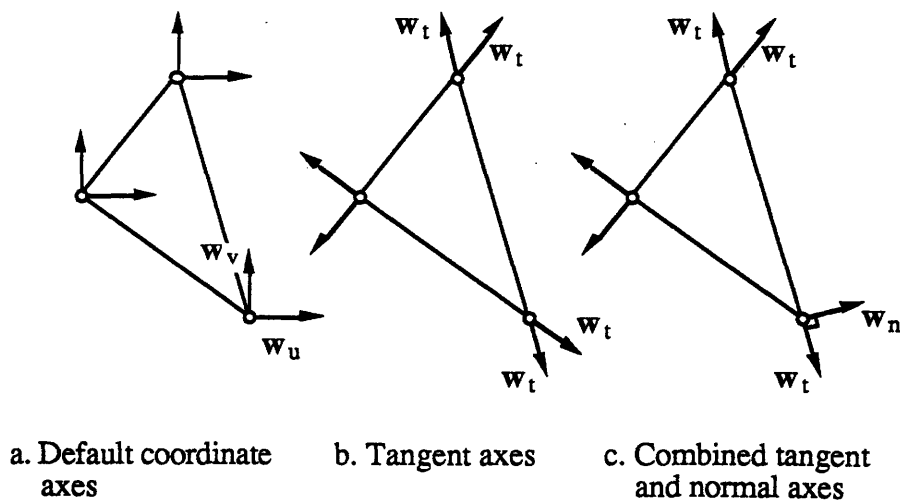


Figure 4.11) Alternative coordinate systems for vertex nodal tangents

To demonstrate that these shape functions can indeed support the hinged edge, all that needs to be done is to transform the current degrees of freedom, $\widehat{\mathbf{a}}^e$, into a new set, $\overline{\mathbf{a}}^e$, as shown in Figure 4.11b, that defines the degrees of freedom as;

$$\overline{\mathbf{a}}^e \equiv \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_{1t12} \\ \mathbf{w}_{1t31} \\ \mathbf{w}_2 \\ \mathbf{w}_{2t23} \\ \mathbf{w}_{2t12} \\ \mathbf{w}_3 \\ \mathbf{w}_{3t31} \\ \mathbf{w}_{3t23} \\ \mathbf{w}_{n12} \\ \mathbf{w}_{n23} \\ \mathbf{w}_{n31} \end{bmatrix} \quad 4.68$$

In the transformed coordinates, any combination or all of the edges can be made hinged by fixing the degrees of freedom as;

$$\text{hinged edge}_{12} \Rightarrow \text{Fixed } \mathbf{w}_1, \mathbf{w}_{1t12}, \mathbf{w}_2, \mathbf{w}_{2t12} \quad 4.69$$

$$\text{hinged edge}_{23} \Rightarrow \text{Fixed } \mathbf{w}_2, \mathbf{w}_{2t23}, \mathbf{w}_3, \mathbf{w}_{3t23}$$

$$\text{hinged edge}_{31} \Rightarrow \text{Fixed } \mathbf{w}_3, \mathbf{w}_{3t31}, \mathbf{w}_1, \mathbf{w}_{1t31}$$

The transformation of a nodal coordinate system from $\mathbf{w}_{2u}, \mathbf{w}_{2v}$ to $\mathbf{w}_{2t12}, \mathbf{w}_{2t23}$ can be made as follows;

$$\begin{aligned} \begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} &= \begin{bmatrix} \cos(\gamma_{12}) & -\sin(\gamma_{12}) \\ \sin(\gamma_{12}) & \cos(\gamma_{12}) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial t_{12}} \\ \frac{\partial}{\partial n_{12}} \end{bmatrix} \\ &= \begin{bmatrix} \cos(\gamma_{23}) & -\sin(\gamma_{23}) \\ \sin(\gamma_{23}) & \cos(\gamma_{23}) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial t_{23}} \\ \frac{\partial}{\partial n_{23}} \end{bmatrix} \end{aligned} \quad 4.70$$

Eliminating the normal derivatives from these equations yields;

$$\begin{bmatrix} 1 & \tan(\gamma_{12}) \\ \tan(\gamma_{23}) & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \begin{bmatrix} \cos(\gamma_{12}) & \tan(\gamma_{12})\sin(\gamma_{12}) \\ \cos(\gamma_{23}) & \tan(\gamma_{23})\sin(\gamma_{23}) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial t_{12}} \\ \frac{\partial}{\partial t_{23}} \end{bmatrix} \quad 4.71$$

and inverting yields;

$$\begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \frac{\begin{bmatrix} \sin(\gamma_{23}) & -\sin(\gamma_{12}) \\ -\cos(\gamma_{23}) & \cos(\gamma_{12}) \end{bmatrix}}{\sin(\gamma_{23})\cos(\gamma_{12}) - \sin(\gamma_{12})\cos(\gamma_{23})} \begin{bmatrix} \frac{\partial}{\partial t_{12}} \\ \frac{\partial}{\partial t_{23}} \end{bmatrix} \quad 4.72$$

This transformation allows the degrees of freedom of that node to be 'rotated' into its tangent coordinates. When edge₁₂ is perpendicular to edge₂₃ the transformation reduces to an orthonormal pure rotational transformation as it should. This transformation is valid as long as $\gamma_{23} \neq \gamma_{12}$, which is true as long as the three vertices of the element are not co-linear.

A fixed edge constraint is characterized by fixing both the shape and the surface normal along an edge. The surface normal is defined to be the unit vector in the direction of the cross product of w_u and w_v . Since the tangent plane at a point is unique, it can be specified by any two independent tangent vectors, not just w_u and w_v . Fixing the edge geometry also fixes the edge tangent derivative. The combination of fixing the edge tangent and normal derivatives acts to fix the surface normal along the edge as well. A fixed edge constraint may be characterized as fixing the geometry and the surface normal along an edge, but is enforced by fixing the geometry and the normal derivative along the edge.

The normal derivative of an edge of a 12 degree of freedom triangle varies parabolically. The normal derivative along the edge can be set independently if like, the hinged edge constraint, the vertex degrees of freedom are rotated to an appropriate coordinate system. The w_u , and w_v derivatives at a vertex node can be rotated so that they align with an edge's tangent and normal directions by the simple rotation matrix;

$$\begin{bmatrix} \frac{\partial}{\partial u} \\ \frac{\partial}{\partial v} \end{bmatrix} = \begin{bmatrix} \cos(\gamma_{ij}) & -\sin(\gamma_{ij}) \\ \sin(\gamma_{ij}) & \cos(\gamma_{ij}) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial t_{ij}} \\ \frac{\partial}{\partial n_{ij}} \end{bmatrix} \quad 4.73$$

The edge normal derivative can now be set independently of the rest of the geometry by rotating the vertices at both ends of an edge so that the three normal derivatives, w_{1n12} , w_{n12} , and w_{2n12} are now explicit. Any parabolic variation in the normal derivative can be made by adjusting these three values. A fixed edge is supported by fixing all 7 degrees of freedom along an edge, the positions and tangents at the vertex nodes are fixed to define the edge shape, and the normals at the vertex and mid-edge nodes are fixed to set the edge's normal derivative

In summary, the constraints for both hinged and fixed edges are enforced by transforming the tangent degrees of freedom at the nodes, $[w_u \ w_v]$, into the appropriate basis, $[w_{tij} \ w_{tik}]$ or $[w_{tij} \ w_{nij}]$ and then the explicit degrees of freedom are fixed as required. When two constrained edges come together at a vertex some overlapping of the constraints is achieved. For example, two fixed edges meeting at a vertex will both require that all three degrees of freedom of the vertex be fixed. Internally, the basis set that is used to fix the degrees of freedom of the system are not important. Fixing three degrees of freedom is fixing three degrees of freedom, and so in this case any basis set is appropriate. Figure 4.13 shows which degrees of freedom are fixed in which basis for all the combinations of intersecting constrained edges as shown in Figure 4.12.

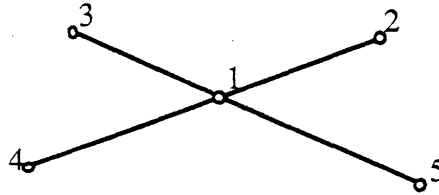


Figure 4.12) Partial mesh for intersecting constrained edges

		Edge ₁₂ Constraint State (see Figure 4.12)			
Edge ₁₅ con- strain state		Edge ₁₂ Free	Edge ₁₂ Pinned	Edge ₁₂ Hinged	Edge ₁₂ Fixed
	Edge ₁₅ Free	0	w_1	w_1, w_{t12}	w_1, w_{t12}, w_{n12}
	Edge ₁₅ Pinned	w_1	w_1	w_1, w_{t12}	w_1, w_{t12}, w_{n12}
	Edge ₁₅ Hinged	w_1, w_{t15}	w_1, w_{t15}	w_1, w_{t12}, w_{t15}	$w_1, w_{t12}, w_{n12} \text{ or } w_{t15}$
	Edge ₁₅ Fixed	w_1, w_{t15}, w_{n15}	w_1, w_{t15}, w_{n15}	$w_1, w_{t15}, w_{n15} \text{ or } w_{t12}$	$w_1, w_{n12} \text{ or } w_{t15}, w_{n15} \text{ or } w_{t12}$

Figure 4.13) Node constraint state due to edge constraints (see Figure 4.12)

In normal practice the user is not required to set the numerical values of these constraints. Typically the user will freeze a shape, by selecting a free edge and constraining it to remain in its current configuration. Once an edge is fixed the user can incrementally adjust the values of the constraints. As shown in section 4.4, this modification of a constraint value can be done as quickly as sculpting with forces. In the system implementation to be discussed later this is fast enough to support real time animation of the surface modifications.

These constraints are all enforced using the reduced transformation constraint technique mentioned in section 4.4.a. Fortunately these transformations only couple the degrees of freedom of individual nodes together. More general linear constraints can be imagined that could couple all the degrees of freedom of a triangle together or even the degrees of freedom of adjacent triangles. The implementation can take advantage of this fact. The rotations can be applied to each node individually which is done by executing a 2x2 matrix multiplication instead of modifying the whole system matrix in one full matrix multiply.

Because of this the constrained transformed matrix equations can be generated in barely more time than some equivalent untransformed constrained system.

4.4.e) Triangular Surface Stiffness and Forcing Matrices

Now that the shape functions, N^{12} , element degrees of freedom, \hat{a}^e and the quadratic minimum principle of chapter 3 have been selected the actual generation of the element stiffness and forcing matrices becomes an exercise in algebra and calculus. The stiffness and forcing matrix integrals are evaluated approximately with Gaussian quadrature because the actual shape functions are too complicated to bother evaluating them analytically due to the rational 'e' functions.

The first step is to rewrite the original minimum principle as the sum of a set of integrals taken over each triangular element of the surface;

$$E_{\text{surface}} = \int_{\sigma} \mathbf{W}_b^T \bar{\beta} \mathbf{W}_b + \mathbf{W}_s^T \bar{\alpha} \mathbf{W}_s \, dudv = \sum_i \int_{\text{element}_i} \mathbf{W}_b^T \bar{\beta} \mathbf{W}_b + \mathbf{W}_s^T \bar{\alpha} \mathbf{W}_s \, dudv \quad 4.74$$

$$\text{Where } \mathbf{W}_b = \begin{bmatrix} \mathbf{W}_{uu} \\ \mathbf{W}_{vv} \\ 2\mathbf{W}_{uv} \end{bmatrix}, \quad \mathbf{W}_s = \begin{bmatrix} \mathbf{W}_u \\ \mathbf{W}_v \end{bmatrix}$$

The next step is to substitute the current shape approximation, $\mathbf{w}^h = \mathbf{N}^{12} \hat{a}^e$, for each element into each element integral;

$$E_{\text{element}_i}^h = \int_{\sigma_i} \hat{a}^{eT} \left(\mathbf{N}_b^{12T} \bar{\beta} \mathbf{N}_b^{12} + \mathbf{N}_s^{12T} \bar{\alpha} \mathbf{N}_s^{12} \right) \hat{a}^e \, dudv \quad 4.75$$

$$\text{Where } \mathbf{N}_b^{12} = \begin{bmatrix} \mathbf{N}_{uu}^{12} \\ \mathbf{N}_{vv}^{12} \\ 2\mathbf{N}_{uv}^{12} \end{bmatrix}, \quad \mathbf{N}_s^{12} = \begin{bmatrix} \mathbf{N}_u^{12} \\ \mathbf{N}_v^{12} \end{bmatrix}$$

The derivatives of the shape functions are evaluated as;

$$\begin{aligned} N_b^{12} &= J_2^{-1} \nabla_L^2 (N^{12}) & \nabla_L^2 (N^{12}) &\equiv N_{L_2}^{12} \\ N_s^{12} &= J_1^{-1} \nabla_L (N^{12}) & \text{where} & \nabla_L (N^{12}) \equiv N_{L_1}^{12} \end{aligned} \quad 4.76$$

The minimum principle written in terms of known matrices can now be written as;

$$E_i = \int_{\sigma_i} \widehat{\mathbf{a}}^T \left(N_{L_2}^{12T} J_2^{-T} \bar{\beta} J_2^{-1} N_{L_2}^{12} + N_{L_1}^{12T} J_1^{-T} \bar{\alpha} J_1^{-1} N_{L_1}^{12} \right) \widehat{\mathbf{a}} - 2 \mathbf{f}^T N^{12} \widehat{\mathbf{a}} \, du dv \quad 4.77$$

The element degrees of freedom, $\widehat{\mathbf{a}}$, can be taken out of the integral since they are constant over the integral domain of u and v yielding;

$$E_i = \widehat{\mathbf{a}}^T K_{\sigma_i} \widehat{\mathbf{a}} - 2 F_{\sigma_i} \widehat{\mathbf{a}} \quad 4.78$$

$$\text{with } K_{\sigma_i} = \int_{\sigma_i} N_{L_2}^{12T} J_2^{-T} \bar{\beta} J_2^{-1} N_{L_2}^{12} + N_{L_1}^{12T} J_1^{-T} \bar{\alpha} J_1^{-1} N_{L_1}^{12} \, du dv$$

$$F_{\sigma_i} = \int_{\sigma_i} \mathbf{f}^T N^{12} \, du dv$$

The remaining integrals define the element stiffness matrix, K_{σ_i} and the element forcing vector, F_{σ_i} . It is not necessary, or even advisable to take these integrals analytically. Consider any single term of the element stiffness matrix. It will be the integral of the product of any two of the shape functions, a rational polynomial of 6th order, up to 21 terms in the numerator and the denominator. Expanding the K_{σ_i} matrix multiplies in the above equation will require multiplying matrices of the following sizes together;

$$\begin{aligned} & [12 \times 6][6 \times 3][3 \times 3][3 \times 6][6 \times 12] \\ & + [12 \times 3][3 \times 2][2 \times 2][2 \times 3][3 \times 12] \end{aligned}$$

There will be 144 integrals to solve, each with up to $3 \times 2 \times 2 \times 3 + 6 \times 3 \times 3 \times 6 = 360$ terms for a total of more than 50,000 terms to integrate. Symmetry reduces this number considerably

and even with the aid of an on line symbolic manipulation package the analytic evaluation of these integrals will result in a form of equations inconvenient for implementation and prone to errors. The alternative is to approximate the integrals numerically with a form of quadrature. Here the integral of the function is approximated as the weighted sum of the values of function taken at a few locations;

$$\int g(u,v) \, dudv \approx \sum_i \delta_i g[u_i, v_i] \quad 4.79$$

Figure 4.14 lists the locations and the weights needed to integrate these element matrices.

Order	Fig.	Error	Points	Triangular Co-ordinates	Weights
Linear		$O(h^2)$	a	1/3,1/3,1/3	1
Quadratic		$O(h^3)$	a	1/2,1/2,0	1/3
			b	0,1/2,1/2	1/3
			c	1/2,0,1/2	1/3
Cubic		$O(h^4)$	a	1/3,1/3,1/3	-27/48
			b	.6,.2,.2	25/48
			c	.2,.6,.2	25/48
			d	.2,.2,.6	25/48
Quintic		$O(h^6)$	a	1/3,1/3,1/3	.22500,00000
			b	a ₁ ,b ₁ ,b ₁	.13239,41527
			c	b ₁ ,a ₁ ,b ₁	"
			d	b ₁ ,b ₁ ,a ₁	"
			e	a ₂ ,b ₂ ,b ₂	.12593,91805
			f	b ₂ ,a ₂ ,b ₂	"
			g	b ₂ ,b ₂ ,a ₂	"
			with		
			a ₁ = .0597158717		
			b ₁ = .4701420641		
a ₂ = .7974269853					
b ₂ = .1012865073					

Figure 4.14) Numerical integration formulae for triangles [Zienkiewicz_67]

In the actual implementation the above matrix integral is separated into components that depend either on the Barycentric coordinates, (L_1, L_2, L_3) , or the Barycentric transformation, terms derivable from \mathcal{B} . At run time all the elements that depend on (L_1, L_2, L_3) are evaluated and stored into memory once. During equation building, the actual triangles determine the values of the Barycentric Mapping and those components of the stiffness and forcing matrices that depend on terms in the \mathcal{B} matrix are evaluated and the

final element stiffness matrix is evaluated. Each element stiffness matrix remains valid as long as the shape of the triangle and the user material properties, α and β are not modified. Changing these values causes the recalculation of the associated stiffness matrix. Figure 4.15 list the matrices needed for each matrix and their properties. The element stiffness matrices do not need to be reevaluated due to changes in the element degree of freedom values, \hat{a}^e , the applied forcing loads, or the values of the applied constraints.

Matrix	Size	Dependency	Consistency
\mathcal{I}	[3x3]	element shape	changes with every remeshing of the uv plane
$N_{L_2}^{12}$	[6x12]	(L_1, L_2, L_3) and \mathcal{I}	
$N_{L_1}^{12}$	[3x12]	(L_1, L_2, L_3) and \mathcal{I}	
N^{12}	[1x12]	(L_1, L_2, L_3) and \mathcal{I}	
J_2^{-1}	[3x6]	\mathcal{I}	
J_1^{-1}	[2x3]	\mathcal{I}	
$\bar{\beta}$	[3x3]	user input bending weights	user modifiable
$\bar{\alpha}$	[2x2]	user input stretching weights	user modifiable
\hat{a}^e	[12x1] (of 3x1 vectors)	Element degrees of freedom	changes each iteration
f	[1x1] (of 3x1 vector)	Element loading function	changes each iteration

Figure 4.15) Intermediate matrix sizes and dependencies

4.4.f) Assembly of the System Matrix

To build the system matrix all the element degrees of freedom are combined into the system degrees of freedom. Since many of the elements in a mesh will share nodes there will be considerable overlap in the system degree of freedom vector. The simplest scheme to generate the total system degrees of freedom without redundancy is to concatenate the degrees of freedom for all the nodes of the system. Each degree of freedom for each node is assigned a unique number running from 1 to n in the finite element mesh. Each element

will connect 12 of these degrees of freedom together. The 12 degrees of freedom of the element matrix will not necessarily be next to each other in the global degree of freedom list because the global degree of freedom ordering is arbitrary.

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_{u1} \\ \vdots \\ \mathbf{w}_i \\ \vdots \end{bmatrix} \quad 4.80$$

To determine how the individual element stiffness matrices are combined to make the system matrix, each element stiffness integral is now evaluated over the whole domain. To do this the original shape functions have to be augmented so that;

$$\mathbf{w}_{\text{element}_i}^h = \mathbf{N}_{\text{element}_i}^{12} \hat{\mathbf{a}}_{\text{element}_i}^e = \mathbf{N}_{\text{element}_i}^{\hat{\mathbf{X}}} \hat{\mathbf{X}} \quad 4.81$$

The $\mathbf{N}_{\text{element}_i}^{\hat{\mathbf{X}}}$ shape functions are defined to equal the $\mathbf{N}_{\text{element}_i}^{12}$ functions whenever the global degree of freedom $\hat{\mathbf{X}}_i$ happens to equal one of the element degree of freedom $\hat{\mathbf{a}}_i^e$; else they are zero. So although $\mathbf{N}_{\text{element}_i}^{\hat{\mathbf{X}}}$ is a $[1 \times n]$ vector it will have only 12 nonzero functions. The element stiffness matrices can now be taken as;

$$\mathbf{K}_{\hat{\mathbf{X}}_i} = \int_{\sigma} \mathbf{N}_{L_{2i}}^{\hat{\mathbf{X}}}{}^T \mathbf{J}_2^{-T} \bar{\beta} \mathbf{J}_2^{-1} \mathbf{N}_{L_{2i}}^{\hat{\mathbf{X}}} + \mathbf{N}_{L_{1i}}^{\hat{\mathbf{X}}}{}^T \mathbf{J}_1^{-T} \bar{\alpha} \mathbf{J}_1^{-1} \mathbf{N}_{L_{1i}}^{\hat{\mathbf{X}}} \, dudv \quad 4.82$$

$$\mathbf{F}_{\hat{\mathbf{X}}_i} = \int_{\sigma} \mathbf{f}^T \mathbf{N}_{\text{element}_i}^{\hat{\mathbf{X}}} \, dudv$$

Each $\mathbf{K}_{\hat{\mathbf{X}}_i}$ integral is a $[n \times n]$ matrix and each $\mathbf{F}_{\hat{\mathbf{X}}_i}$ forcing term is a $[n \times 1]$ vector. Where only the $12 \times 12 = 144$ terms associated with the original element stiffness matrix are nonzero. The locations of these terms are now appropriate so that all the element stiffness matrices can be added together to form the system matrix. In an implementation the mapping of the terms of the individual element stiffness matrices to their locations in the system matrix can be handled directly by keeping a map of the element degree of freedom numbers to the global degree of freedom numbers for each element. To build the system matrix each term of the element stiffness matrices are added into the system matrix based on

that mapping. For example, a 12 degree of freedom element degree of freedom mapping can be described as an [1x12] integer array as;

[1, 4, 8, 3, -1, -1, 2, 65, 13, 18, -1, 23]

related to the element degrees of freedom;

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

This mapping indicates that element degree of freedom number 2 maps to the global degree of freedom number 4. The -1s in the mapping indicate that element degrees of freedom 5, 6, and 11 are constrained and will not appear in the system stiffness matrix. The mapping is used to decide to add the element stiffness term, [3,4] to the global location [8,3]. After all the terms in all the element stiffness matrices are added the system matrix is complete.

4.5 A Deformable Curve Primitive

In this section the finite element method is applied to the curve equations to develop a deformable curve primitive. The resulting curve is piecewise cubic, C^1 continuous and like the surface primitives, opportunistically fair seeking. The curves were made C^1 piecewise cubic so that they could easily interface with the surface primitive during the skinning step. Each segment of a deformable character line will map to one edge of a deformable surface primitive.

4.5.a) Cubic Shape Functions

Any set of piecewise cubic basis functions can be used as the shape functions of the deformable curve finite element. Like the surface primitive, the actual choice of basis functions was determined by the ease of implementing boundary conditions. At a minimum, a curve will need to be able to support point location and tangent geometric constraints for the development of well defined artifacts.

Hermite polynomials, like any other parametric representation of a piecewise cubic curve represent each cubic segment as a weighted sum of 4 terms.

$$\mathbf{w}^h(u) = \sum_{i=0}^3 \mathbf{x}_i \phi_i \quad 4.83$$

Hermite polynomials have the advantage that the four weights \mathbf{x}_i correspond to the end point locations and end point tangents of the curve. As a counter-example, Bernstein polynomials have weights that correspond to the control point locations somewhere nearby the curve. In this work the Hermite polynomials were adopted as the basis functions for the deformable curve primitive. The Hermite polynomials and their associated geometric weights are listed below in Figure 4.16.

i	\mathbf{x}_i	ϕ_i
0	$\mathbf{w}(0)$	$1 - 3(u/h)^2 + 2(u/h)^3$
1	$\mathbf{w}_u(0)$	$h(u/h - 2(u/h)^2 + (u/h)^3)$
2	$\mathbf{w}(1)$	$3(u/h)^2 - 2(u/h)^3$
3	$\mathbf{w}_u(1)$	$h(-(u/h)^2 + (u/h)^3)$

where $0 \leq u \leq h = \text{length of element}$

Figure 4.16) Hermite polynomials

The finite element primitive based on Hermite polynomials is shown in Figure 4.17. It has 4 degrees of freedom distributed between two nodes located at the ends of the curve segment. The degrees of freedom at each node correspond to its position and tangent.

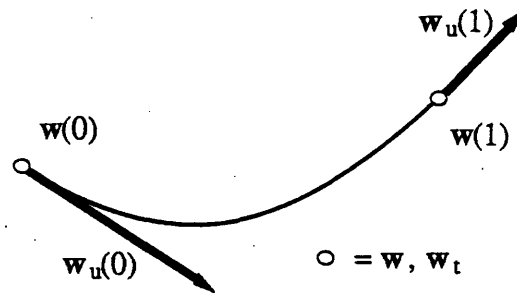


Figure 4.17) Curve element, 4 degrees of freedom

Hermite polynomials are not used commonly as the foundations for a CAGD tool. They are notably unstable if the user is required to directly select the positions and tangents at each node. For example, variations in the magnitude of the tangent can introduce large oscillations, extra inflection points and even cusp and loop singularities in between the nodal points. Hermite polynomials, along with any other piecewise cubic representation, will be well behaved with the energy minimization algorithm. The shape that is being found is the closest shape to the actual solution of the original minimization problem prior to the finite element discretization approximation. By letting the energy minimization select all but the geometrically constrained degrees of freedom for a curve instead of the user any curve stability problems are resolved. One caveat, if the user constrains all the degrees of freedom of a curve, the user is essentially turning off the energy minimization algorithm and returning to the direct manipulation of the parametric weights.

The Hermite polynomials were selected as the shape functions because they explicitly represent the geometric terms that will be constrained. This simplifies the implementation, increases the understanding of the system, and uses the reduced transformation constraint approach in its simplest form. The reduced transformation constraint strategy can also be extended to all constraints that can be expressed as a linear function of the element degrees of freedom. This fact greatly relaxes the constraint on selecting a shape basis function. If desired, any piecewise cubic parametric representation could serve as acceptable shape functions for the curve element. For curves the selection of the shape basis functions is a question of preference and convenience.

4.5.b) The Curve Stiffness Matrix

The stiffness and curve matrices are made by substituting the approximate shape into the

energy minimum principles. With $\mathbf{w}^h = \sum_{i=0}^3 \mathbf{x}_i \phi_i = \Phi^T \hat{\mathbf{a}}_e$ so that ;

$$\Phi = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{a}}_e = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_{1u} \\ \mathbf{w}_2 \\ \mathbf{w}_{2u} \end{bmatrix} \quad 4.84$$

The approximate curve equation becomes;

$$\begin{aligned}
 E_{\text{curve}} &= \int_c \mathbf{w}_{uu}^T \beta \mathbf{w}_{uu} + \mathbf{w}_u^T \alpha \mathbf{w}_u - 2 \mathbf{f}^T \mathbf{w} \, du \\
 &\approx \int_c \hat{\mathbf{a}}_e^T (\Phi_{uu}^T \beta \Phi_{uu} + \Phi_u^T \alpha \Phi_u) \hat{\mathbf{a}}_e - 2 \mathbf{f}^T \Phi \hat{\mathbf{a}}_e \, du
 \end{aligned}
 \tag{4.85}$$

The stiffness matrix can be evaluated as the α and β weighted sum of two terms;

$$\mathbf{K}_{\text{curve}} = \alpha \mathbf{K}_s + \beta \mathbf{K}_b
 \tag{4.86}$$

$$\begin{aligned}
 \text{where } \mathbf{K}_b &= \int_c \Phi_{uu}^T \Phi_{uu} \, du = \frac{1}{h^3} \begin{bmatrix} 12 & 6h & -12 & 6h \\ 6h & 4h^2 & -6h & 2h^2 \\ -12 & -6h & 12 & -6h \\ 6h & 2h^2 & -6h & 4h^2 \end{bmatrix} \\
 \mathbf{K}_s &= \int_c \Phi_u^T \Phi_u \, du = \frac{1}{30h} \begin{bmatrix} 36 & 3h & -36 & 3h \\ 3h & 4h^2 & -3h & -h^2 \\ 36 & -3h & 36 & -3h \\ 3h & -h^2 & -3h & 4h^2 \end{bmatrix}
 \end{aligned}$$

The element stiffness and bending matrices are functions of h , the length of the element in the parametric space. Unlike the solution derived by the finite difference method for these equations, each element in the model can be of different length. The advantage of this is that in regions of complicated geometry several small elements can be used, while relatively simple geometries can be captured by a few large elements saving on the size of the model and hence the number of calculations.

4.5.c) The Curve Forcing Matrix

The element forcing vector is found by evaluating the Φ weighted integrals of the forcing functions,

$$\mathbf{F}_c = \int_c \mathbf{f}^T \Phi \, du \quad 4.87$$

Taking this integral exactly requires a unique calculation for every type of forcing function supplied to the user. To simplify the implementation, all forcing functions are approximated by a linear interpolation between the actual force nodal values. Integrating this linear force function in the above equation is simple and gives a uniform method of handling any kind of force field that might be added at a later date for the user's convenience in manipulating the deformable curves. The force element distribution integral becomes;

$$\mathbf{F}_c = \int_0^h \left(\mathbf{f}_0 + \frac{u}{h}(\mathbf{f}_1 - \mathbf{f}_0) \right) \Phi \, du \quad 4.88$$

where \mathbf{f}_0 = applied force on node₀ and

\mathbf{f}_1 = applied force on node₁

And for cubic Hermite polynomials this integral can be taken to yield the element force distribution matrix equation;

$$\mathbf{F}_c = \frac{h}{60} \begin{bmatrix} 21 & 9 \\ 3h & 2h \\ 9 & 21 \\ -2h & -3h \end{bmatrix} \begin{bmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \end{bmatrix} \quad 4.89$$

where \mathbf{f}_0 = applied force on node₀ and

\mathbf{f}_1 = applied force on node₁

Assembling a system stiffness and forcing matrix for the curve is done exactly the same as for surfaces using these curve element stiffness and forcing matrices.

4.6 Chapter 4 Summary

The first half of this chapter was a tutorial style review of the mathematical foundations of the deformable primitives developed in the second half of the chapter. A solution technique for solving for deformable shapes based on the finite difference method was shown to result in surface representations inappropriate for manufacture and of limited topological range. Such solutions do not yield continuous surface models.

The mechanics and the theory of the Ritz finite element method were presented. It was shown that the solution found with the finite element theory is the closest possible solution to the actual solution as measured by the energy of the error. Because of this property, C^1 shape functions can be used to find solutions that tend to C^3 shapes as the number of degrees of freedom of the model is increased. As such we found that extremely fair shape could be generated with only C^1 continuity between shape patches. The finite element method can be applied to any space of shape functions. For example it is possible to develop deformable Bezier patches. In this work shape functions which simplified the enforcement of geometric constraints were selected.

Finite elements for both surface and curve elements were developed. The surface finite element is triangular instead of square to extend the range of topological shapes that can be modeled. The shape of the element varies cubically along each edge and has a surface normal that varies parabolically. The shape functions enforce C^1 continuity between adjacent elements. The element has twelve degrees of freedom which include the position and parametric tangents at all the triangle vertices plus the three mid-side tangents normal to the edge direction. It was shown how these shape functions can directly enforce the pinned, hinged and fixed edge geometric constraints required by the ShapeWright paradigm.

A curve finite element was presented to be used as the basis of the character lines used in the first step of the ShapeWright paradigm. The elements were made C^1 continuous, piecewise cubic and represented by Hermitian shape functions to maximize their ability to interface easily with the shape primitives. Each element has four degrees of freedom, the elements end point positions and tangents.

5 Survey of Constraint Methods

At the heart of the ShapeWright design paradigm is the notion that surfaces can be sculpted while they continually maintain a set of geometric constraints where they skin the set of character lines. Geometric constraints are defined here as the class of constraints which limits the space in which a shape can move. An example of a geometric constraint is a bead fixed to slide along a curve or surface.

For an implementation, these geometric constraints will have to be enforced numerically at each time integration step. In the first half of this chapter the mechanics and properties of enforcing geometric constraints sufficient to support the ShapeWright paradigm are discussed. The Lagrange multiplier, the penalty method and the reduced transformation techniques for constraint enforcement are developed and compared. A technique by Baumgarte [Baumgarte_72] that shows how to use Lagrange multipliers in a manner appropriate for the explicit integration of the dynamic equations is also given.

The last half of the chapter presents a proposal on how to enforce geometric constraints that are described parametrically. The need for parametrically described geometric constraints will be common in a parametric shape modeler. If parametric constraints can be applied to geometric surfaces then the internal representation used for constraints and geometry can be the same. For the user this means that any geometry that is built can be used as a constraint on geometry to be added.

Example applications for parametric geometric constraints include making one piece of geometry interact with another and solving the lightly scattered data problem. Imagine attaching a handle to a cup and then being able to slide the handle over the surface of the cup until a desirable location for it is found. One major difficulty in solving the lightly scattered data problem is how to assign the measured point locations to particular parametric locations in the surface. The parametric constraints can be used to accomplish this step automatically and optimally. For example, interpolating a curve through a set of points can be accomplished if a deformable curve could be made to slide continuously over the set of points in space like a rope going through a network of pulleys. One small step and its easy to imagine a surface sliding through a set of two dimensional pulleys to interpolate a set of measured surface points.

In general a geometric modeling package will be limited by the range of geometric constraints it can enforce. The very limited set of constraints implemented in this work, including free, pinned, hinged and fixed edges, are sufficient for the ShapeWright program. Extending the set of constraints in future work will extend the range of applications to which deformable models can be applied. Parametrically described geometric constraints represent a uniform method for supporting a wide range of constraints and applications.

5.1 Reduced Transformation Equations

The reduced transformation technique is the simplest and best behaved method for enforcing geometric constraints on a linear set of equations such as $\mathbf{AX} = \mathbf{B}$. This technique can enforce geometric constraints which are described as linear functions of the explicit degrees of freedom of the unconstrained system. This includes the most common constraint of fixing a particular system degree of freedom to a set value. The technique is best described by example. Consider the following [3x3] symmetric system of equations;

$$\begin{bmatrix} 4 & 3 & 1 \\ 3 & 5 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \quad 5.1$$

$$\mathbf{A} \quad \mathbf{X} = \mathbf{B}$$

The desire will be to find values of x_1 and x_2 for arbitrary values of f_1 and f_2 when x_3 is set equal to 3; $x_3 = 3$. This constraint written as a general linear equation of the original variables is expressed as;

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \quad 5.2$$

$$\mathbf{X} = \mathbf{D}_0 \quad \mathbf{Y} + \mathbf{D}_1$$

where the \mathbf{Y} vector is a reduced set of degrees of freedom that automatically accommodate the constraint of equation 5.2.

Substituting the constraint equation into the original equation set and pre-multiplying by D_0^T to preserve symmetry yields;

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 & 1 \\ 3 & 5 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 & 1 \\ 3 & 5 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \quad 5.3$$

$$D_0^T \quad A \quad D_0 \quad Y = D_0^T \quad B - D_0^T \quad A \quad D_1$$

which when multiplied out is just;

$$\begin{bmatrix} 4 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \end{bmatrix} 3 \quad 5.4$$

$$D_0^T A D_0 \quad Y = D_0^T B - D_0^T A D_1$$

The reduced transformation method replaces the original system equations, $AX = B$, with a linear constraint equation, $X = D_0 Y + D_1$, and a transformed set of system equations, $D_0^T A D_0 Y = D_0^T B - D_0^T A D_1$. The new system equations are solved for Y , and the constraint equation is used to find X .

For the case in this example, of constraining one degree of freedom to a fixed value, the transformation is remarkably simple. The matrix $D_0^T A D_0$ is made by deleting a row and a column from the original A matrix. The vector $D_0^T B$ is made by deleting the row of the constrained degree of freedom from the original B matrix. The extra forcing term is made by subtracting the deleted column of the A matrix multiplied by the value of the constraint from the forcing vector. Changing the value of the constraint, for example from $x_3 = 3$ to $x_3 = 4$, will result in changing the new load vector but not in changing the system matrix.

The general properties of the reduced transformation approach for enforcing a set of fixed degree of freedom constraints will be the same as for this example of enforcing a single constraint. The new stiffness matrix will be made by deleting rows and columns from the original stiffness matrix preserving its symmetry and positive definiteness while reducing its size. Each constraint will add forcing terms to the new system of equations. Modifying the value of a constraint (a term in the D_1 matrix) will only change the forcing vector and not the system matrix. In an implementation, moving a fixed node can happen as quickly and simply as applying a force.

In general the transformation can be used to enforce any constraint as long as it can be expressed in the linear form given above. As long as the rank of the D_0 matrix is equal to its shortest dimension the transformed system matrix will preserve the symmetry and positive definiteness of the original A matrix.

5.2 Penalty Methods

The Reduced transformation technique compares favorably to the well known penalty method of constraint enforcement. The penalty method applies to the same range of constraints as the reduced transformation equations but fails to enforce the constraints exactly and causes some numerical difficulties. The idea is that constraints can be enforced approximately by augmenting the minimum principle related to the $AX = B$ system of equations. The linear constraints can be written as ;

$$0 = D_0 X + D_1 = D_{\text{penalty}} \quad 5.5$$

Each row of the $[m \times n]$ D_{penalty} matrix represents a different constraint. The minimum principle related to $AX = B$ is;

$$I(w) = X^T A X - 2X^T B \quad 5.6$$

It is augmented so that energy is added to the system whenever any of the constraints are violated.

$$I_p(w) = X^T A X - 2X^T B + D_{\text{penalty}}^T \Gamma D_{\text{penalty}} \quad 5.7$$

$$\text{with } \Gamma = \begin{bmatrix} \gamma_1 & & 0 \\ & \ddots & \\ 0 & & \gamma_m \end{bmatrix}$$

Each of the γ_i are gains that weight the relative importance of each constraint. As a constraint weight γ_i is increased the system of equations increasingly acts to reduce the error of that constraint. The terms of $I_p(w)$ can be expanded as;

$$I_p(w) = X^T A X - 2X^T B + X^T D_0^T \Gamma D_0 X + 2D_1^T \Gamma D_0 X + D_1^T \Gamma D_1 \quad 5.8$$

The minimum of this augmented functional can be found by solving the following equation set;

$$(A + D_0^T \Gamma D_0)X = B - D_1^T \Gamma D_0 \quad 5.9$$

When the constraints are simple, as in the previous example of constraining a single degree of freedom to a particular value, the above matrices are very simple. The matrix added to the A matrix becomes zero everywhere except for one term, equal to γ_i , to be added to the diagonal element of the row for the constrained degree of freedom. The forcing function is also modified by only one term, equal to the degree of freedom value $\times \gamma_i$, added to the row of the constrained degree of freedom.

The advantage of the penalty method is that it is very easy to add the few terms needed to enforce a constraint to the current A and B matrices. The disadvantages are that the constraints are not enforced exactly and that as the γ_i gets large the numerical condition number of the augmented A matrix becomes poor making it difficult for numerical algorithms to run accurately.

The one advantage the penalty method has over the reduced transformation approach is that conflicting constraints can be accommodated. The penalty method will enforce neither precisely. The reduced transformation technique avoids such problems by the limitation placed on the rank of the D_0 matrix. Since the reduced transformation technique enforces its constraints exactly while preserving the numerical stability of the original problem without increasing the implementation difficulties, it is generally preferable to the penalty method.

5.3 Lagrange Multipliers

Constraints enforced by Lagrangian multipliers can be much more general than the linear constraints supported by the reduced transformation equations. The Lagrange multiplier method works with the energy functional statement of the problem. We seek some shape, \mathbf{w} , which is the one shape out of all allowed shapes which minimizes the energy functional,

$$I(\mathbf{w}) = \int_a f(\mathbf{w}, \mathbf{w}', \mathbf{w}'') \, du \quad 5.10$$

where $f(\mathbf{w}, \mathbf{w}', \mathbf{w}'')$ is some arbitrary function of \mathbf{w} and its derivatives.

We wish to constrain the search for \mathbf{w} such that only shapes that meet some constraint $g(\mathbf{w}) = 0$ are considered. The problem is now to minimize $I(\mathbf{w})$ subject to the constraint $g(\mathbf{w}) = 0$.

The constraint $g(\mathbf{w})$ is integrated into the variational framework by adding it to the energy functional as;

$$I(\mathbf{w}) = \int_a f(\mathbf{w}, \mathbf{w}', \mathbf{w}'') - \lambda g(\mathbf{w}) \, du = \int_a h(\mathbf{w}, \mathbf{w}', \mathbf{w}'', \lambda) \, du \quad 5.11$$

where $\mathbf{w}, \mathbf{w}', \mathbf{w}''$, and λ are all functions of $u \in a$

This augmented energy functional will have a minimum when its related Euler differential equations are solved as;

$$\frac{d^2(\partial h / \partial \mathbf{w}'')}{du^2} - \frac{d(\partial h / \partial \mathbf{w}')}{du} + \frac{\partial h}{\partial \mathbf{w}} = 0 \quad \text{and} \quad g(\mathbf{w}) = 0 \quad 5.12$$

or equivalently;

$$\frac{d^2(\partial f / \partial \mathbf{w}'')}{du^2} - \frac{d(\partial f / \partial \mathbf{w}')}{du} + \frac{\partial f}{\partial \mathbf{w}} - \lambda \frac{dg}{d\mathbf{w}} = 0 \quad \text{and} \quad g(\mathbf{w}) = 0 \quad 5.13$$

Examination of the Euler equations reveals that the minimum of the augmented functional will indeed satisfy the constraint exactly. The original constraint equation, $g(\mathbf{w})$ is generated as one of the Euler equations. A physical insight to Lagrange variables can be made by considering the simpler minimum principle,

$$I(\mathbf{w}) = \int_a f(\mathbf{w}) - \lambda g(\mathbf{w}) \, du \quad 5.14$$

Which is minimized when:

$$\frac{df}{d\mathbf{w}} = \lambda \frac{dg}{d\mathbf{w}} \quad \text{and} \quad g(\mathbf{w}) = 0 \quad 5.15$$

When \mathbf{w} is a vector of two variables, u and v , so that $f = f(u,v)$ and $g(u,v) = 0$ the Euler equations become,

$$\nabla f = \lambda \nabla g \quad 5.16$$

which has a simple geometric interpretation. Figure 5.1a shows the constraint $g(u,v)$ as a curve in the plane and the function $f(u,v)$ as a set of contour lines. In Figure 5.1b an arbitrary point on the constraint is considered. The gradient of the constraint and the gradient of the minimizing function are not aligned. In such a situation it will be possible to continue decreasing the value of $f(u,v)$ while satisfying the constraint $g(u,v)$ by moving along the constraint to the position shown in Figure 5.1c. Here the gradient of the constraint and the minimizing function are aligned. This is the solution to the problem. There is no other point on the constraint that has a lower value of $f(u,v)$ than this point.

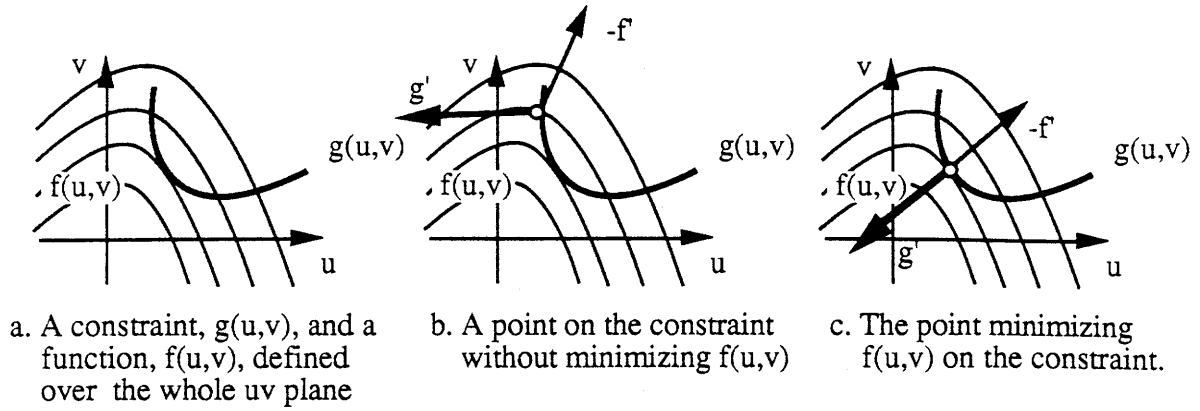


Figure 5.1) The geometry of Lagrangian constraints

Physically, the lambda weighted term in the Euler equations can be interpreted as a force that acts to eliminate the component of the net force acting on the body that would act to violate the constraint. In this view, Figure 5.1 can be interpreted as a bead constrained to stay on a wire, $g(u,v)$, subject to forces generated by the potential field, $f(u,v)$. At any moment of time the total force acting on the bead will always point along the direction of the constraint, only allowing the bead to move along the length of the wire and not off the wire. At the equilibrium location, the body force will be equal and opposite the constraint force and the potential energy of the system, the value of $f(u,v)$, will be minimized.

To see how the Lagrange multiplier method effects the system of equations being solved by the Ritz finite element method consider a general non-linear constraint $g(\mathbf{w})$. Such constraints can be used to make the volume of a closed container remain constant during deformation or to cause two points in a system to remain in contact. Applying the Ritz

approximation to shape, $\mathbf{w}^h = \sum_i \mathbf{x}_i \phi_i$, and to the lambda function itself, $\lambda^h = \sum_i \lambda_i \phi_{\lambda i}$ the constraint and the lambda function becomes;

$$g(\mathbf{w}) \approx g(\mathbf{w}^h) = g(x_1, x_2, \dots, x_n) \quad \text{and} \quad 5.17$$

$$\lambda(u) \approx \lambda^h = \sum_i \lambda_i \phi_{\lambda i} = \Phi_\lambda \Lambda$$

The linear approximation to $g(\mathbf{w}^h)$ for the current configuration, $\mathbf{w}^{h_0} = \Phi \mathbf{X}_0$ is given by;

$$g(\mathbf{X}_0 + \mathbf{dX}) \approx g(\mathbf{X}_0) + \sum_i \frac{\partial g}{\partial x_i} dx_i \quad 5.18$$

Using Taylor series expansions about the shape $\mathbf{w}^{h_0} = \Phi \mathbf{X}_0$ for the quadratic and linear terms of the functional yields;

$$\begin{aligned} (\mathbf{X}_0 + \mathbf{dX})^T \mathbf{K}_\sigma (\mathbf{X}_0 + \mathbf{dX}) &= \mathbf{X}_0^T \mathbf{K}_\sigma \mathbf{X}_0 + 2\mathbf{X}_0^T \mathbf{K}_\sigma \mathbf{dX} + \mathbf{dX}^T \mathbf{K}_\sigma \mathbf{dX} \quad \text{and} \\ \mathbf{F}_\sigma^*(\mathbf{X}_0 + \mathbf{dX}) &= \mathbf{F}_\sigma \mathbf{X}_0 + \mathbf{F}_\sigma \mathbf{dX} \end{aligned} \quad 5.19$$

and the resulting minimum principle after dropping constant terms and taking the integrals becomes;

$$\begin{aligned} I_\lambda(\mathbf{w}^h) &= \mathbf{dX}^T \mathbf{K}_\sigma \mathbf{dX} + 2\mathbf{X}_0^T \mathbf{K}_\sigma \mathbf{dX} - 2\mathbf{F}_\sigma \mathbf{dX} - \Lambda^T \mathbf{K}_\lambda \mathbf{dX} \\ \text{where } \mathbf{K}_\lambda &= \int_a \Phi_\lambda^T \mathbf{C}_\lambda \mathbf{da} \quad \text{and} \quad \mathbf{C}_\lambda = \begin{bmatrix} \partial g / \partial x_1 \\ \partial g / \partial x_2 \\ \vdots \\ \partial g / \partial x_n \end{bmatrix} \end{aligned} \quad 5.20$$

which can be solved as the augmented equation set;

$$\begin{bmatrix} \mathbf{K}_\sigma & \mathbf{K}_\lambda \\ \mathbf{K}_\lambda^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{dX} \\ \Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{F}_\sigma - \mathbf{X}_0^T \mathbf{K}_\sigma \\ 0 \end{bmatrix} \quad 5.21$$

This new system matrix is symmetric but has lost its positive definite property. This can be seen by using elimination to rewrite the system matrix from its current form to an upper sub matrix form as,

$$\begin{bmatrix} \mathbf{K}_\sigma & \mathbf{K}_\lambda \\ \mathbf{K}_\lambda^T & 0 \end{bmatrix} \text{ becomes } \begin{bmatrix} \mathbf{K}_\sigma & \mathbf{K}_\lambda \\ 0 & -\mathbf{K}_\lambda^T \mathbf{K}_\sigma^{-1} \mathbf{K}_\lambda \end{bmatrix} \quad 5.22$$

This related matrix is non-definite because \mathbf{K}_σ is positive definite which implies \mathbf{K}_σ^{-1} and $\mathbf{K}_\lambda^T \mathbf{K}_\sigma^{-1} \mathbf{K}_\lambda$ will be positive definite, meaning that $-\mathbf{K}_\lambda^T \mathbf{K}_\sigma^{-1} \mathbf{K}_\lambda$ is negative definite. The combination of a positive and negative sub matrix on the diagonal guarantees the non-

definiteness of the system matrix. In general, standard Gaussian elimination will not work for such a system of equations.

One strategy for using the Lagrange multiplier method for nonlinear constraints is to build the above equation set every time the geometry of the system is changed and then solve for the augmented set of unknowns, $d\mathbf{X}$ and Λ . Unlike the reduced transformation constraint method the number of unknowns is increased and the positive definiteness of the original matrix equations is not preserved. The advantage of such an approach is that very arbitrary constraints can be enforced.

5.4 Stabilized Constraints

The Lagrange multiplier method results in an augmented system matrix that is non-positive definite. This loss of positive definiteness implies that the step to invert the system matrix required by the implicit integration of the dynamic deformable model equations may not work. Explicit instead of implicit integration can be used to solve the dynamic equations through time without inverting the system matrix. In this section a variation of the Lagrange multiplier technique, developed by J. Baumgarte [Baumgarte_72], appropriate for the stable explicit integration of a set of dynamic equations subject to nonlinear constraints is presented.

5.4.a) Constrained Equations of Motion

Baumgarte considered the problem of explicitly integrating a set of dynamic equations subject to algebraic constraints.

As a simple example he considered the equations of motion for a system of n distinct mass particles with locations expressed as \mathbf{w}_i . The equations of motion for this system are;

$$M_i \frac{d^2 \mathbf{w}_i}{dt^2} - \mathbf{F}_i = 0 \quad \text{for } i = 1, 2, 3, \dots, 3n \quad 5.23$$

where M_i are the masses of the particles
 \mathbf{F}_i are the applied external forces

The geometric constraints to be applied to the particles are limited to functions which are linear in the acceleration of the particles as;

$$g(\mathbf{w}_1, \dot{\mathbf{w}}_1, \ddot{\mathbf{w}}_1; t) = \sum_i g_i(\mathbf{w}_1, \dot{\mathbf{w}}_1; t) \ddot{\mathbf{w}}_i + G(\mathbf{w}_1, \dot{\mathbf{w}}_1; t) = 0 \quad 5.24$$

where \mathbf{w}_1 = a subset of the degrees of freedom

The Lagrange multiplier technique for enforcing these constraints starts by augmenting the associated minimization principle for the original equations of motion as;

$$\frac{1}{2} \sum_i M_i \left(\dot{\mathbf{w}}_i - \frac{\mathbf{F}_i}{M_i} \right)^2 - \lambda g(\mathbf{w}_1, \dot{\mathbf{w}}_1, \ddot{\mathbf{w}}_1; t) \quad 5.25$$

Setting the partials of this new constrained minimization principle with respect to all the accelerations and the Lagrange λ variable to zero, yields the following set of equations to be solved;

$$\begin{bmatrix} \mathbf{M} & \mathbf{K}_g^t \\ \mathbf{K}_g & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{X}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ -\mathbf{G} \end{bmatrix} \quad 5.26$$

where \mathbf{F} = column vector of all applied external forces
 \mathbf{X} = column vector of all system degrees of freedom
 \mathbf{M} = Mass matrix of all particles
 \mathbf{K}_g^t = column vector of all g_i values

This system of equations can now be used by an explicit integration scheme to find the accelerations of the system based on its current state to simulate its performance through time.

5.4.b) Holonomic Constraints

Holonomic constraints are expressed as functions of the positions of the current system.

$$N(\mathbf{x}_1, t) = 0 \quad 5.27$$

This constraint can be differentiated twice in time to cast it into the form required by the previous section as;

$$\dot{N} = \sum_i \frac{\partial N}{\partial \mathbf{w}_i} \dot{\mathbf{w}}_i + \frac{\partial N}{\partial t} = 0 \quad 5.28$$

$$\ddot{N} = \sum_i \frac{\partial N}{\partial \mathbf{w}_i} \ddot{\mathbf{w}}_i + \sum_i \frac{\partial^2 N}{\partial \mathbf{w}_i^2} \dot{\mathbf{w}}_i^2 + \frac{\partial^2 N}{\partial t^2} = 0 \quad 5.29$$

$$= \sum_i g_i \dot{\mathbf{w}}_i + G; \quad \text{where} \quad g_i = \frac{\partial N}{\partial \mathbf{w}_i}$$

The minimization principle can be written as,

$$\frac{1}{2} \sum_i M_i \left(\dot{\mathbf{w}}_i - \frac{\mathbf{F}_i}{M_i} \right)^2 - \lambda \dot{N} \quad 5.30$$

The resulting equations of motion from this system will guarantee that

$$\dot{N} = 0 \quad 5.31$$

5.4.c) Stabilized Constraints

Baumgarte points out that this system of equations is unstable for the value of N . The solution of $\dot{N} = 0$ is,

$$N = \dot{N}_{t=0} t + N_{t=0} \quad 5.32$$

As long as the initial conditions of the integration are such that

$$\dot{N}_{t=0} = 0 \quad \text{and} \quad N_{t=0} = 0 \quad 5.33$$

The resulting integration will yield a state that evolves through time with $N = 0$ always being true. However, initial condition errors or perturbations introduced through numerical methods will cause N to diverge linearly through time. Baumgarte suggests one remedy to this problem by changing the differential operator, D , applied to the holonomic constraint N

$$\text{from } D(\) = \frac{d^2(\)}{dt^2} \quad \text{to} \quad \tilde{D}(\) = \frac{d^2(\)}{dt^2} + 2\xi\omega_n \frac{d(\)}{dt} + \omega_n^2(\) \quad 5.34$$

The new operator is the familiar second order oscillator. By setting $\xi = 1$ the constraint system becomes critically damped. Any errors in N caused by initial conditions or perturbations will be eliminated as the integration proceeds through time without overshoot. Setting $\xi = .707$ will cause the constraint to be enforced in fewer time steps but will allow overshoot.

In general, this system will cause the constraint to be enforced exactly as time runs to infinity. In practice, the constraint will be effectively enforced in a few time steps. The constraint enforcing natural frequency, ω_n , can be chosen so that the constraints are enforced much faster than the dynamics of the system being simulated. A rapid constraint natural frequency will affect the requirements placed on the time step used in the integration scheme to ensure accuracy and stability.

In matrix form this change in the equations will effect only the G term.

$$\tilde{D}(N) = \frac{d^2(N)}{dt^2} + 2\xi\omega_n \frac{d(N)}{dt} + \omega_n^2(N) = \sum_i g_i \ddot{w}_i + \tilde{G} \quad 5.35$$

$$\text{where } \tilde{G} = \sum_i \frac{\partial^2 N}{\partial w_i^2} \dot{w}_i^2 + \frac{\partial^2 N}{\partial t^2} + 2\xi\omega_n \left(\sum_i \frac{\partial N}{\partial w_i} \dot{w}_i + \frac{\partial N}{\partial t} \right) + \omega_n^2(N)$$

yielding

$$\begin{bmatrix} M & K_g^t \\ K_g & 0 \end{bmatrix} \begin{bmatrix} \dot{X} \\ \lambda \end{bmatrix} = \begin{bmatrix} F \\ -\tilde{G} \end{bmatrix} \quad 5.36$$

5.5 Stabilized Parametric Constraints

In this section a technique is proposed to enforce parametrically described constraints. The technique is an extension of the Lagrange multiplier technique and as such can support nonlinear constraints. Also like the Lagrange multiplier technique, the resulting system of equations can be described by a non-positive definite system matrix. As such the Baumgarte stabilized constraint idea can be used to generate a system of equations suitable for explicit integration.

The major advantage of considering explicitly represented parametric constraints is that the internal representations used in an implementation of a shape modeler for geometry and constraints can be the same. In this manner any geometry built by a user can be used as a constraint on geometry to be built.

Conceptually, this constraint strategy can be thought of as creating a bead for each constraint. Each constraint bead exists in the parametric space of its associated constraint. A constraint bead is subject to the forces applied to its related degree of freedom projected into the constraint space. As the bead moves through the constraint space it traces out a trajectory in 3 space as defined by the parametric constraint. Baumgarte's stabilized Lagrange multiplier technique is then used to force the constrained degree of freedom to follow the bead's trajectory through space and time.

An alternative to this idea is to use the constraint equations to eliminate the constrained degrees of freedom from the system of equations. The advantage of the parametric constraint being maintained explicitly in the constraints is that the resulting equations can be easily divided into sub-sections consisting of terms that are constant through the integration and terms that change with each iteration. This serves as a potential benefit for organizing the implementation of the resulting equations.

The following equations were written in a form suggested by Baumgarte's stabilized constraints appropriate for explicit integration. The same ideas of parameterized constraints and constraint beads subject to projected forces that track with the degrees of freedom of the original system could have been included in the original energy functions as written in the section on Lagrange constraints resulting in a set of equations more appropriate for implicit integration. In retrospect, after experimenting with explicit integration, it probably

would have been wiser to have taken this alternative and investigated solution techniques capable of solving the non-positive definite matrices of the Lagrange multiplier method in a rapid fashion. Such techniques probably already exist and are just not known to the author since the Lagrange multiplier matrices can be made to be invertible even though they are not positive definite.

5.5.a) Geometric Parametric Constraint

The geometry of the surface or curve acting as a constraint can be expressed in parametric form as a set of functions depending on a parametric vector, s .

$$R_p = [x(s), y(s), z(s)] \quad 5.37$$

$$\text{where } s = [s_1, s_2, \dots, s_p]$$

When $p = 1$, s is just a scalar and the shape described is a curve. When $p = 2$ the shape is a surface and so forth.

A three space point geometric constraint applied to the system of particles considered in the previous sections can be written as the set of parametric equations,

$$N_p \equiv R_p(s) - R_i = 0 \quad 5.38$$

$$\text{where } R_i = \text{location of the } M_i \text{ particle}$$

This equation can be thought of as requiring the location of the i^{th} particle of the system to be the same as the position of the constraint determined by the parametric variable, s . The parametric variable s is an unknown that will be determined.

The Lagrange multiplier method for enforcing constraints by augmenting a minimization principle require that the constraint be a function of only the state variables. To meet this requirement the original state vector of particle accelerations is augmented by the acceleration of the parametric variable, \ddot{s} .

Each equation in the parametric constraint can be introduced into the minimization principle using the stabilized constraint procedure of the previous section.

$$\frac{1}{2} \sum_i M_i \left(\ddot{\mathbf{R}}_i - \frac{\mathbf{F}_i}{M_i} \right)^2 - \sum_j \lambda_j \tilde{D}(N_p) \quad 5.39$$

$$\text{where } \tilde{D}(N_p) = C_p \ddot{\mathbf{s}} + G_p$$

$$\text{and } C_p \equiv \frac{\partial \mathbf{R}_p}{\partial \mathbf{s}} \text{ and, } G_p \equiv \frac{\partial^2 \mathbf{R}_p}{\partial \mathbf{s}^2} \dot{\mathbf{s}}^2 + 2\xi\omega_n(C_p \dot{\mathbf{s}} - \dot{\mathbf{R}}_i) + \omega_n^2(\mathbf{R}_p - \mathbf{R}_i)$$

Seeking the minimum of this relation by taking the partial with respect to the free variables and setting each such result to zero yields the following matrix equation set.

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & C_p^T \\ -\mathbf{I} & C_p & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{R}} \\ \ddot{\mathbf{s}} \\ \Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \\ -G_p \end{bmatrix} \quad 5.40$$

Unfortunately, this set of equations is not well behaved. The bottom row of the above matrix equation is $\ddot{\mathbf{R}} = C_p \ddot{\mathbf{s}}$ which state that the acceleration of the particle will be equal to the acceleration of a point moving along the parametric constraint up to linear order. This is the equation which binds the acceleration of the constraint bead to the acceleration of the constrained degree of freedom. As long as the particle and the constraint bead start out at the same location and no perturbations this equation will act to fix the degree of freedom's location to the constraint bead's location. There are two major problems with the constraint equations a presented. In addition to the Baumgarte stabilization problem, these equations lack any information which specify where the constraint bead is supposed to move in the parametric constraint space. Over time any initial error in the relation between a fixed particle and location of the constraint bead will not be eliminated.

5.5.b) Change of Basis

One solution to this problem is to subject the imagined constraint bead to the same forces applied to the moving particle. Since the constraint bead can only move in the parameter space only those forces which project into that space are actually applied. The acceleration

of a particle in three space subject to a constraint can be related to the acceleration of the parametric variable by differentiating the constraint twice with respect to time.

$$\dot{R}_p = \frac{\partial R_p}{\partial s} \dot{s} \quad \text{and} \quad \ddot{R}_p = \frac{\partial R_p}{\partial s} \ddot{s} + \frac{\partial^2 R_p}{\partial s^2} \dot{s}^2 \quad 5.41$$

Rewriting yields,

$$\ddot{R}_p = C_p \ddot{s} + B_p \quad 5.42$$

$$\text{where } B_p = \frac{\partial^2 R_p}{\partial s^2} \dot{s}^2$$

Matrices C_p and B_p define a transformation from the vector s in parameter space to the vector R_i in 3 space. The original dynamic equation can now be expressed in the parametric subspace. Starting with,

$$M \ddot{R}_p = F \quad 5.43$$

and substituting the change of basis yields,

$$M(C_p \ddot{s} + B_p) = F \quad 5.44$$

and pre-multiplying by C_p^T gives the relation,

$$C_p^T M C_p \ddot{s} = C_p^T F - C_p^T M B_p \quad 5.45$$

This relation states that the bead on the wire is subjected to the projection of the external forces into the space of the constraint plus a term generated by the parameterization of the constraint. A little work will show that $C_p^T M B_p$ will be zero whenever the parameterization of the constraint is linear in arc length. The leading term $C_p^T M C_p$ will be diagonal whenever the components of the parametric vector s_i are orthogonal in the subspace of the constraint. With properly crafted constraint parameterizations this equation set will be very easy to build.

This information can be included in the minimization principle as,

$$\frac{1}{2} \sum_i M_i \left(\ddot{\mathbf{R}}_i - \frac{\mathbf{F}_i}{M_i} \right)^2 + \frac{1}{2} \mathbf{C}_p^T \mathbf{M} \mathbf{C}_p \left(\ddot{\mathbf{s}} - (\mathbf{C}_p^T \mathbf{M} \mathbf{C}_p)^{-1} (\mathbf{C}_p^T \mathbf{F} - \mathbf{C}_p^T \mathbf{M} \mathbf{B}_p) \right)^2 - \sum_j \lambda_j \tilde{\mathbf{D}}(\mathbf{N}_p) \quad 5.46$$

Seeking the minimum of this equation set results in the following matrix equation set.

$$\begin{bmatrix} \mathbf{M} & 0 & -1 \\ 0 & \mathbf{C}_p^T \mathbf{M} \mathbf{C}_p & \mathbf{C}_p^T \\ -1 & \mathbf{C}_p & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{R}} \\ \ddot{\mathbf{s}} \\ \Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{C}_p^T \mathbf{F} - \mathbf{C}_p^T \mathbf{M} \mathbf{B}_p \\ -\mathbf{G} \end{bmatrix} \quad 5.47$$

The advantage of this scheme is that the original equations of the unconstrained system remain unaltered and the geometric constraints can be expressed explicitly as parametric functions. The bulk of the above matrix equations will remain constant for the entire time integration. All terms depending on the linearization of the constraint trajectory acceleration, \mathbf{C}_p will have to be rebuilt at each time step.

5.6 Chapter 5 Summary

The range and sophistication of a geometric modeler will be limited by the range of geometric constraints that can be enforced. A scheme that uniformly enforces as large a range of constraints as possible is desirable for the ease of implementation and consistency of use. Constraint schemes appropriate for enforcing the ShapeWright paradigm were reviewed. The range of constraints needed for this purpose are defined to be the pinned, hinged and fixed edge constraints. It was pointed out that the simplest and stablest constraint technique, the reduced transformation equations, can be used for this purpose.

Lagrange multiplier constraint methods result in matrix equations that are non-positive definite. Explicit time integration of the dynamic deformable model equations can be used to solve for the deformable shape without having to invert the system stiffness matrix. Baumgarte's stabilized constraints represent a scheme to enforce Lagrangian multiplier geometric constraints using explicit integration in a stable fashion, enforcing constraints exactly in the limit as time runs to infinity.

Extending Baumgarte's idea, a proposal was made to enforce explicitly represented geometric constraints. Such geometric constraints would be very valuable to a geometric modeler allowing both geometry and constraints to share the same internal representation.

Actual implementations using the explicit integration techniques failed to produce systems that were interactive due to the small time steps required to make the integrations stable. Future research on extending the range of geometric constraints suitable for deformable models would probably be well spent on developing implicitly integrated schemes that can support the Lagrange multiplier non-positive definite matrices.

6 Implementations

In this chapter the structure and some sample output from three different programs that have been written to implement the material presented in chapters 3 and 4 are discussed. In addition, some of the steps taken to guarantee the rapid rendering of objects are presented. All of these programs were implemented on a Silicon Graphics 4D/70GT work station in the C language using the Unix operating system.

The first program discussed is GGT, which solves the surface equation using a finite difference approach. This program, the first to be written, was created to verify the notion of using deformable surfaces for shape design. The next two programs in the chapter, CUBIC and ECS, were written to demonstrate that the positive attributes of deformable surface design could be integrated with an approach appropriate for the definition of manufacturable objects. These programs are based on the finite elements discussed in chapter 4. CUBIC is an implementation of the deformable curve and ECS is an implementation of the deformable surface.

Both ECS and CUBIC have been implemented in an object oriented style and are appropriate for future integration into a single program.

6.1 A Finite Difference Program for Deformable Surface Design, GGT

The program GGT is a finite difference based implementation of the deformable surface equations. It limits itself to surfaces that are represented as a uniform grid of points defined over the unit square in the uv domain. Each solution point on the surface can be assigned its own α matrix, β matrix, and applied force. Any point in the array can be fixed to a location in xyz space. Additionally, the grid can support periodic boundary conditions where the column of grid points bounding the u dimensions are treated as if they are next to each other. This is useful for the design of cylindrical type objects.

The interface to this package has been designed so that any of the input commands can be used to parameterize the surface. One such example is the pressure command. Loads have been written to simplify the task of assigning an applied load to each point in the solution array. A pressure applies a load in the direction of the surface normal to every affected

point. A pressure load can be applied to any subset of the solution points be they adjacent or not. The interface command to apply a pressure load is;

`pr(value)`

6.1

where value is the magnitude of the applied pressure load.

When the user types `pr(?)`, an instance of a slider bar is placed on the screen that allows the value of the pressure command to be set interactively. Adjusting the slider bar allows for the animation of the design process. The current hardware can render a scene at a maximum of 30 cycles per second. For moderate size problems, GGT can solve the system of equations, update the applied load and render the image at about 10 cycles per second. At these speeds the animation of the parameterized commands seems completely interactive and is satisfactory to use. All user command arguments can be adjusted dynamically in this fashion to parameterize a design.

6.1.a) Program Structure

The general structure of the program has been designed to allow for the continuous simulation of the energy equations. All I/O is treated as interrupt to the basic simulation that never ceases. The general structure of the program is shown in Figure 6.1.

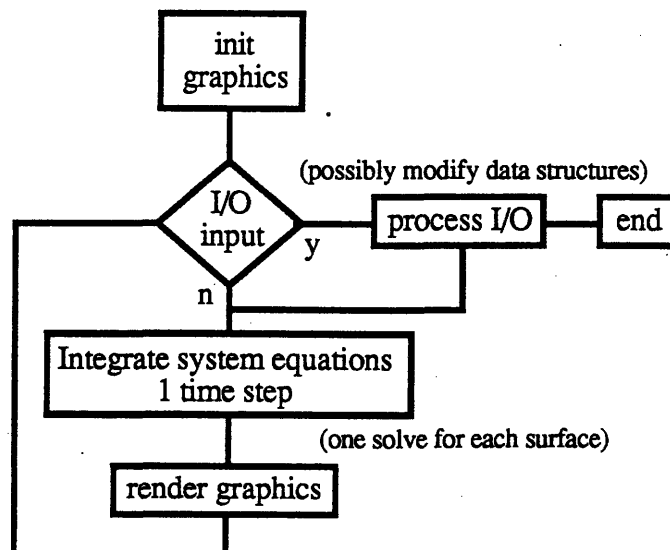


Figure 6.1) Program structure to support interactive simulations

The system of equations is solved continuously. Whenever the user chooses to modify the environment be it forces, loads, constraints, or material properties, the surface will begin to respond immediately. Surfaces may come to rest in equilibrium configurations, but the solver never stops running.

The solver consists of two parts, a problem building piece and a problem solving piece. The bulk of the user commands will not require the rebuilding of the system equations. However, whenever the user modifies the constraint state, by freeing or fixing a surface point, or changes the material properties of a point, the system equations need to be rebuilt and refactored. Since the bulk of user input consists of modifying constraint locations and applying loads, this separation of the solver results in a considerable savings in iteration time. The time to build and factor an equation set is greater than the time to solve it.

6.1.b) A Deformable surface Data Structure

The information required for a finite difference surface can be broken down into the following general classes;

- o Geometry
- o Solver data
- o Constraint data
- o Load data

The geometry consists of a set of points. Each point needs to know the following;

- o current location
- o previous location (needed for the time dynamic simulation)
- o α matrix (array of stretch resistances)
- o β matrix (array of bending resistances)

The points are stored in a two dimensional array. The index location of the point in the array defines its uv location. All the points are restricted to lie on a regular grid of the uv space. This constraint is required by the finite difference solving scheme that has been used for this program.

The solver data consists of a system matrix, in either its pre or post factorization state, and an applied load vector. The size of these matrices is determined by the number of free points in the surface. When the number of free points in a surface is n , the size of the

system matrix will be $[nxn]$ and the size of the applied load vector will be $[nx3]$ since each load like the solution is defined in x , y and z . The system matrix does not have to be part of the data structure since it is completely derivable from the geometry, constraint and load data. However, including it in the data structure makes it possible to reuse the factored system matrix during the simulation loop resulting in a considerable saving in time for each time iteration of the solution. Whenever the user changes the constraint state, the uv unit square grid density, or the α or β matrices the system equations have to be rebuilt and refactorized.

GGT only supports two kinds of geometric constraints, the point location constraint and the periodic boundary condition. The constraint status of each point, either free or fixed, is stored with each point data structure. The system equation building algorithm uses this information to directly build the constrained state equations. The algorithm is structured in this manner to save the time of having to build the full set of equations only to have to then reduce them to enforce the constraints. The state of the periodic boundary condition is saved by two bits. Either the u sides are periodic or the v sides are. Once again the system build algorithm is sensitive to these parameters and appropriately maps the finite difference template across borders as needed.

The finite difference template is not saved as data, but instead has been hard coded into the equation building algorithm. Storing the template with the surface would greatly generalize the range of the solver and could be exploited to marry together shape and analysis packages, but in this program only one template was used.

The load data in GGT was stored as a unique forcing vector acting on each point in the solver. This approach limited the applicability of parameterizing shape with loads. See the ECS program description for a simple and superior technique for load management.

In addition to this basic data, each surface also records enough information to define how it is to be rendered and positioned in space.

6.1.c) Sample Deformable Applications and Designs

6.1.c.1) A Deformable Surface Design of a Goblet

As an example of the geometry that can be made, the look and feel of the program, and the ease at which shape can be generated, this section presents a step by step account of the generation of a goblet.

Start with a surface that has been divided into a [12x17] grid of points subject to a periodic boundary condition in the u direction. This surface has also been constrained so that the top and bottom row of points lie on circles both centered on the z -axis. Only the constrained points on the edges of the circle have been placed by the user. All other points are free to select their positions based on the applied loads, their neighbors positions, and their desire to minimize the surface energy. Figure 6.2 shows the resulting shape as a set of points in space, a grid line drawing and a shaded interpolated surface. The shape is hyperbolic because the surface is responding to its α weighted term which tends to minimize the surface area.

The upper left corner of the GGT window consists of a plot of the uv parametric plane. Its included in the interface so that particular points in the surface can be easily selected using a mouse. Color coding of the points in the surface report the state of the point. Points are either fixed or free, and either selected or unselected.

Additional constraints are applied to make the surface more interesting in the center of the goblet. The commands;

```
shape(7,2,1,-3)    (equation set is rebuilt)  and
shape(9,2,3,0)     (equation set is rebuilt)
```

cause the 7th and the 9th rows of points to be placed evenly on squares centered on the z axis with cross diagonal sizes of 1 and 3 units at the $z = -3$ and $z = 0$ heights. Geometric constraints enforce explicit shape requirements and require the geometry input to be complete as in a typical CAD package. Figure 6.3 shows how the goblet has changed its geometry to accommodate the local change. Applying these constraints is equivalent to attaching a deformable surface to more and more complicated character line sets. The surface naturally blends the shape between the squares and circles in a smooth manner. The final shape of the goblet is achieved with sculpting.

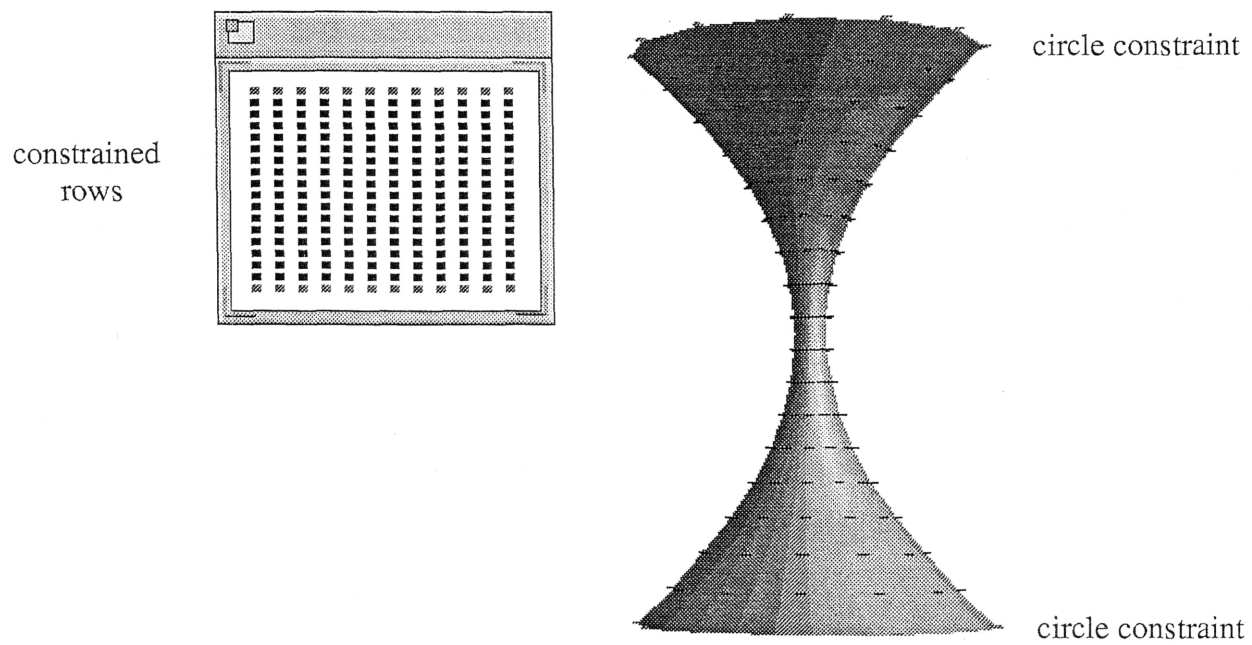


Figure 6.2) Goblet shape after skinning initial character line set

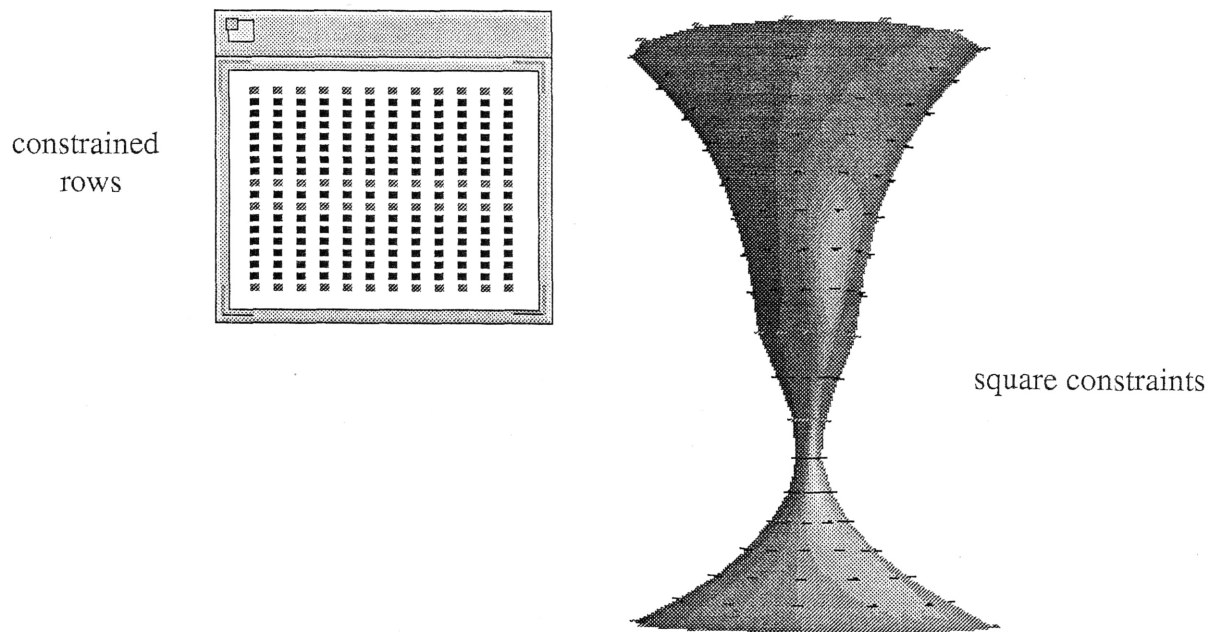


Figure 6.3) Goblet shape after skinning augmented character line set

The command, `tsens(1)`, is executed so that GGT knows to add future loads only to those points that have been preselected in the uv window. The default for many loads is to be applied to all points in the surface. Using the mouse in the uv window all the points above the square constraints are selected interactively. Typing the command `pr(?)`, brings up a pressure slider bar which can be used to sculpt the surface interactively. Interactive manipulation of the slider bar varies the amount of pressure applied to the selected points. The sculpting continues until the bowl of the goblet achieves a pleasant looking shape. Figure 6.4 shows how the applied pressure on the top section has modified the shape.

Now the stem of the goblet is sculpted. Once again the uv window is used to define a reasonable set of points to receive the pressure loads. The bottom two rows of free points are selected and the top section of points is unselected. Pressures with negative magnitudes are applied to the bottom of the goblet to cause the stem to become lighter. Not surprisingly, people's aesthetics are very sensitive to small changes in shape. A little sculpting and the weight of the stem is reduced to produce a shape more suggestive of a wine glass instead of a water glass, Figure 6.5. During the sculpting phase, loads are applied to regions of the goblet, but all the unconstrained points in the system are free to respond. Order is maintained by the energy minimizing algorithm. All the sculpted shapes were acceptably smooth. Sculpting helps to rapidly find the one shape, out of an infinite selection of smooth shapes, that meets the current design needs and satisfies personal taste.

Just for fun, Figure 6.6 shows the goblet after a large point force has been applied to the surface to cause it to collapse. Note that even though the forcing functions are highly discontinuous, the resulting surface still struggles to maintain its fairness.

The speed at which the goblet is made is limited by how quickly the commands can be issued. The character lines were created in 4 commands, 2 for the outer circle shapes and 2 for the inner square shapes. The system was parameterized with 2 more commands, `tsens(1)` and `pr(?)`. The final shape was defined by sculpting which was executed as a search of the parameter space consisting of the pressure magnitude. The sculpting required the selection of regions in the uv command which was accomplished in 3 more mouse driven commands. Sculpting is interactive and animated and the pressure magnitude can be modified at about 10 times per second. This total design sequence can be re-executed in less than a minute. The original design session, including the selection and placement of the character lines, the selection of the load parameterization and a good amount of alternative shape exploration took about 10 minutes.

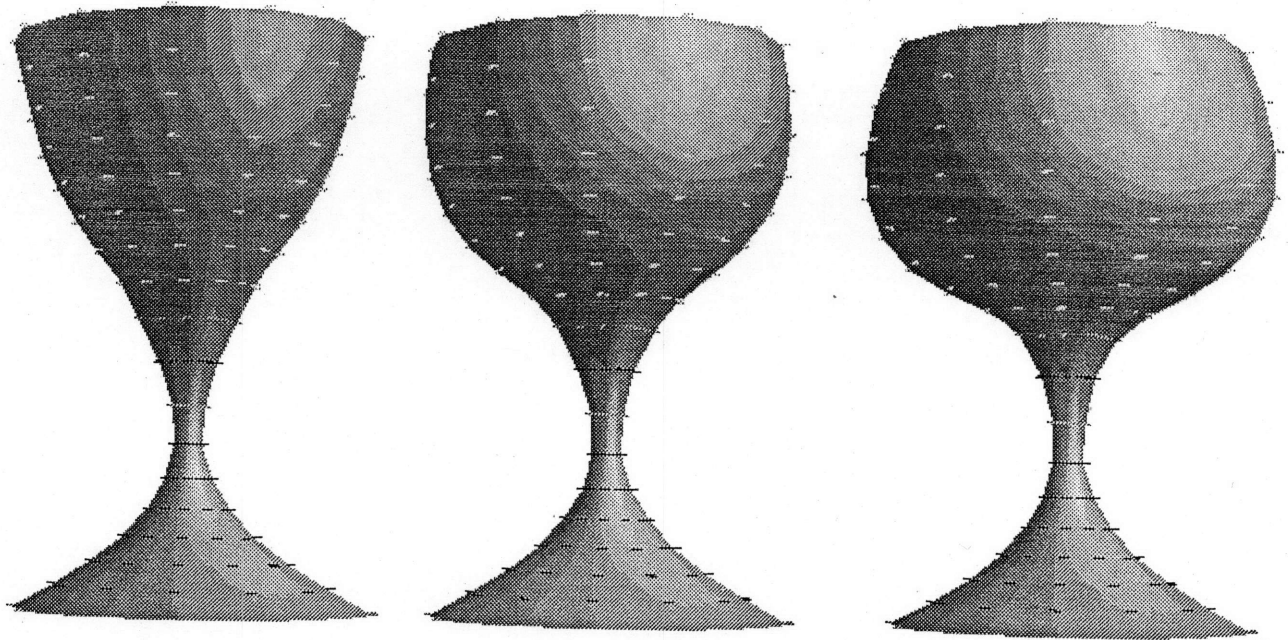


Figure 6.4) Sculpting on the body of the goblet

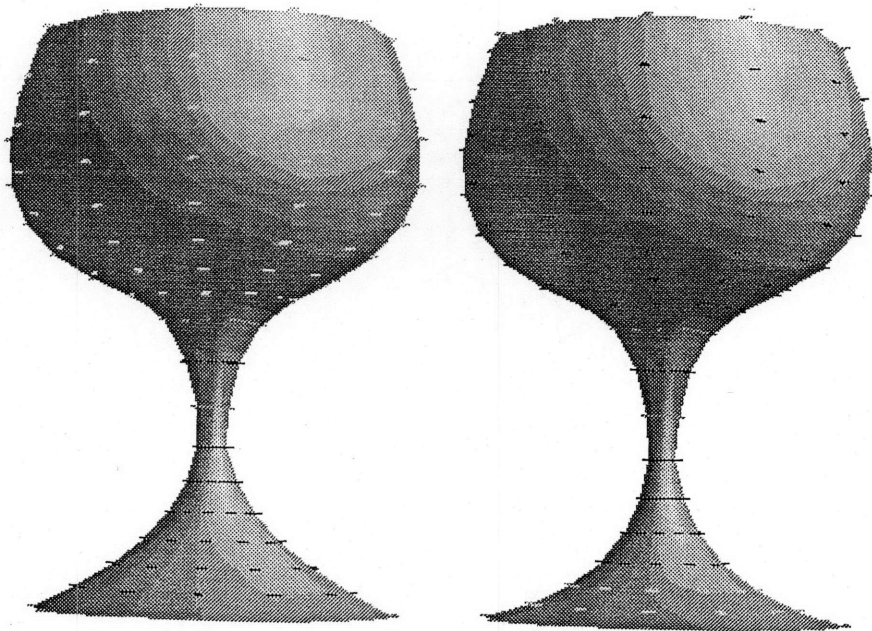


Figure 6.5) Sculpting on the stem of the goblet

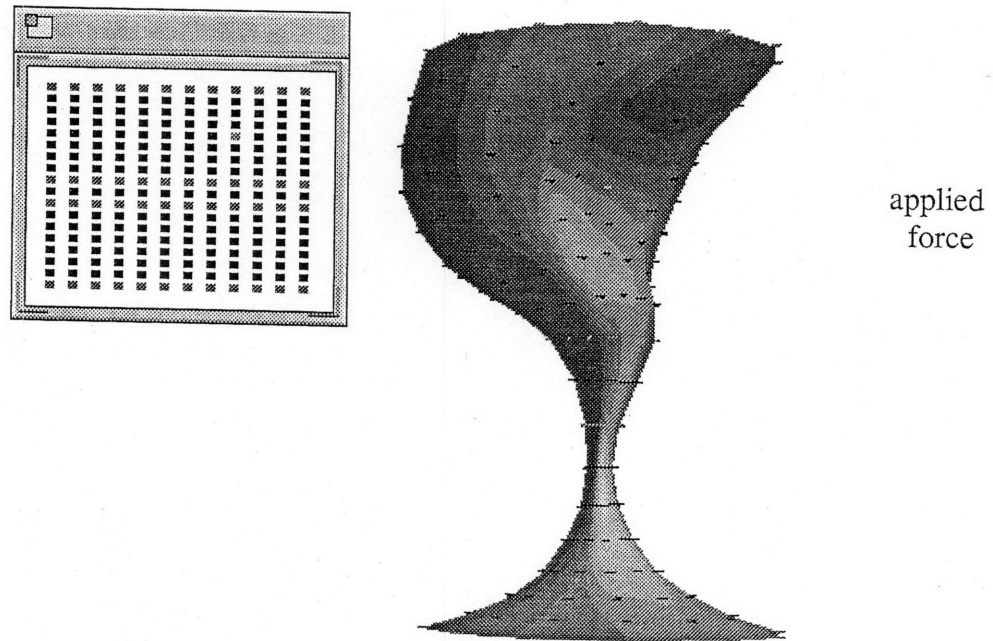


Figure 6.6) A point force dents the goblet

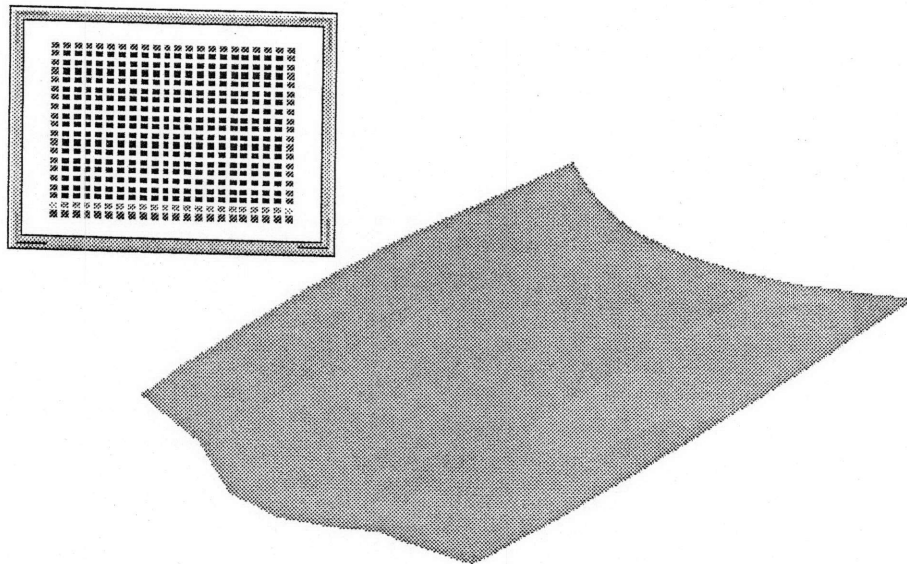


Figure 6.7) An outer body panel for an automobile

6.1.c.2) The Design of a Car Hood Inner Body Panel

The GGT program was used to demonstrate the possibility of designing inner body panels for automobiles. Inner body panels are sheet metal pieces that are formed by stamping and are attached to outer body panels to increase their stiffness. Inner body panels are the rib like structures that can be seen in the hood of any car when the hood is raised.

The design requirements of the inner body panels are that they attach to the outer body shape, increase the stiffness for a minimum of weight, and not interfere with the components on the inside of the car. The proposal to use deformable surfaces to accomplish this design task is presented by example as follows.

The starting point is the shape of the outer body panel, generated by deformable surfaces or in any other manner. Figure 6.7 shows an outer body panel shape that was made by constraining the edges of the geometry to follow character line data sent from ford motor company. The interior shape was made by sculpting forces. The uv window is used to select regions of the outer body panel where the rib like structure can be added.

All points out of the ribbing area are fixed in space. The points in the ribbing structure are then subjected to pressures until reasonable size ribbing is made. Alternatively, ribs can also be made by offsetting some of the points in the surface an appropriate distance and using the others to create the necessary blends. Figure 6.8 shows a series of increasing rib sizes to produce the inner body panel design.

Deformable surfaces can also be made to automatically satisfy the non-interference requirement for the ribs in the panel. Deformable surfaces respond to their environment through forces. Surfaces can be taught to avoid intersecting other surfaces by placing a repulsive force field around each surface. Figure 6.9 suggests how this might be used to help the inner body design problem. The deformable surface in Figure 6.9 is the square. Imagine that it represents one small section of the bottom of an inner body panel rib. As pressure is increased it is forced closer and closer to the ellipsoid which might represent an air filter on a car. When the deformations are large enough to cause intersection the repulsive force field makes the surface deform to avoid the intersection of the car air filter. In this manner low level design rules can be enforced directly by the behavior of the deformable surface freeing the user to work on other design considerations.

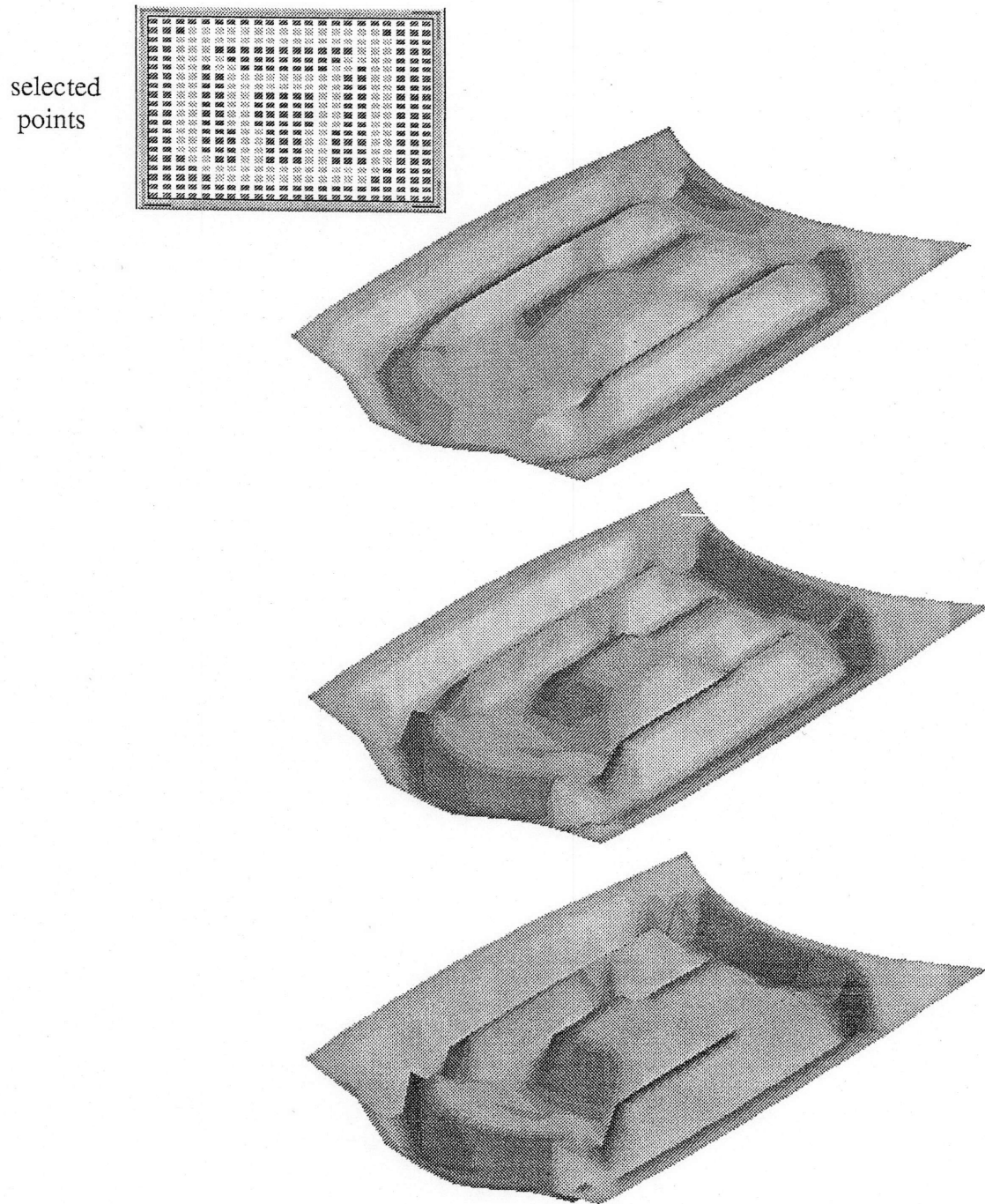


Figure 6.8) Design of the matching inner body panel

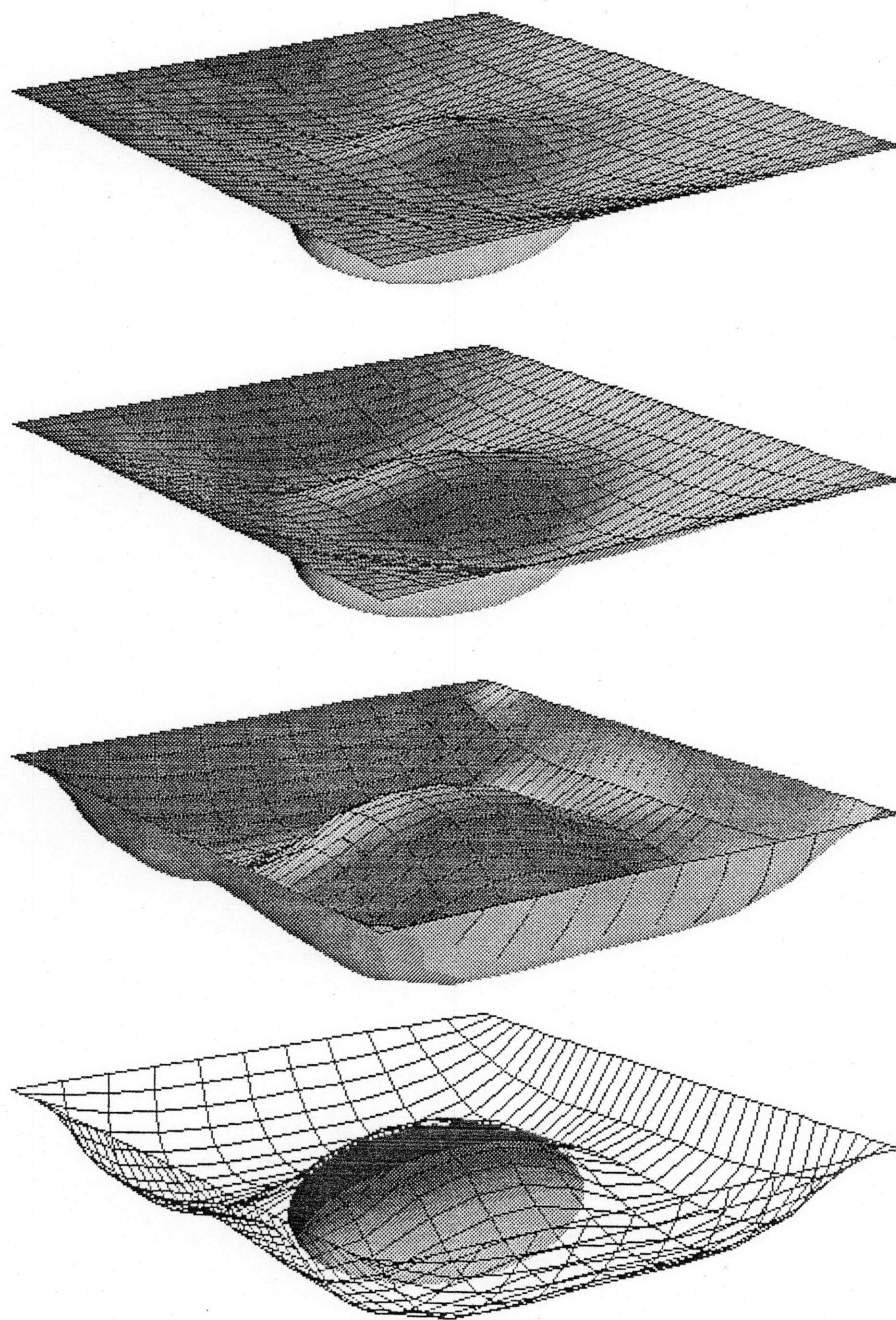


Figure 6.9) A surface avoids an object when sculpted with increasing pressure

The GGT experiment was very successful. It demonstrated the strengths and the weaknesses of the finite difference formulation of the deformable surface design. Primarily we conclude that deformable surface design can be made to work. We also conclude that this formulation of the problem is inadequate due to limitations on the manufacturability and topology of the class of shapes that can be generated. The shape representation remains a set of points in space that map to the unit square in the uv parameter domain.

6.2 A Finite Element Program for Deformable Surfaces, ECS

The program ECS is a finite element based implementation of the deformable surface equations. Any number of triangular elements can be made to tessellate any polygon shape in the uv domain. Each element can be assigned a load, can have any shape and size in the uv plane and maps to a continuous region in the xyz space. Each element has its own α and β matrices, but in ECS these are currently set to the same value throughout the surface. The finite elements have been made to support the free, pinned, fixed and hinged boundary conditions needed for the ShapeWright skinning step. Adjacency constraints such as a periodic boundary condition could be implemented but have not. With such constraints cylindrical and closed surfaces can be designed.

The ECS program has been created under the same interface as GGT and enjoys the same opportunities to parameterize the input for animation. ECS also runs under the same continuous simulation program structure as GGT.

6.2.a) A Finite Element Deformable Surface Data Structure

The information required of the finite element deformable surface falls into the same categories as the finite difference data structure;

- o Geometry
- o Solver data
- o Constraint data
- o Load data

The ShapeWright surface element data structure is shown in Figure 6.10.

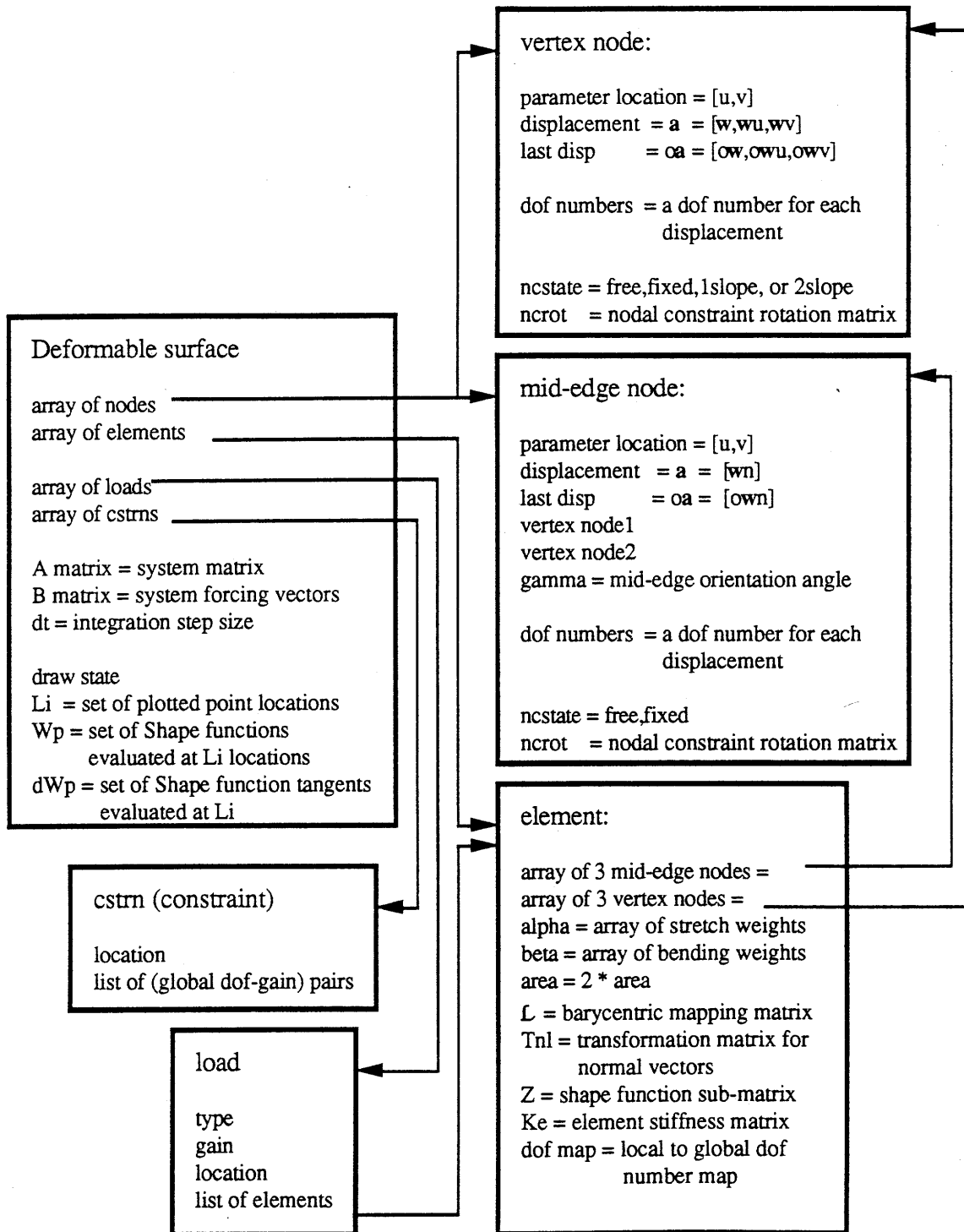


Figure 6.10) ShapeWright deformable surface data structures

The geometry of a deformable surface consists of a set of nodes and a set of elements. Each node is either a vertex node or a mid-edge node. The geometric information at each

vertex node consists of the node's xyz space position, the node's xyz space surface tangents in the u and v parametric directions, and the node's location in the uv space. The mid-edge nodes only store the value of the xyz space surface tangent in the direction normal to the element edge in the uv plane. Each mid-edge node data structure points to two vertex nodes. The node's position and edge slope angle are both calculated from these vertex node positions. Each mid-edge node is assumed to be placed half way between an edge that stretches between the two pointed vertex nodes. Moving the uv position of a vertex node will always move all the mid-edge nodes that point to it. All xyz space location data is stored for the current and previous time step to support the dynamics in the time integration of the system equations.

An element consists of a list of 6 nodes; 3 vertex and 3 mid-edge nodes. The position of the 3 vertex nodes completely defines the Barycentric transformation for the element saved in terms of the \mathbf{B} matrix, the \mathbf{T}_m transformation matrix used to calculate tangent derivatives in the edge normal directions and the Zienkiewicz \mathbf{Z} matrix needed to extend the 9 degree of freedom shape functions into the 12 degree of freedom shape functions. These matrices are all stored with the element to increase the speed of calculating the element stiffness matrix, \mathbf{K}_e , and of rendering the element in its deformed xyz space shape. The element stiffness matrix is also stored to save time in assembling the system matrix equations.

Constraint information affects the nodes, the building of the system matrix and the content of the system forcing vector. Each constraint has a data structure. In this data structure the value of the constraint is stored e.g. the position in space of a fixed node, and a column of gains to be added to the forcing vector. This column is made by concatenating the columns of all the effected element stiffness matrices. At run time each constraint adds a set of values to the forcing vector that are weighted by the value of the constraint. Modifying the position of a fixed point changes the value of the constraint gain in the associated constraint data structure that modifies the terms of the forcing vector. In this manner constraints can be enforced as quickly as loads can be applied. This technique supports the animated interactive manipulation of constraint locations in the xyz window.

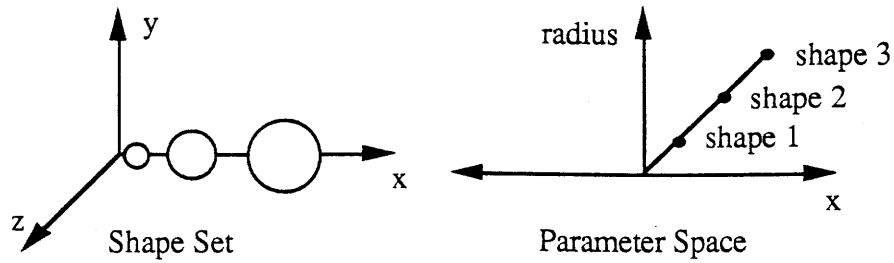
Some constraints require that a vertex node be rotated into an appropriate space. Each vertex node has a rotation matrix that defaults to the identity matrix when unconstrained and is set to the appropriate transformation matrix when constrained. All nodes store their constraint state so that the system equation building algorithms can act accordingly. The vertex node constraint state can be one of, FREE, FIXED, 1SLOPE, or 2SLOPE indicating

whether the node is free, has a fixed xyz location, or has a fixed location and 1 or 2 fixed tangent directions as well.. The mid-edge node constraint can be one of FREE or FIXED.

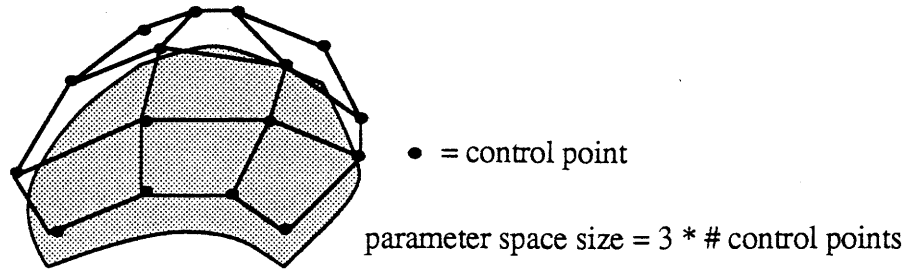
Loads affect the content of the system forcing vector. Each load has its own load data structure to support the parameterization of the surface through the parameterization of the loads. The load data structure records which type of load is applied, the magnitude and location of the load and a list of the affected nodes and elements. At run time the load structures are used to generate the set of forces that are applied to the surface at each time iteration. The user can manipulate the gains and locations of a load data structure dynamically. Elements and nodes can be added or removed from the load by command based on their select state. Each node and element stores its current select state to simplify this and other I/O activities.

This scheme of storing loads as separate entities allows the user to parameterize any number of surface shaping effects simultaneously. Any desired load parameters can be assigned to a slider bar which can then be used to sculpt shape as an interactive, dynamic search of a parameter space. The advantage of this particular parameter space is that it can be designed and modified to simplify the control of the shape. In the previous goblet example, the loads for sculpting the bowl and the stem of the goblet could now be placed on separate slider bars and manipulated on the screen in any order to achieve more rapid flexibility in design. The goblet has been parameterized by two variables, one that controls the bowl size and one that controls the weight of the stem. The flexibility of the approach allows any number of alternative parameterizations. For example, the size and locations of the square character line constraints could also be used to parameterize the goblet. Figure 6.11 shows how the load concept and the energy minimization algorithm can make the controlled modification of a free-form surface as simple as changing the radius on a sphere.

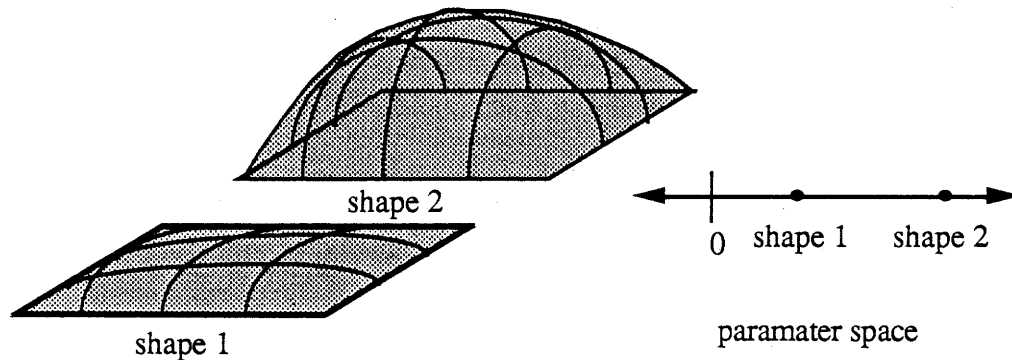
The solver information, like the GGT program, consists of a system stiffness matrix and load vector. Additionally, a mapping which assigns each degree of freedom of all the nodes a unique system degree of freedom number is saved. To save time during the building of the system stiffness and load matrices each element stores a local map of how the element degrees of freedom numbers line up with the system degree of freedom numbers.



a.) Sphere: Parameters = location(x) and radius



b.) Control Point Surface: Parameters = locations of control points



c.) Deformable Surface: Parameter (user defined and modifiable) = pressure

Figure 6.11) User leverage through parameterization of shape

6.2.b) Some Continuous Deformable Surface Applications and Results

6.2.b.1) The Sample Surface

Figure 6.12 shows a sample ECS surface consisting of five triangular elements. The parametric shape is shown in the plot of the uv window in the upper left hand corner. It has been selected to be a pentagon so that it would not look like a square. In general it can be any polygon shape. The resulting xyz deformed shape is made by pinning the five outside edges to xyz point locations. Figure 6.13 shows the surface in different views and

in different rendering modes to try and convey the extreme fairness of the surface. In these static images, destined to be reproduced in black and white, the best plot for conveying surface shape is the contour line image. Additionally, some plots have been made where the rendering of some of the elements have been suppressed so that the shape of the internal edges can be seen better.

This shape can be modified by applying loads, moving the constrained points, changing the constrained state of the edges and changing the original uv shape. Figure 6.14 shows some of the shapes that were generated in that fashion. The images in the column of Figure 6.14a were made by moving a control point location of an edge in the pinned and hinged fixed states. The sequence of images in the column of Figure 6.14b were made by applying an increasing pressure to the original pinned shape. The angles of the original parametric shape are preserved in the object's deformed shape. A uv pentagon shape will have five corresponding corners in its deformed shape. Figure 6.15 shows how the uv shape of the pentagon can be manipulated into a triangle to yield a surface that interpolates the original five points with only three corners.

6.2.b.2) The Ford Door and an Interface to Artifact Analysis

The ECS program was used to model a shape suggestive of a Ford car door as an exercise in more controlled geometry building as shown in Figure 6.16. The general shape of the door was made by applying hinged constraint conditions on all the outer edges and using a pressure to generate a nicely convex shape. The rib in the door panel was made in two steps. First the appropriate edges were constrained in advance so that the rib deformation would be controlled and isolated from the rest of the door shape. To do this the edges bordering the rib were all fixed. This isolated the deformations in the rib region from the rest of the door. Then the edge running along the center of the rib was constrained as a hinge. Once the appropriate constraints were in place the actual shape of the rib was made by moving the hinged edge a fixed distance normal to the surface. In this manner the doubly curved shape of the door is reflected in the geometry of the rib.

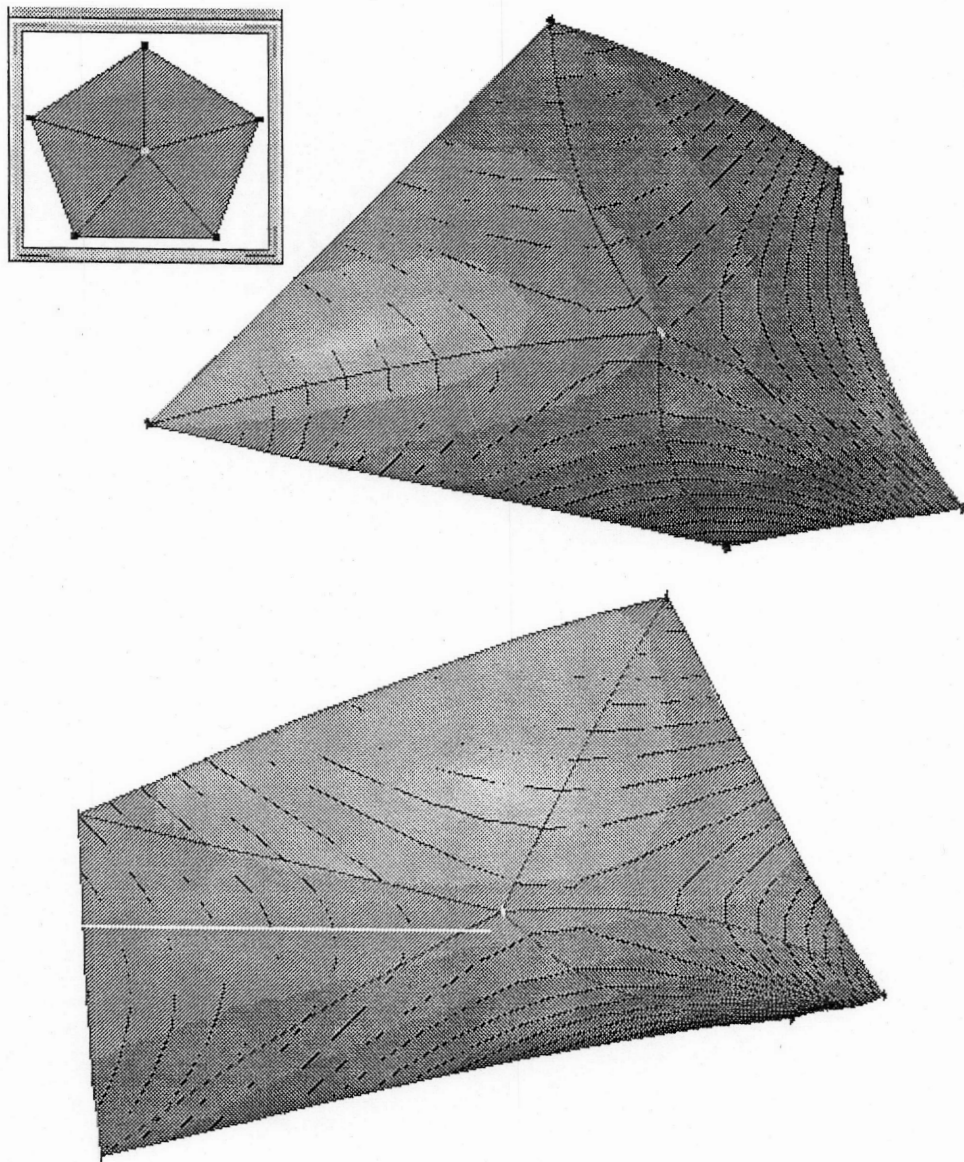


Figure 6.12) A 5 element ShapeWright surface constrained to interpolate 5 point locations

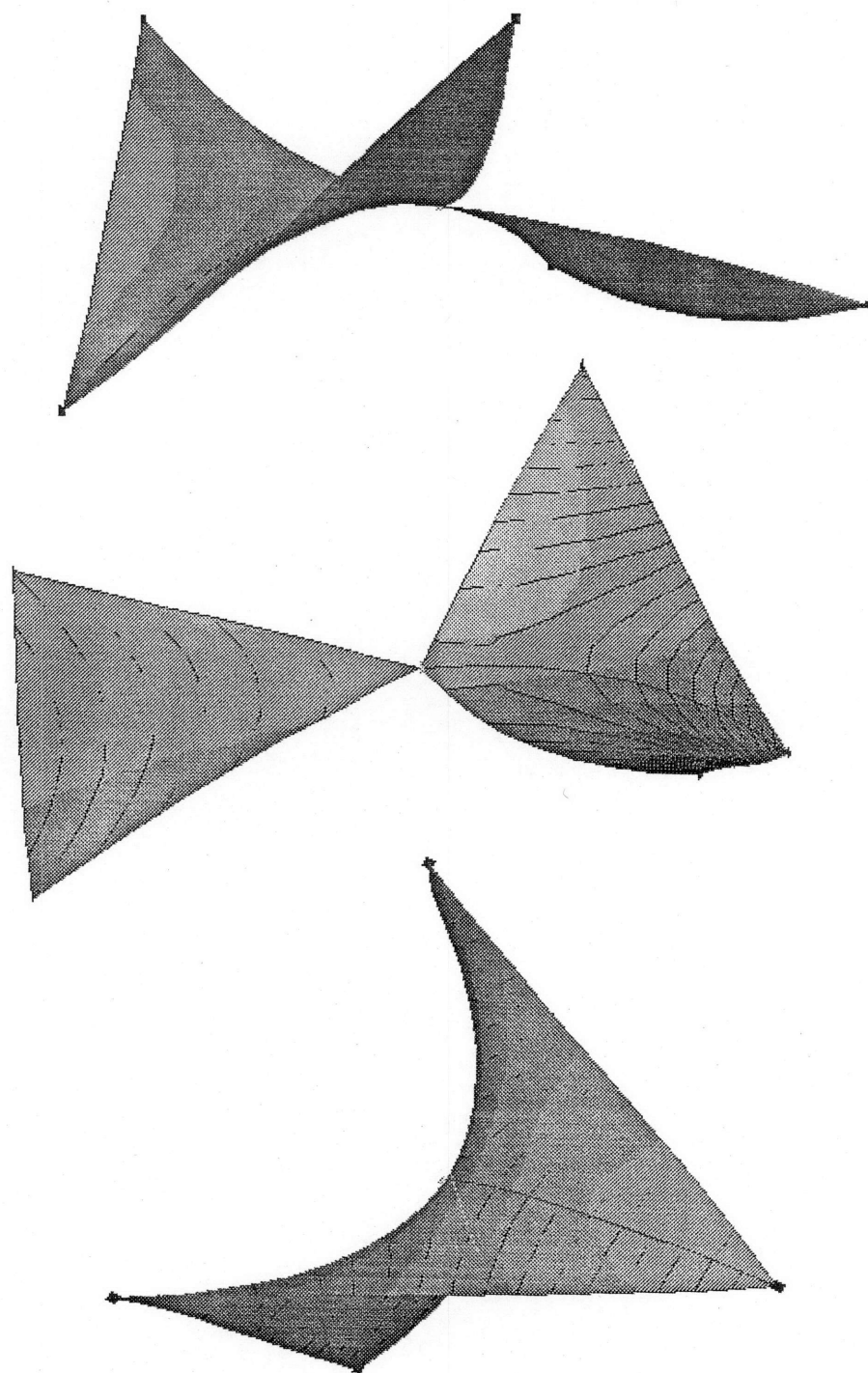


Figure 6.13) Views of a shape

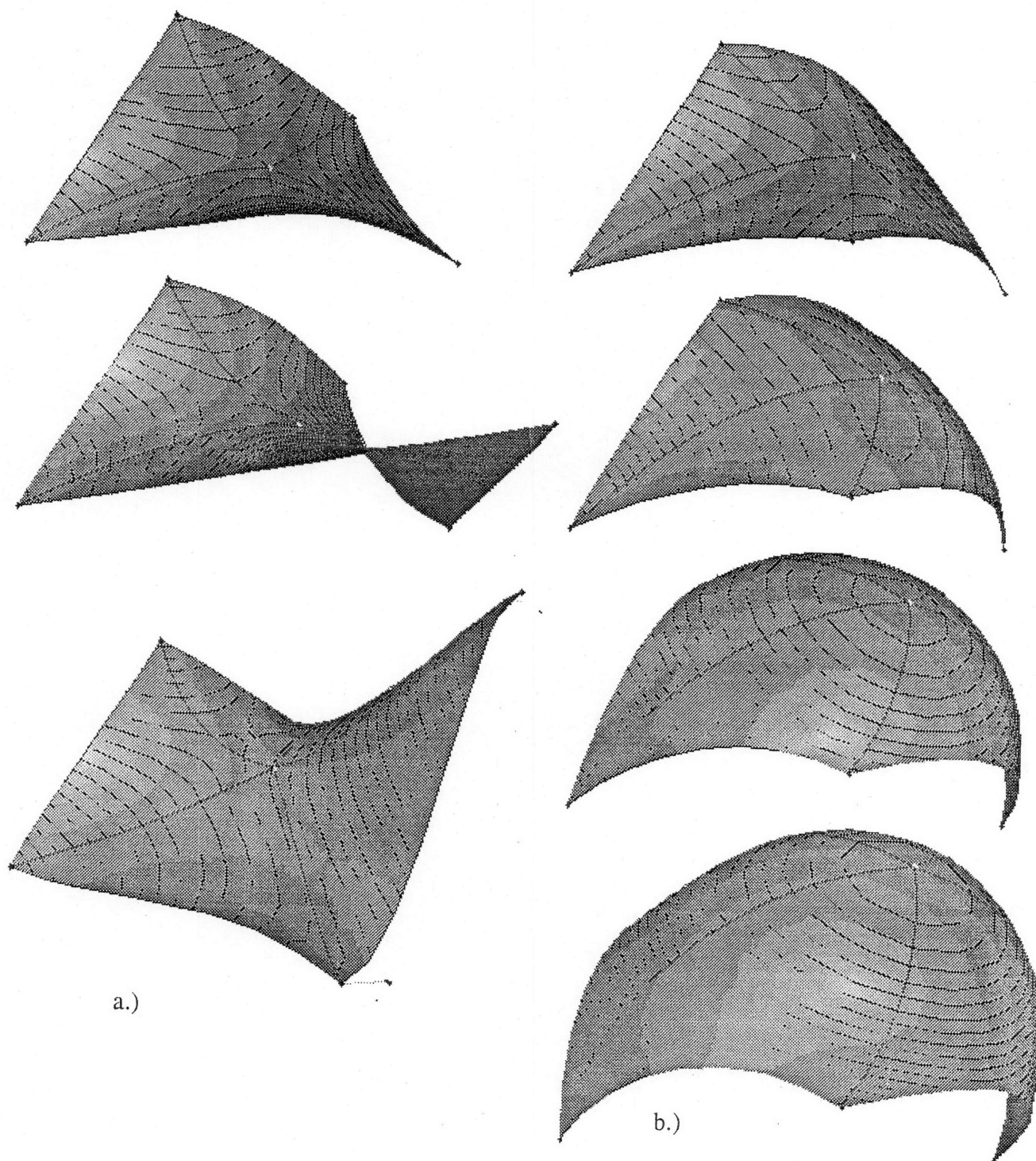


Figure 6.14) Variations on a theme of shape
a.) changing point and edge shape constraint locations
b.) sculpting with increasing pressure

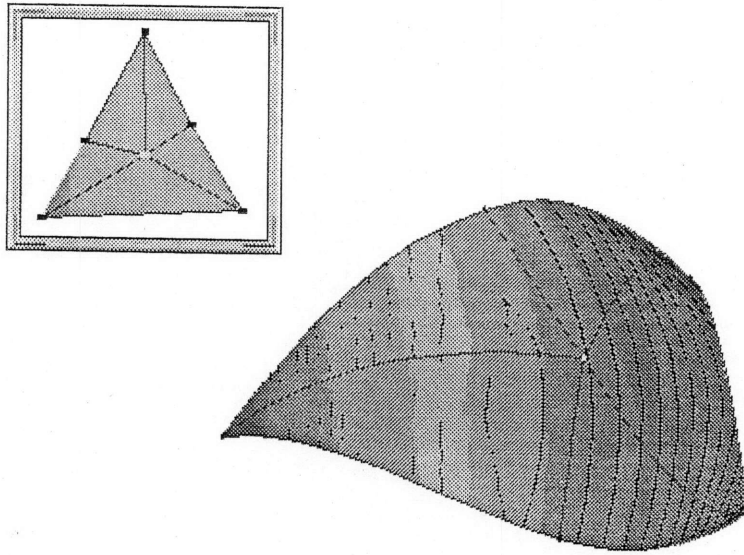


Figure 6.15) 3 corners in uv shape map to 3 corners in 3D shape interpolating same 5 points as Figure 6.12

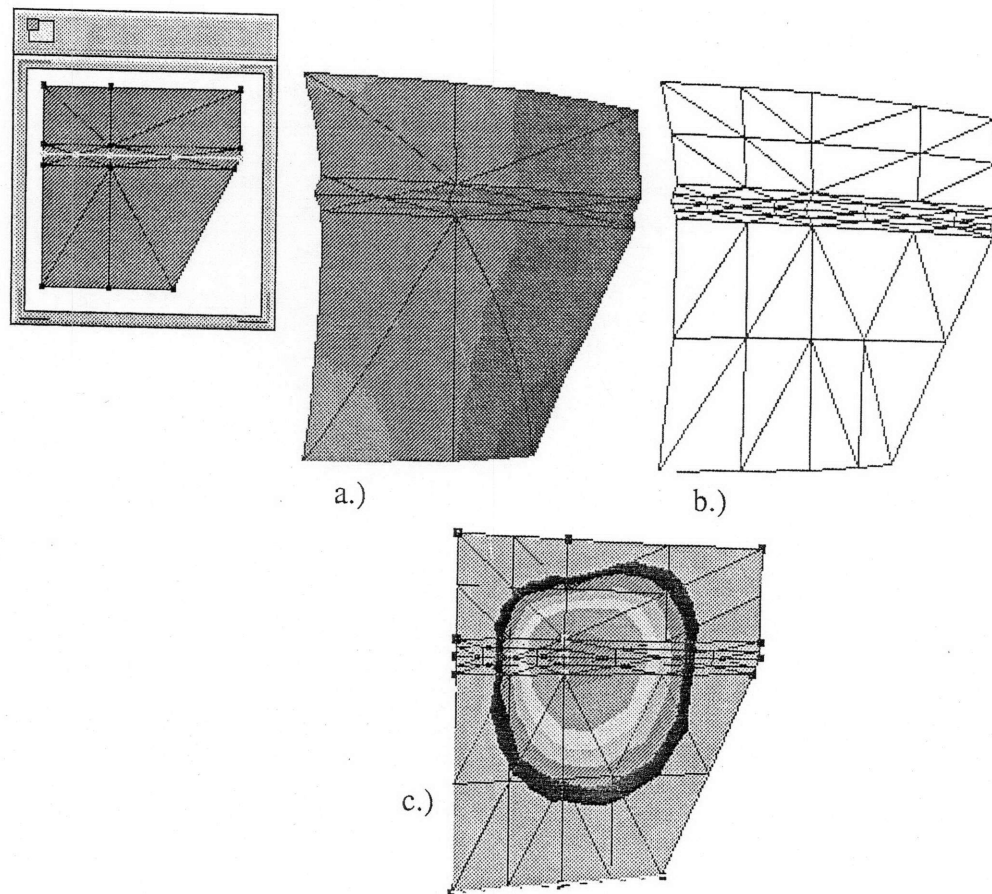


Figure 6.16) The Ford door; interfacing shape and analytic modeling

- a.) finite element shape model
- b.) automatically generated finite element application model
- c.) constant displacement contours resulting from a point load

One of the advantages of using finite elements for modeling shape is that the same finite element model can be exploited for analytic applications. The final images in Figure 6.16 show a linear elastic finite element model of the final door panel shape. The material properties for this analysis were assumed to be that of steel. The final image shows the results of a displacement analysis of the steel panel subject to a applied point load. The contours show the magnitude of the displacement from the original shape. Adding the linear elastic finite element analysis capability to the ECS program was done by Dr. Hiroshi Sakurai of the MIT CADlab. In industry today the generation of an acceptable finite element model from a shape model is a labor intensive task. This work identifies an approach which automatically integrates the generation of shape and analysis models.

6.3 A Finite Element Program for Deformable Curves, CUBIC

The program CUBIC is a finite element based implementation of the deformable curve equations. In Cubic, any number of finite element curve segments can be connected together to generate a deformable curve. The lengths of each element can be set independently so that areas of rich geometry can be modeled with several elements while smooth and simple geometry can be modeled with a few elements. These curve segments have been implemented so that they can support fixed point and tangent constraints at any of the nodes between elements.

The CUBIC program is written using the same interface and continuous simulation structure as ECS. In fact these two programs have been implemented in an object oriented fashion so that both primitives can be used in the same package.

6.3.a) A Deformable Curve Data Structure

The deformable curve data structure is a simplification of the deformable surface data structure. The information storage and usage is the same for loads, constraints and the solver. The only difference in the storage of information is in the geometry. The deformable curve is a set of order nodes. Elements are presumed to exist between each pair of neighboring nodes. Each node stores its position and curve tangent in the xyz space and the distance between it and its neighbor in the uv space. As before all xyz location data is stored for the current and the previous time steps to support dynamics in the time integration of the system equations.

6.3.b) Some Continuous Deformable Curve Applications and Results

6.3.b.1) A Simple Curve Avoiding an Obstacle

A very simple 3 element curve with fixed end point constraints is shown in Figure 6.17. An obstacle, modeled as a $1/\text{distance}$ repulsive force field, has been added to show how the deformable curve can be made to react to its environment and how smoothly it deforms. Figure 6.17b shows the object rendered as a set of straight line interpolations to show clearly where the finite elements connect.

Figure 6.17c shows the curvature report for the curve. A deformable curve's fairness is evaluated by examining its curvature. The most effective curvature report for a curve is the curvature comb. In such a curvature plot a vector is drawn in the normal direction to the curve whose length is equal to the local curvature. The resulting drawing, makes it easy to see how the curvature is changing over the length of the curve and highlights points of inflection.

Note in Figure 6.17c how the curvature is not only continuous between elements but also exhibits a strong degree of C^1 continuity. Although the shape basis functions enforce only C^1 continuity between elements, the energy minimization algorithm attempts to find C^3 solutions to the input constraints and loads. The final image shows how the same shape is maintained even though the grid has been refined on the left hand side of the curve. Allowing for grid refinement is important in modeling complicated shapes where large and small details are mixed together. The small discontinuity of curvature on the right hand side of the curve is due to the curve grid density and the effect of the constraint. Refining the grid on the right side of the curve would allow it to more closely find the actual C^3 shape that the energy algorithm is approximating.

One thought, which was not implemented, was to automate the deformable curve's grid refinement based on the magnitude of discontinuities in curvature between elements.

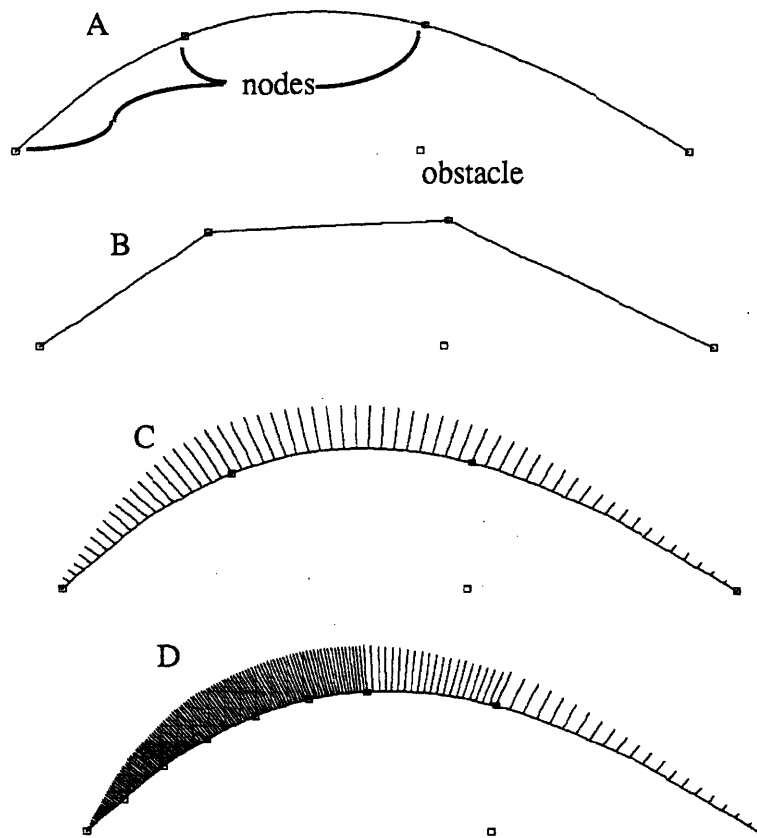


Figure 6.17) Curve deforming to avoid an obstacle
 a.) A 3 element curve deforming to avoid an obstacle.
 b.) Same curve shown as a linear interpolation of the nodes.
 c.) The curve's curvature report.
 d.) An 8 variable length element curve avoiding the same obstacle.

6.3.b.2) An Application of Deformable Curves to Fairing Input Data

The deformable curves were used in an application of fairing hand drawn fully three dimensional curves. Andy Roberts and Dave Stoops have created the 3-Draw program at the MIT CADlab. The program, 3-Draw, using a magnetic detector made by the

Polhemous Company, captures discrete point locations while a user draws three dimensional curves in space. This package has been optimized to assist a user in drawing complicated geometric wire frame models. The geometric primitives of the package included static linear interpolations of the sampled points and the dynamic deformable curves.

The preferred method of use that evolved for developing 3-D models was a three step procedure. First, the basic shape of the curve is captured as a set of sampled points. Second, the position and size of the curve are adjusted to match the model. Finally, the curve is faired using the properties of the deformable curves.

The fairing scheme used in 3-Draw captures the original curve as a set of discrete points. Then the user is allowed to select a sub-region of the curve to be faired. The ends of the sub-region are geometrically fixed in position and slope and all of the intervening point locations are associated with the nodes of a deformable curve. The time simulation is run by the user until either the curve comes to equilibrium or a desired intermediate state is achieved. Iterating through over-lapping sections of the curve allows the user to preserve the desired general curve shape while eliminating unnecessary deviations.

Figure 6.18 shows a curve as originally captured in 3-Draw and subsequently as it is faired. The curve dynamics act to eliminate high frequency shape more rapidly than low frequency shape. The first application of fairing is applied to the whole curve and executed just long enough to eliminate the noise (shown amplified for the example) in the shape signal. The second phase of fairing allows the user to artfully smooth out larger scale disturbances in the curve.

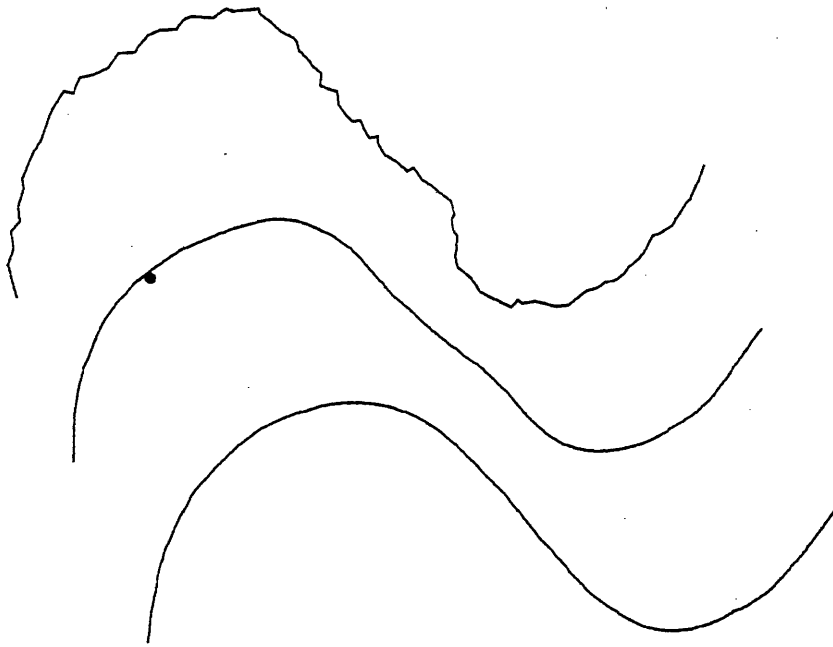


Figure 6.18) Fairing a curve.
a.) Curve as input.
b.) Curve after initial smoothing
c.) Curve after final smoothing

Figure 6.19 shows some examples of the wire frame models that can be built with 3-Draw. Figure 6.19a is a sketch of a winged keeled sail boat and Figure 6.19b is the right front bumper of an imaginary sports car. These models took from 30 minutes to one hour to draw and are fully 3-dimensional.

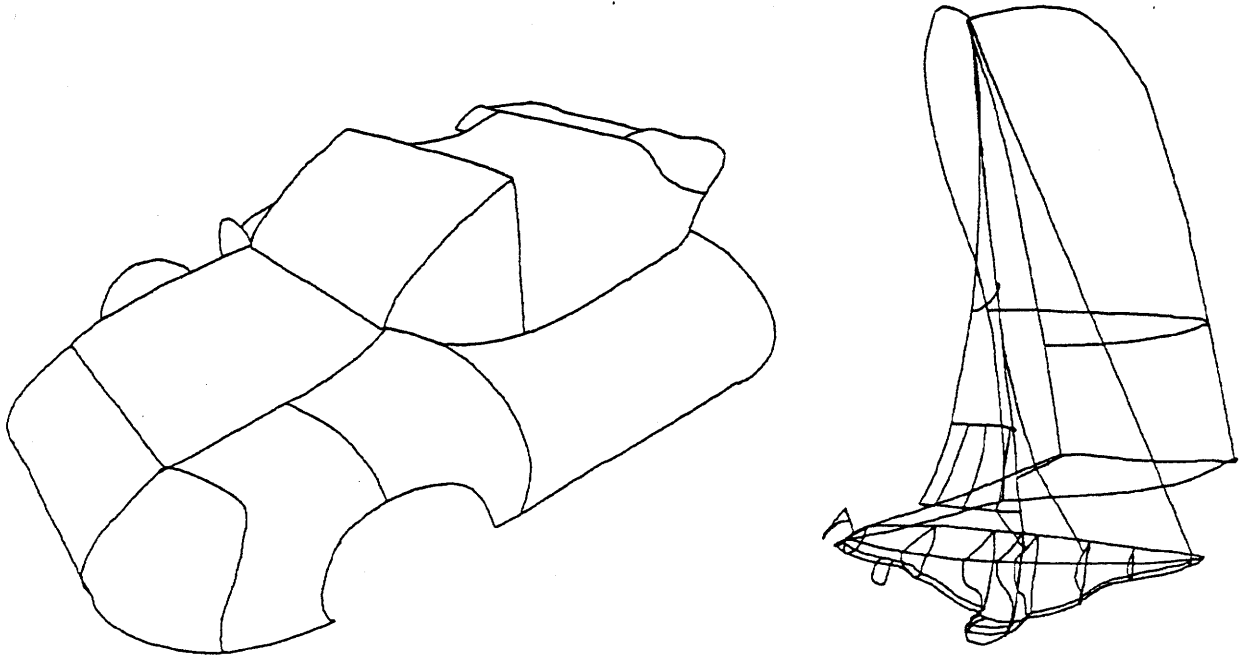


Figure 6.19) Sample 3-Draw sketches (by Andrew Roberts, MIT CADlab)

6.4 Rendering

6.4.a) Shaded Images

Rendering a deformable surface can take almost as much time as solving the energy minimization problem. As such it was important to structure the render loops in such a fashion as to speed up the rendering of the surface.

The Silicon Graphics work station has the capacity to render light shaded polygons in hardware so long as the position and the surface normal direction at each vertex is supplied. The surfaces were rendered as sets of polygons to take advantage of this capability. To do this, a set of points has to be computed that can be used to generate the polygons that represent the surface. Figure 6.20 shows the rendering grid patterns for a variety of precisions.

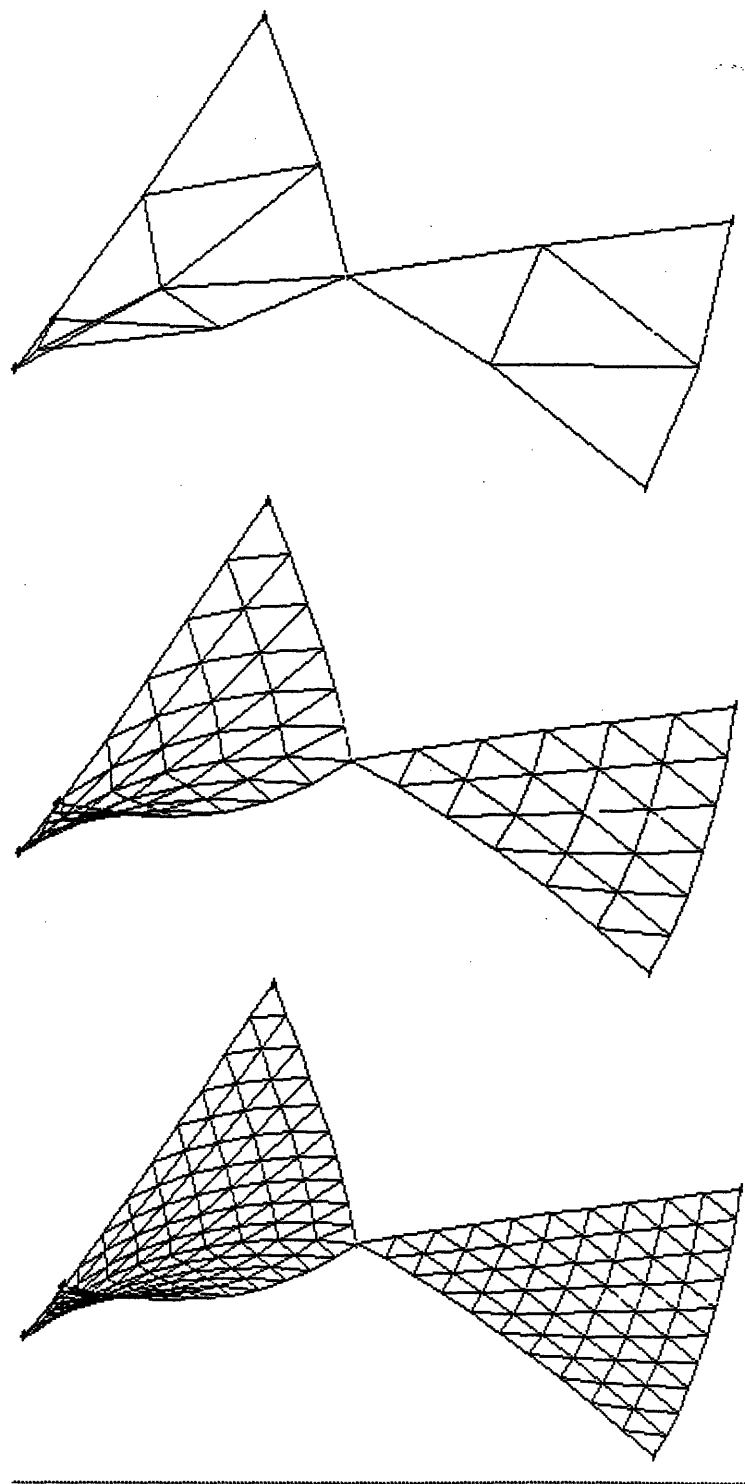


Figure 6.20) Different precisions for a deformable surface rendering grid

As much of the calculation of the grid points is accomplished off line as possible and saved for use during the rendering loop. The shape at a particular uv value is given by;

$$w(u,v) = \sum_i \phi_i(\mathcal{L}, L_1, L_2, L_3) \mathbf{x}_i \quad 6.2$$

Ideally, the ϕ_i functions could be saved at each point to be rendered in the triangles and then during the simulation loop only 12 multiplies and 11 adds need be executed to generate each of the rendering mesh point locations. Similar actions can be taken for calculating the surface tangents needed to calculate the surface normal. This would require $4*12*\text{number_of_points_rendered}*\text{no_of_elements}$ of storage to store the functions needed to calculate shape and the surface tangents.

To save on the amount of data that needs to be stored for rendering, each element is plotted with the same array of (L_1, L_2, L_3) values. The shape functions are rewritten to separate the functions of L_1 , L_2 and L_3 from the functions depending on triangle Shape as ;

$$w(u,v) = \sum_i \phi_{ai}(\mathcal{L}) \phi_{bi}(L_1, L_2, L_3) \mathbf{x}_i \quad 6.3$$

The functions depending on ϕ_{bi} are the same for all elements and are calculated once and stored for use in the rendering cycle of all elements. The $\phi_{ai}(\mathcal{L})$ elements are stored as the \mathcal{L} , T_m , and the Z matrices for each element. The amount of storage needed for the program rendering is limited to;

$$[\text{plot_point_count}]*[\text{max_part_count}]*[4] \quad 6.4$$

where plot_point_count = number of rendering grid points per element
 max_part_count = number of functions needed to describe the shape of the surface
 max_part_count is multiplied by 4 because the first partials with respect to L_1 , L_2 , and L_3 are stored to speed the calculation of the tangent derivatives of the surface.

This scheme has increased the speed of rendering by an order of magnitude with only moderate costs in data storage.

6.4.b) Contour Plots

We have found that the perception of surface shape can be greatly enhanced if the surfaces are rendered as a combination of shaded light models and contour lines. Anomalies in the surface become very evident in the behavior of the contours. In the interest of speed only the contours of the approximate polygon rendering model are plotted and not the actual contours shape. Since the precision of the surface rendering grid can be made to be as high as desired any accuracy of the surface contours can be achieved this way.

6.4.c) Curvature Combs

The fairness of a curve was found to be best displayed with the curvature comb. The curvature comb is made by adding a vector whose magnitude is equal to the local curvature of the curve pointing in the direction of the surface normal to a set of the points of the curve. Curvature combs were implemented so that each element of the curve displayed the same number of curvature points regardless of the length of the elements.

6.4.d) Curvature Plots of the Surface

Surface curvature plots were made by color coding the surface based on the local curvature. These plots were not effective at helping to understand the shape of the object, but did highlight regions of rapid change. This capability, implemented in GGT was not extended to the ECS package.

7 Conclusions and Future Work

Because the requirements for shape are ubiquitous, the potential impact of the deformable shape technology is large. Shape representations are needed for analysis, simulation, manufacturing, and design. The energy based modeling of shape represents a method to greatly reduce the amount of user management required to define and modify shape. Designing with intelligent shapes that automatically seek to satisfy a set of design rules offer the potential to integrate the various phases of design together. The more design rules that the deformable models can be taught by increasing the sophistication of the energy functionals and the range of constraints that can be enforced, the more uses to which this technology can be applied.

In this chapter the general ShapeWright modeling paradigm and the work contribution made in this thesis is reviewed. The conclusions that have been derived in the course of this work are then presented. The final section suggests three general areas to which future work in this field might be applied.

7.1 Thesis Summary

Energy based shape modeling is the combination of parametrically described geometry plus an energy minimization algorithm that acts to give the geometry natural properties. This supplies a means to control shape behavior during an interactive modeling process that can greatly reduce the amount of input effort it takes to build or modify shape. The savings in input effort comes because the energy minimization algorithm directly controls the simultaneous manipulation of the huge number of degrees of freedom needed to parametrically represent free-form shapes while the user indirectly modifies shape with global operators described as forces, material properties and geometric constraints.

This thesis proposes a new paradigm, the ShapeWright paradigm, for the generation of free-form shapes founded on energy based shape modeling. The approach comes from the observation that people have a rich experiential exposure to deformable surfaces. It's easy to imagine folding a piece of paper or expanding a soap bubble with an internal pressure. Historically, such physical surfaces have commonly been employed as the basis for free-form design. Thin wooden splines were used to loft the shape of ship hulls and aircraft

wings. Actual soap films were used to design the roof of the Olympic stadium in Munich, Germany.

The ShapeWright paradigm is a three step procedure for generating objects as shown in Figure 1.2 and reproduced here as Figure 7.1. First, the essence of the object is defined as a set of three dimensional character lines. These are lines which define hard geometric constraints for the shape such as edges and creases. These do not include lines typically used to characterize the interior shape of a surface such as silhouette lines. Then the object is skinned. Imagine dipping the wire frame object into a bucket of soap and pulling it out slowly such that over every face there is now a deformable soap film like surface. Finally, the object shape is completed by interactively sculpting the surfaces with forces. Modifying the shape can be achieved by changing the character lines or the sculpting forces.

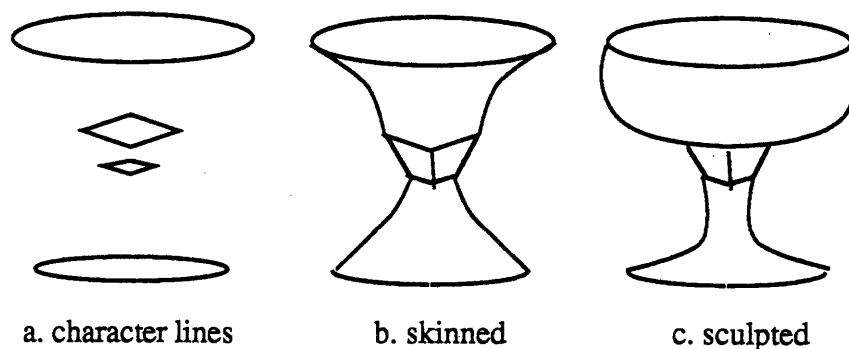


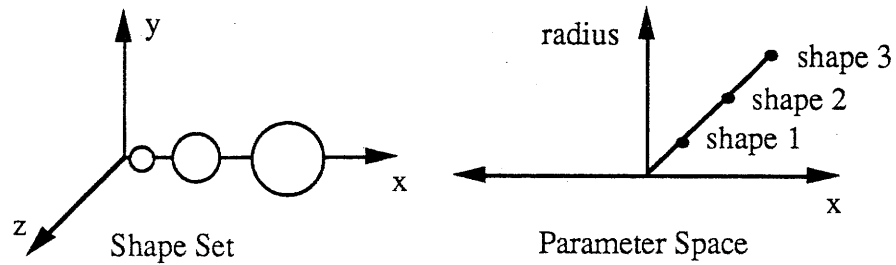
Figure 7.1) The ShapeWright design paradigm

The exciting part of the paradigm, that can support endless innovation, is that the energy functional to be minimized can be designed. In this manner it is possible to engineer surfaces with different natural and desirable properties. Surfaces which minimize their area and distribute their curvature smoothly are built in this thesis to create geometry which is opportunistically fair seeking.

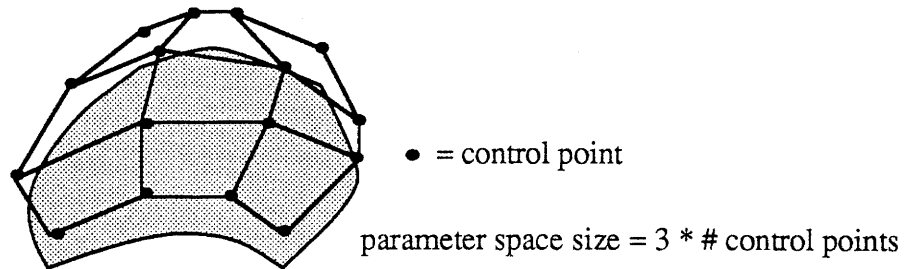
The work contribution of this thesis consists of defining the mathematical foundations appropriate for solving the energy based shape problem in such a way as to support a wide range of topologies, automatic free-form surface manufacture, and interactive speeds. Geometric curve and surface primitives were developed that are appropriate for the ShapeWright shape definition paradigm.

7.2 Conclusions

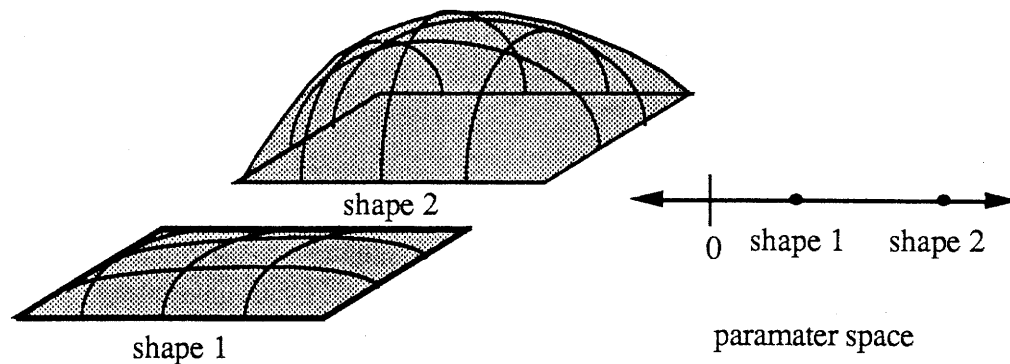
The ShapeWright deformable surface modeling paradigm works well for the interactive definition of free-form surfaces. The notion that shape can be defined indirectly with the use of minimum principles subject to geometric constraints and loads acts to greatly reduce the user input in manipulating the surface. Complicated free-form shapes were defined with very few commands. Additionally, they can be parameterized with the use of loads so that the design of shape can be accomplished as a search in a small user built parameter space independent of the number of degrees of freedom in the surface. The essence of this idea and how it compares to other shape representations is shown in Figure 6.11, reproduced here as Figure 7.2;



a.) Sphere: Parameters = location(x) and radius



b.) Control Point Surface: Parameters = locations of control points



c.) Deformable Surface: Parameter = pressure (user defined and modifiable)

Figure 7.2) User leverage through parameterization of shape

The artful design of the energy functional resulted in the creation of surfaces which could be used interactively, opportunistically seek fair shapes and are well “behaved” during the interactive manipulation of shape. This was done by weighting the resistance to bending and the resistance to stretch in such a way that the energy minimization principle could be built in a quadratic form.

The energy principles selected for these geometric primitives are equivalent to a differential equations that map functions from the \mathcal{H}^0 Hilbert space to the \mathcal{H}^4 Hilbert space. If the

user is limited to applying piecewise continuous and point loads to surfaces spanning well behaved boundaries, the solutions for shape will be C^3 continuous. Corners in the boundary conditions are accommodated by introducing corners into the parametric domain of the surface. Surfaces built in this manner will be very smooth and fair.

Solving the deformable model equations using the Ritz finite element method supports continuous shape models suitable for aesthetic design and automatic manufacture of the resulting shapes. A review of the mechanics and the theory of the Ritz finite element method showed that only C^1 continuity be required of the functions used in the finite element solution. It was shown that the solution found with the finite element theory is the closest possible solution to the actual solution as measured by the energy of the error. Because of this property, the C^1 shape functions used to find solutions tend to C^3 shapes as the number of degrees of freedom of the model is increased. As such we extremely fair shapes were generated with only C^1 continuity enforced between shape patches. The conclusion is that shape fairness is a global property and not solely determined by the degree of smoothness enforced at the bounding edges between adjacent shape patches.

The theory of the finite element method and the needs of aesthetic design require the generation of shape functions which are C^1 continuous between finite elements. Several C^1 continuous shape functions are available in the literature and any of them could be used. The shape functions that were used in the development of the finite elements were selected based on their ability to directly enforce the limited set of geometric constraints needed for the ShapeWright design paradigm. As an aside we conclude that the many properties for which Bezier-Bernstein basis functions are usually selected are satisfied by the use of the energy minimization algorithm.

The range of topological structures that could be supported was greatly extended by developing triangular instead of rectangular surface finite elements. Triangular shape patches introduced no difficulties in the implementation and yet greatly enhanced the range of topological structures. Future CAGD systems should seriously consider the use of triangular patch elements over rectangular patches.

Using the finite element method to define shape results in finite element shape models. These finite element shape models can be directly exploited to automatically create application finite element models of the final shape. Thus this technique can be used to

automate the integration of the shape design process and the derivation of the finite element model for analytic purposes.

7.3 Future Directions

The deformable shape design algorithm can be thought of as a kind of expert system. It divides the tasks of free-form design between the user and the computer. The computer based algorithm automatically enforces a class of design rules while the user is freed to work on higher level design tasks. In this thesis the design rules built into the energy algorithm were to opportunistically seek fair shapes while exactly enforcing a set of design constraints. This acted to reduce the user load considerably by freeing the user from these tasks to concentrate on overall shape design. Future work can be applied to greatly extending the set of design rules that are built into the energy functional. The variety of this set directly defines the range of applications to which this technology can be applied.

What follows is a list of application areas that might be supported if very general design rules could be built into the deformable surfaces. These areas include generalizing the kinds of geometric constraints that can be supported, the automatic definition of complete geometry given partial geometric descriptions of shape, and the automated integration of part shape and part manufacturing process design.

One large class of design rules deals with setting, minimizing or maximizing the value of different geometric properties of the shape. Aircraft applications constantly seek to minimize weight while maximizing strength through variations in shape geometry. Industrial designers are constantly designing packaging of fixed volume and free-form shape. The work in this thesis needs to be extended to support in a uniform method the rich range of geometric constraints that can be defined as functionals of the entire surface. Lagrange constraint methods can be used to do this but some efforts will be needed to make such a system stable and interactive.

If such constraints can be supported by the deformable models they could be used to help automate the generation of part shapes in assemblies. In an assembly about 10% of the surfaces are actually used to define the function of the artifact. The rest of the surfaces are made in some sense to hold the functional surfaces in their proper place. An ideal assembly level modeler would allow a user to define the minimum number of surfaces to define the

function of the assembly. Then the system using deformable surfaces would automatically fill in the partial geometry of the parts subject to non-interference, minimum strength, and maximum weight criterions.

Deformable surfaces can also be used in such an assembly functional modeler to help abstract shape. Imagine shrink wrapping an engine of an automobile with a deformable surface. The resulting shape would capture the essence of the size and shape of the engine without needing to know of the hidden complexities. This abstracted engine could then be used in a higher level assembly to guarantee non-interference between the engine and other parts of the car such as the hood. Resolving interference conflicts could be automated with deformable surfaces. The hood, modeled in a deformable manner, changes its shape to accommodate a taller engine. If the hood model were supporting constraints that capture the aerodynamics of the problem, the hood could be made to deform automatically while preserving the fuel economy of the car. Once again we are back at the need to model more sophisticated constraints in the energy functional.

Another promising area of future application is to integrate the design of shape and process for free-form shapes. Instead of using an artificial energy functional designed for fairness and its ability to mimic familiar deformable surface behaviors, use an energy functional that accurately models the materials and processes to be used to build the part. That way the forces used to sculpt the surface could be used to automatically define the manufacturing of the surface. Additionally, the model could be sophisticated enough to detect when the material was being stressed beyond its failure limits and prevent the user from designing a free-form surface that can't be built much like the current deformable models help the user to build only fair shapes.

Accurate process models for such processes as stamping and casting are a current research area. Instead of waiting for these models to be developed it might be possible to build an approximate energy functional which could guarantee manufacturability of the surface without being able to define the actual behavior during a failure mode.

Many new manufacturing processes are being developed that will greatly increase the range of shapes that can be easily built. Laser lithography and 3-D printing of ceramic materials can produce very free-form shape prototypes automatically. Standard multi-axis machine technology is widely available to industry. The current shape design practices are restricted by the range and difficulty in developing shape inherent in the current shape modeling

technologies and the need to guarantee part manufacturability and functionality. The design and manufacture of shape is rapidly becoming limited by software practices. The deformable modeling technology as embodied in the ShapeWright modeling paradigm represents a path to extending the range of shape modeling while providing a means to integrate the needs of manufacturing, analysis and design.

References

The references for this work have been listed in seven different subject categories. These categories are;

CAGD: A very broad collection of topics that all have to do with Computer Aided Geometric Design relevant to the geometry presented in this thesis.

smooth = Shape Preservation, Smoothing and Fairing: Works concentrating on the interpolation of data with continuous curves, and on the incremental improvement of shape for aesthetic reasons.

triangle = Triangular and N-sided patches: Publications concentrating on extending the range of topological structures that can be modeled with a special emphasis on non-four sided patches.

ECAGD: Previous work that has used energy considerations or optimization algorithms to control shape geometry for a variety of purposes.

defmodel = Deformable Models: The Previous work most directly related to the subject matter of this thesis; the use of deformable models for the generation of shapes.

physmodel = Physical Modeling: A short selection of references describing relevant models of physical systems including elasticity, plate, and shell theory.

math = Mathematical References: A few of my favorite math books in the area of this thesis.

- | | |
|----------------------------|--|
| CAGD | "Computer Aided Design Report", Computer Aided Design, Vol. 10, No. 5, May, 1990 |
| [Andersson_87]
ECAGD | Andersson, E. and Andersson, R. and Boman, M. and Elmroth, T. and Dahlberg, B. and Johansson B., "Automatic construction of surfaces with prescribed shape", Computer-Aided Design, vol. 20, no. 6, July/August 1987 |
| [Baker_88]
CAGD | Baker, H. Harlyn, "Building, Visualizing, and Computing on Surfaces of Evolution", IEEE Computer Graphics and applications, vol. 8, no. 4, July, 1988 |
| [Barnhill_74a]
CAGD | Barnhill, R.E. and Risenfeld, R.F., Computer Aided Geometric Design, Academic Press, 1974 |
| [Barnhill_74b]
triangle | Barnhill, R.E.(triangle), "Smooth interpolation over triangles", In: Barnhill RE and Risenfelds RF(eds) Computer Aided Geometric Design, Academic Press, New York, London, 1974 |

- [Barnhill_83]
CAGD Barnhill, R.E. and Bohm, W. eds.(CAGD), Surfaces in CAGD, North-Holland Amsterdam., 1983
- [Barnhill_84]
smooth Barnhill R.E. and Whelan T., "A geometric interpretation of convexity conditions for surfaces", Computer Aided Geometric Design 1, pp. 285-287 North-Holland, 1984
- [Barnhill_87]
CAGD Barnhill, R.E. and Farin, G. and Fayard, L. and Hagen, H., "Twists, curvatures and surface interrogation", Computer-Aided Design, volume 20, number 6, July/August 1987
- [Barr_84]
CAGD Barr, A.H., "Global and Local Deformations of Solid Primitives", Computer Graphics (Proc. SIGGRAPH 84) July, pp 21-31, July 1984
- [Barsky_80]
smooth Barsky, B.A. and Greenberg, D.P., "Determining a Set of B-Spline Control Vertices to Generate an Interpolating Surface", Computer Graphics and Image Processing, Vol. 14, No. 3, pp. 203-226, Nov., 1980
- [Barsky_82a]
CAGD Barsky, B. and Greenberg, D., "Interactive surface representation system using a B-spline formulation with interpolation capability", Computer-Aided Design, Vol. 14, No. 4, pp. 187-194, July 1982
- [Barsky_82b]
CAGD Barsky, B.A., "End Conditions and boundary conditions for uniform B-spline curve and surface representations", Computers in Industry, Vol. 3, pp. 17-29, 1982
- [Barsky_83]
smooth Barsky, B.A. and Beatty, J.C., "Local Control of Bias and Tension in Beta-splines", ACM Transactions on Graphics 2, pp. 109-134, April 1983
- [Barsky_89]
smooth Barsky, Brian A. and DeRose, Tony D., "Three Characterizations of Geometric Continuity of Parametric Curves - Part I.", IEEE Computer Graphics and Applications, vol. 9, no. 2, November, 1989
- [Baumgarte_72]
math Baumgarte, J., "Stabilization of Constraints and Integrals of Motion in Dynamical Systems", Computer Methods in Applied Mechanics and Engineering, 1972
- [Bazeley_65]
triangle Bazeley, G.P. and Cheung, Y.K. and Irons, B.M. and Zienkiewicz, O.C., "Triangular elements in plate bending - conforming and non-conforming solutions", Proc. Conf. Matrix Methods in Struct. Mech., Air Force Inst. of Tech., Wright Patterson A.F. Base, Ohio, October, 1965
- [Beck_86]
smooth Beck, J.M. and Farouki, R.T. and Hinds, J.K., "Surface Analysis Methods," IEEE Computer Graphics and Applications, Vol. 6, No. 12, pp. 18-36, Dec. 1986

- [Bedi_89]
smooth Bedi, S. and Vickers, G.W., "Surface lofting and smoothing with skeletal-lines", Computer Aided Geometric Design 6, pp. 87-96, North-Holland, 1989
- [Beeker_86]
smooth Beeker, Etienne(smooth), "Smoothing of shapes designed with free-form surfaces", Computer-Aided Design, vol. 18, No. 4, May, 1986
- [Bell_69]
triangle Bell, Kolbein, "A refined triangular plate bending finite element", International Journal for Numerical Methods in Engineering, Vol. 1, 1969
- [Bezier_74]
CAGD Bezier, P., "Mathematical and practical possibilities of UNISURF", in Barnhill, R.E. and Riesenfeld R.F. (eds) Computer Aided Geometric Design Academic Press, pp. 259-302, 1974
- [Bezier_86]
CAGD Bezier, P.E., The Mathematical Basis of the UNISURF CAD system, Butterworth, London UK, 1986
- [Bloor_89a]
defmodel Bloor, M.I.G. and Wilson, M.J., "Generating n-Sided Patches with Partial Differential Equations", New Advances in Computer Graphics, Proceedings of CG International '89, 1989
- [Bloor_89b]
defmodel Bloor, M.I.G. and Wilson, M.J., "Blend Design as a Boundary-Value Problem", Theory and Practice of Geometric Modeling, Wolfgang Staber and Hans-Peter Seidel (Eds.), 1989
- [Bloor_89c]
defmodel Bloor, M.I.G. and Wilson, M.J., "Representing PDE surfaces in terms of B-Splines", Department of Applied Mathematics, The University of Leeds, Leeds LS2 9JT, 1989
- [Bloor_89d]
defmodel Bloor, M.I.G. and Wilson, M.J., "Generating blend surfaces using partial differential equations", Computer-Aided Design, vol. 21, no. 3, April, 1989
- [Bloor_89e]
defmodel Bloor, M.I.G., and Wilson, M.J., "Representing PDE surfaces in terms of B-Splines", Department of Applied Mathematics, The University of Leeds, Leeds LS2 9JT, 1989
- [Bloor_90]
defmodel Bloor, M.I.G., and Wilson, M.J., "Using Partial Differential Equations to Generate Free-Form Surfaces", Computer-Aided Design, March, 1990
- [Boehm_85]
triangle Boehm, W., "Triangular spline algorithms", Computer Aided Geometric Design 2, pp. 61-67, North-Holland, 1985
- [Boehm_87]
smooth Boehm, W., "Smooth Curves and Surfaces", in: Geometric Modeling: Algorithms and New Trends, Edited by G. Farin, SIAM, Philadelphia, PA, 1987
- [Boehm_88a]
smooth Boehm, W., "Visual Continuity", Computer-Aided Design, Vol. 20., No. 5, pp 307-311, 1988

- [Boehm_88b] Boehm, W., "On the Definition of Geometric Continuity",
smooth Computer Aided Design, Vol 20, No 7, pp 370-372, 1988
- [Bohm_84] Bohm, W. and Farin, G. and Kahmann, J., "A survey of curve and
CAGD surface methods in CAGD", Computer-Aided Design, R.E. Barnhill
and W. Boehm (eds), volume 1, number 1, July, 1984
- [Bonfiglioli_86] Bonfiglioli, Luisa, "An algorithm for silhouette of curved surfaces
smooth based on graphical relations", Computer-Aided Design, Vol 18, No.
2, March, 1986
- [Brady_84] Brady, Michael, and Yuille, Alan, "Representing Three Dimensional
CAGD Shape", Romansy Conference, AI Lab, MIT, Naval Research
Contract N00014-75-c-0643, 1984
- [Carlbom_88] Carlbom, Ingrid, "Modeling and Display of Empirical Data", IEEE
smooth Computer Graphics and applications, vol. 8, no. 4, July, 1988
- [Casale_87] Casale, Malcolm, "Free-Form Solid Modeling with Trimmed
CAGD Surface Patches", IEEE Computer Graphics and applications, vol.
7, No. 1, January, 1987
- [Catmull_78] Catmull, E. and Clark, J., "Recursively Generated B-spline
triangle Surfaces on Arbitrary Topological Meshes", Computer-Aided
Design, pp. 350-355, Nov. 1978
- [Celniker_89] Celniker G., Gossard D., "Energy-Based Models for Free-Form
defmodel Surface Shape Design", ASME Design Automation Conference,
Montreal Canada, Sep. 1989
- [Chang_84] Chang, G. and Davis, P.J., "An improved condition for the
smooth convexity of Bernstein-Bezier surfaces over triangles", Computer
Aided Geometric Design, Vol 1, 1984
- [Charrot_84] Charrot, P. and Gregory, J.A., "A pentagonal surface patch for
triangle computer aided geometric design", Computer-Aided Design, volume
1, number 1, July, 1984
- [Chiyokura_87] Chiyokura, Hiroaki, "An Extended Rounding Operation for
CAGD Modeling Solids with Free-Form Surfaces", IEEE Computer
Graphics and applications, vol. 7, No. 12, December, 1987
- [Clark_76] Clark, J.H., "Designing surfaces in 3D", Communications of the
CAGD ACM, vol. 19 No. 8, pp. 454-460, August 1976
- [Clough_65] Clough, Ray W., and Topcher, James L., "Finite element stiffness
physmodel matrices for analysis of plate bending", Proc. Conf. Matrix Methods
in Struct. Mech., Air Force Inst. of Tech., Wright Patterson A.F.
Base, Ohio, October, 1965
- [Coates_79] Coates, D., "Understanding Aesthetics: From Old Shoes to a
smooth Teacup", Industrial Design, pp. 32-35, Sept./Oct. 1979

- [Cohen_80]
CAGD Cohen, E. and Lyche, T. and Riesenfeld R., "Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics", Computer Graphics and Image Processing, Vol. 14, No. 2, pp. 87-111, Oct., 1980
- [Coons_67]
CAGD Coons, S.A., "Surfaces for the computer-aided design of space forms", M.I.T. MAC-TR-41, available from NTIS U.S. Department of Commerce, Springfield, VA 22151, 1967
- [Coquillart_87]
CAGD Coquillart, Sabine, "A control-Point-Based Sweeping Technique", IEEE Computer Graphics and applications, vol. 7, No. 11, November, 1987
- [Crow_87]
CAGD Crow, Frank (from conversations with Martin Newell and Jim Blinn), "The Origins of the Teapot", IEEE transactions of Computer Graphics and applications, January, 1987
- [de Boor_78]
CAGD de Boor, C., A Practical Guide to Splines, (Applied mathematical Science Series No 27) Springer, Berlin-Heidelberg-New York, 1978
- [DeRose_85a]
smooth DeRose, Tony D., Geometric Continuity: a Parameterization Independent Measure of Continuity for Computer Aided Geometric Design, Ph.D. Thesis, University of California, Berkely, California (August, 1985) Available as tech. Report No. UCB/CSD 86/255, Computer Sciences Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1985
- [DeRose_85b]
smooth DeRose, T.D. and Barsky, B.A., "An intuitive approach to geometric continuity for parametric curves and surfaces", Computer Generated Images - The State of the Art, edited by Magnenat - Thalmann, Nadia and Thalmann, Daniel, Springer-Verlag, Heidelberg, pp. 159-175, 1985
- [DeRose_88]
triangle DeRose, Tony D., "Composing Bezier Simplexes", ACM Transactions on Graphics, vol. 7, no. 3, pp. 198-221, July, 1988
- [DeRose_90a]
smooth DeRose, Tony D., "Necessary and Sufficient Conditions for Tangent Plane Continuity of Bezier Surfaces", To appear in Computer Aided Geometric Design, 1990
- [Derose_90b]
CAGD Derose, Tony D., "Interactive Surface Modeling in the Three-dimensional Design Space", Proceedings of NSF Design and Manufacturing Systems Conference, January, 1990
- [Dill_81]
smooth Dill, J.C., "An Application of Color Graphics to the Display of Surface Curvature", Computer Graphics (Proc. Siggraph), Vol 15, No. 3, pp. 153-161, Aug. 1981
- [Douglas_86]
CAGD Douglas, R.B., "The application of CAD/CAM techniques at Harland and Wolff", Computer-Aided Design, vol 18, no. 5, June, 1986

- [Dym_74]
physmodel Dym, Clive, Introduction to the Theory of Shells, Pergamon Press, Oxford, 1974
- [Farin_82a]
smooth Farin, G., "A construction for visual C1 continuity of polynomial surface patches", Computer Graphics and Image Processing, pp. 272-282, 1982
- [Farin_82b]
triangle Farin, G., "Describing a C1 surface consisting of triangular cubic patches", Computer-Aided Design, Vol. 14, pp. 253-256, 1982
- [Farin_86a]
triangle Farin, G., "Triangular Bernstein-Bezier patches", Computer Aided Geometric Design, pp. 83-127, North-Holland, 1986
- [Farin_86b]
CAGD Farin, G. and Barry, P.J., "Link between Bezier and Lagrange curve and surface schemes", Computer-Aided Design, Vol 18, No. 10, December, 1986
- [Farin_88a]
smooth Farin, Gerald and Sapidis, Nickolas, "Shape Representation of Sculpted Objects: the Fairness of Curves and Surfaces", Proceeding of Sea Grant Conference, MIT, October, 1988
- [Farin_88b]
CAGD Farin, Gerald, Curves and Surfaces for Computer Aided Geometric Design, Academic Press Inc., Harcourt Brace Jovanovich Publishers, Boston, 1988
- [Farin_89]
smooth Farin, Gerald and Sapidis, Nickolas, "Curvature and the Fairness of Curves and Surfaces", IEEE Computer Graphics and Applications, March, 1989
- [Farouki_88]
CAGD Farouki, R.T., and Rajan, V.T., "On the numerical condition of algebraic curves and surfaces 1. Implicit equations", Computer Aided Geometric Design, 1988
- [Ferguson_86]
ECAGD Ferguson, D.R., "Construction of Curves and Surfaces using Numerical Optimization Techniques", Computer-Aided Design, Vol. 18 No.1, pp. 15-21, January/February 1986
- [Ferguson_88]
ECAGD Ferguson, D. and Frank, P., and Jones A., "Surface shape control using constrained optimization on the B-spline representation", Computer Aided Geometric Design, Vol. 5, pp. 87-104, North-Holland, 1988
- [Ferguson_64]
smooth Ferguson, J., "Multivariable curve interpolation", Journal of the ACM II/2, pp. 221-228 (also Report No D2-22504 Boeing, Seattle, WA, USA 1963), 1964
- [Filip_89]
CAGD Filip, Daniel J., and Ball, Thomas W., "Procedurally Representing Lofted Surfaces", IEEE Computer Graphics and Applications, November, 1989
- [Fischer_87]
ECAGD Fischer, B. and Opfer, G., "Shape-preserving interpolation with optimal non-negative cubic splines", to appear in Numer. Math., 1987

- [Fletcher_87]
ECAGD Fletcher, G. Yates and McAllister, David F., "An Analysis of Tension Methods for Convexity-Preserving Interpolation", IEEE Computer Graphics and applications, vol. 7, No. 8, August, 1987
- [Fog_84]
CAGD Fog, N.C., "Creative definition and fairing of ship hulls using a B-spline surface", Computer-Aided Design, vol. 16, No. 4, July, 1984
- [Forsey_88]
CAGD Forsey, David R. and Bartels, Richard H., "Hierarchical B-Spline Refinement", ACM Computer Graphics, Vol. 22, No. 4, August, 1988
- [Fuchs_77]
CAGD Fuchs, H. and Kedem, Z.M. and Uselton, S.P., "Optimal Surface Reconstruction from Planar Contours," Communications of the ACM, Vol. 20, No. 10, pp. 693-702, Oct. 1977
- [Gordon_71]
CAGD Gordon, W.J., "Blending-function methods of bivariate and multivariate interpolation and approximation", SIAM Journal Numer. Anal. 8, pp. 158-177, 1971
- [Gordon_72]
CAGD Gordon, William J. and Wixom, James A., "Psuedo-Harmonic Interpolation on Convex Domains", Research Laboratories, General Motors Corporation, Warren Michigan, GMR-1248, August 1972
- [Gordon_74]
CAGD Gordon, W.J. and Risenfeld, R.F., "B-spline curves and surface", in Barnhill, R.E. and Riesenfeld R.F, (eds) Computer Aided Geometric Design Academic Press, pp. 95-126, 1974
- [Gregory_74]
smooth Gregory, J.A., "Smooth interpolation without twist constraints", in Barnhill, RE and Riesenfeld (eds) Computer Aided Geometric Design, Academic Press, New York, NY, USA pp 71-87, 1974
- [Gregory_80]
triangle Gregory, J.A. and Charrot, P., "A C1 triangular interpolation patch for computer-aided geometric design", Computer Graphics and Image Processing 13, pp. 80-87, 1980
- [Gregory_83]
triangle Gregory, J.A., "C1 rectangular and non-rectangular surface patches", In: Barnhill RE and Boehm W Surfaces in Computer Aided Geometric Design, North-Holland, Amsterdam, pp 25-33, 1983
- [Gregory_85]
triangle Gregory, J.A., "Interpolation to boundary data on the simplex", Computer Aided Geometric Design 2, pp. 43-52, North-Holland, 1985
- [Gregory_86]
triangle Gregory, J.A.(triangle), "N-sided surface patches", In: The mathematics of Surfaces", edited by J.A. Gregory, 1986
- [Gregory_89]
triangle Gregory, John A. and Hahn, Jorg M., "A C2 polygonal surface patch", Computer Aided Geometric Design 6, pp. 69-75, North-Holland, 1989

- [Grieger_85]
defmodel Grieger, Ingolf, "Geometry Cells and surface definition by finite elements", Computer Aided Geometric Design 2, pp. 213-22, North-Holland, 1985
- [Hagen_85]
CAGD Hagen, H., "Geometric Spline Curves", Computer Aided Geometric Design 2, pp.223-227, North-Holland, 1985
- [Hagen_86]
CAGD Hagen, H., "Bezier-curves with curvature and torsion continuity", Rocky Mountain Journal of Mathematics 16, pp. 629-638, 1986
- [Hagen_87]
ECAGD Hagen, H., and Schulze, G., "Automatic Smoothing with Geometric Surface Patches", Computer Aided Geometric Design, Vol. 4, pp. 231-236, 1987
- [Harada_84]
CAGD Harada, K. and Kaneda, K. and Nakamae, E., "A further investigation of segmented Bezier interpolants", Computer-Aided Design, Vol 16, No. 4, July, 1984
- [Herron_85]
triangle Herron, Gary, "Smooth closed surfaces with discrete triangular interpolants", Computer Aided Geometric Design 2, pp. 297-306, North-Holland, 1985
- [Harron_87]
smooth Harron, G., "Techniques for Visual Continuity", In: Geometric Modeling: Algorithms and New Trends, Edited by G. Farin, SIAM, Philadelphia, PA, 1987
- [Hess_87]
smooth Hess, W. and Schmidt, J.W., "Convexity preserving interpolation with exponential splines", to appear in Computing , 1987
- [Ho-Le_88]
triangle Ho-Le, K., "Finite element mesh generation methods: a review and classification", Computer-Aided Design, vol. 20, no. 1, January/February, 1988
- [Ho-Le_89]
triangle Ho-Le, K., "Deriving shape functions for triangular mid-edge finite elements through blending-function interpolation", Computer-Aided Design, vol. 21, no. 9, November, 1989
- [Hoschek_84]
smooth Hoschek, J., "Detecting regions with undesirable curvature", Computer Aided Geometric Design, Vol. 1 No. 2 pp 183-192, 1984
- [Hoschek_85]
smooth Hoschek, J., "Smoothing of curves and surface", Computer Aided Geometric Design, Vol. 2, pp. 97-105, 1985
- [Irons_69]
triangle Irons, Bruce M., "A conforming Quartic triangular element for plate bending", International Journal for Numerical Methods in Engineering, Vol. 1, 1969
- [Jerard_89]
smooth Jerard, Robert B. and Drysdale, Robert L. and Hauck, Kenneth, and Schaudt, Barry and Magewick, John, "Methods for Detecting Errors in Numerically Controlled Machining of Sculptured Surfaces", IEEE Computer Graphics and applications, vol. 9, No. 1, January, 1989

- [Jones_87]
ECAGD Jones, A.K., "Shape control of curves and surfaces through constrained optimization", In: Geometric Modeling: algorithms and new trends Farin, G E (ed) SIAM, Philadelphia, PA, USA, 1987
- [Jones_88]
triangle Jones, A.K., "Nonrectangular surface patches with curvature continuity", Computer-Aided Design, Vol. 20, No. 6, pp. 325-335, 1988
- [Kalay_83]
CAGD Kalay, Y.E., "Modeling polyhedral solids bounded by multi-curved parametric surfaces", Computer-Aided Design, volume 15, number 3, May, 1983
- [Kallay_90]
ECAGD Kallay, Michael and Ravani B. , "Optimal twist vectors as a tool for interpolation a Network of curves with a minimal energy surface", CAGD, 1990
- [Kass_88]
defmodel Kass, Michael and Witkin, Andrew, and Terzopoulos Demetri, "Snakes: Active Contour Models", International Journal of Computer Vision, 1988
- [Kaufmann_87]
smooth Kaufmann, E. and Klass, R., "Smoothing surfaces using reflection lines for families of splines", Computer-Aided Design, vol. 20, no. 6, July/August 1987
- [Kimura_84]
triangle Kimura, F. and Hosaka, M., "Non-four-sided patch expressions with control points", Computer-Aided Design, volume 1, number 1, July, 1984
- [Kjellander_83a]
ECAGD Kjellander, J.A., "Smoothing of bicubic parametric surfaces", Computer-Aided Design, Vol. 15, pp. 288-293, 1983
- [Kjellander_83b]
ECAGD Kjellander, J.A.P., "Smoothing of cubic parametric splines", Computer-Aided Design, vol 15, No. 3, May, 1983
- [Lanczos_49]
math Lanczos, Cornelius, The Variational Principles of Mechanics, University of Toronto Press, Toronto, 1949
- [Lang_68]
math Lang, Serge, Calculus of Several Variables, Addison-Wesley Publishing Co., Reading Massachusetts, 1968
- [Lee_89]
smooth Lee, Chih and Ravani B. and Yang, An Tzu, "A theory of Geometric Contact for Computer Aided Geometric Design of Parametric Curves", Submitted to the Journal of Computer Aided Geometric Design, Sept., 1989
- [Lee_90]
ECAGD Lee, E.T.Y., Energy, fairness and a counter example, Computer-Aided Design, Vol. 22, No. 1, pp. 37-40, Jan/Feb 1990
- [Lin_88]
CAGD Lin, C-H, "Generalized Bernstein-Bezier curves and surfaces", Computer-Aided Design, Vol. 20 No. 5, June, 1988

- [Loop_89]
triangle Loop, Charles T. and DeRose, Tony D., "A Multisided Generalization of Bezier Surfaces", to appear in ACM Transactions on Graphics, vol. 8, no. 3, July 1989
- [Loop_90]
triangle Loop, Charles T., "A G1 Triangular Spline Surface of Irregular Topology", in preparation, 1990
- [Lott_88]
ECAGD Lott, N.J. and Pullin, D.I., "Methods for fairing B-Spline surfaces", Computer-Aided Design, vol. 20, no. 10, December, 1988
- [MacCarthy_88]
CAGD MacCarthy, B.L., "Quintic splines for kinematic design", Computer-Aided Design, Vol 20, No. 7, September, 1988
- [Mansfield_64]
physmodel Mansfield, E. H., The Bending and Stretching of Plates, Pergamon Press, Macmillan Co., New York, 1964
- [Marguerre_69]
physmodel Marguerre, Karal and Woernle, Hans-Theo, Elastic Plates, Blaisdell Publishing Co., Waltham, Massachusetts, 1969
- [Mehlum_69]
ECAGD Mehlum, E., "Curve and surface fitting based on a variational criterion for smoothness", Central Institute of Industrial Research, Oslo, Norway, 1969
- [Mori_86]
smooth Mori, Kazuhiro and Mori, Hiromi and Tanaka, Toshio and Okuyama, Yoshihito, "Color Shading Technology for Design CAD Systems", SAE Technical Paper Series, 861175, September, 1986
- [Munchmeyer_87]
smooth Munchmeyer, F., "Shape interrogation: A case study", in: Geometric Modeling: Algorithms and New Trends, G. Farin, editor, SIAM, pp. 291-302, 1987
- [Nielson_74]
ECAGD Nielson, G.M., "Some piecewise polynomial alternatives to splines in tension", in Barnhill, RE and Riesenfeld, RF (eds) Computer Aided Geometric Design, Academic Press, 1974
- [Nielson_83]
triangle Nielson, G.M. and Franke, R., "Surface construction based upon triangulations", in Barnhill R.E. and Bohm W. eds SURFACES IN CAGD, North-Holland Amsterdam., 1983
- [Nowacki_83]
smooth Nowacki, H. and Reese, D., "Design and fairing of ship surfaces", in Barnhill R.E. and Boehm, W. (eds), Surfaces in CAGD, North-Holland, Amsterdam, pp 121-134, 1983
- [Pentland_86]
CAGD Pentland, Alex P., "Perceptual Organization and the Representation of Natural Form", Artificial Intelligence, 28, 1986
- [Pentland_87]
CAGD Pentland, Alex P., "Toward an Ideal 3-D CAD system", SPIE vol. 758, Image Understanding and the Man-Machine Interface, 1987
- [Pentland_88]
CAGD Pentland, Alex P. and Williams, John, "Virtual Construction", M.I.T. Media Lab Vision Sciences Technical Report, 105, July, 1988

- [Pentland_89]
ECAGD Pentland, Alex and Kuo, Jeff, "The Artist at the Interface", M.I.T. Media lab Vision Science Technical Report 114, January, 1989
- [Pentland_89]
ECAGD Pentland, Alex and Williams, John, "Good Vibrations: Modal Dynamics for Graphics and Animation", M.I.T. Media lab Vision Science Technical Report 115, January, 1989
- [Petersen_84]
smooth Petersen, Carl S., "Adaptive contouring of three-dimensional surfaces", Computer Aided Geometric Design, vol. 1, no. 1, pp 61-74, North-Holland, July, 1984
- [Piegl_84]
CAGD Piegl, L., "A Generalization of the Bernstein-Bezier method", Computer-Aided Design, Vol. 16 No. 4, pp. 209-215, July 1984
- [Piegl_85]
CAGD Piegl, L., "Recursive algorithms for the representation of parametric curves and surfaces", Computer-Aided Design, Vol 17, No. 5, June, 1985
- [Piegl_86]
CAGD Piegl, L., "A Geometric Investigation of the Rational Bezier Scheme of Computer Aided Design", Computers in Industry, pp. 401-410, Oct. 1986
- [Piegl_86]
CAGD Piegl, L., "Representation of Rational Bezier Curves and Surfaces by Recursive Algorithms", Computer-Aided Design, pp. 137-142, Sept. 1986
- [Piegl_87]
CAGD Piegl, L. and Tiller, W., "Curve and surface constructions using rational B-splines", Computer-Aided Design, Vol. 19 No. 9, pp 485-498, 1987
- [Piegl_87]
smooth Piegl, L., "Interactive Data Interpolation by Rational Bezier Curves", IEEE Computer Graphics and applications, Vol 7, No. 4, April, 1987
- [Platt_88]
defmodel Platt, John C. and Barr, Alan H., "Constraint Methods for Flexible Models", ACM Computer Graphics, vol. 22, no. 4, August 1988
- [Poeschl_84]
smooth Poeschl, T., "Detecting surface irregularities using isophotes", Computer Aided Geometric Design, Vol. 1 No. 2 pp 163-168, 1984
- [Powell_77]
triangle Powell, M.J. and Sabin, M.A., "Piecewise quadratic approximations on triangles", ACM transactions on Mathematical Software 3, pp. 316-325, 1977
- [Pramila_78]
defmodel Pramila, A., "Ship Hull Surface design using finite elements", Int. Shipbuild. Prog. Vol. 25 No. 284, pp. 97-107, 1978
- [Ravani_90]
smooth Ravani B. and Ku T.S., "Bertrand Offsets of Ruled and Developable Surfaces", Accepted CAD Journal, Oct. 1990, January, 1990

- [Rogers_83]
CAGD Rogers, David F. and Satterfield, G. and Rodriguez, Francisco A., "Ship Hulls, B-Spline Surfaces, and CAD/CAM, IEEE Computer Graphics and applications, vol. 3, No. 9, December, 1983
- [Rosenfeld_88]
CAGD Rosenfeld, Azriel, "Computer Vision: Basic Principles", Proceedings of the IEEE, vol 76, no 8, August 1988
- [Sapidis_90]
ECAGD Sapidis, N. and Farin, G., "Automatic fairing algorithm for B-spline curves", Computer-Aided Design, Vol. 22, No. 2 pp. 121-129, March 1990
- [Schwartz_88]
CAGD Schwartz, Eric L. and Merker, Bjorn and Wolfson, Estarose and Shaw, Alan, "Applications of Computer Graphics and Image Processing to 2D and 3D Modeling of the Functional Architecture of Visual Cortex", IEEE Computer Graphics and applications, vol. 8, no. 4, July, 1988
- [Schweikert_66]
ECAGD Schweikert, D.G., "An interpolation curve using a spline in tension", Journal of Math and Phys. No 45, pp. 312-317, 1966
- [Schwenkel_83]
defmodel Schwenkel, D., "Form studies relating to air-supported structures by minimization of element surface areas", Computer-Aided Design, vol. 15, No. 1, January, 1983
- [Sederberg_88]
CAGD Sederberg, Thomas W., and Meyers, Ray J., "Loop detection in surface patch intersections", Computer Aided Geometric Design, 1988
- [Selesnick_81]
CAGD Selesnick, S.A., "Local invariants and twist vectors in CAGD", Computer Graphs and Image Processing, Vol. 17, pp. 145-160, 1981
- [Shapiro_88]
CAGD Shapiro, Vadim and Voelcker, Herb, "On the Role of Geometry in Mechanical Design", Research in Engineering Design, Springer-Verlag New York, Inc., 1988
- [Stevens_81]
smooth Stevens, K.A., "The visual interpretation of surface contours", Artificial Intelligence, 17, 47-75, 1981
- [Storry_89]
triangle Storry, D.J.T. and Ball, A.A., "Design of an n-sided surface patch from Hermite boundary data", Computer Aided Geometric Design 6, pp. 111-120, North-Holland, 1989
- [Strang_73]
math Strang, Gilbert and Fix, George, An Analysis of the Finite Element Method, Prentice-Hall, Englewood Cliffs, N.J., 1973
- [Strang_86]
math Strang, Gilbert, Introduction to Applied Mathematics, Wellesley-Cambridge Press, Cambridge Massachusetts, 1986

- [Sutherland_63]
CAGD Sutherland, I., "Sketchpad: A Man-Machine Graphical Communications System", In: 1963 Spring Joint Computer Conference, reprinted in Interactive Computer Graphics, H. Freeman, ed. IEEE Comp. Soc., 1980, pp. 1-19, 1963
- [Terzopoulos_87a]
defmodel Terzopoulos, Demetri and Platt, John and Barr, Alan and Fleischer, Kurt, "Elastic Deformable Models", ACM, Computer Graphics, vol. 21, no. 4, July, 1987
- [Terzopoulos_87b]
defmodel Terzopoulos, Demetri and Witkin, Andrew and Kass, Michael, "Symmetry-Seeking Models and 3D Object Reconstruction", International Journal of Computer Vision, 1987
- [Terzopoulos_88a]
physmodel Terzopoulos, Demetri and Fleischer, Kurt, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", Proceedings of Siggraph 1988, Atlanta, August, 1988
- [Terzopoulos_88b]
defmodel Terzopoulos, Demetri and Witkin, Andrew, "Physically Based Models with Rigid and Deformable Components", IEEE Computer Graphics and applications, 1988
- [Terzopoulos_88c]
ECAGD Terzopoulos, Demetri, "The Computation of Visible-Surface Representations", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 4, July, 1988
- [Terzopoulos_88d]
defmodel Terzopoulos, Demetri and Witkin, Andrew and Kass, Michael, "Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion", Artificial Intelligence, 36, 1988
- [Tiller_83]
CAGD Tiller, Wayne, "Rational B-Splines for Curve and Surface Representation", IEEE Computer Graphics and applications, September, 1983
- [Timoshenko_34]
physmodel Timoshenko, S., Theory of Elasticity, McGraw-Hill Book Co., New York and London, 1934
- [van Wijk_86]
triangle van Wijk, J.J., "Bicubic Patches for Approximating Non-Rectangular Control-Point Meshes", Computer Aided Geometric Design, pp. 1-13, May 1986
- [Varady_82]
CAGD Varady, T., "An Experimental system for interactive Design and Manufacture of Sculptured Surfaces", Computers in Industry 3, pp. 125-135, 1982
- [Varady_87]
triangle Varady, T., "Survey and new results in n-sided patch generation", In: The Mathematics of Surfaces II, edited by R.R. Martin, 1987
- [Wever_88]
ECAGD Wever, U., "Non-negative exponential Splines", Computer-Aided Design, Vol 20, No. 1, January/February, 1988

- [Whelan_86]
triangle Whelan, T., "A representation of a C2 interpolant over triangles", Computer Aided Geometric Design 3, pp 53-66, North-Holland, 1986
- [Williams_86]
physmodel Williams, C.J.K., "Defining and Designing Curved Flexible Tensile Surface Structures", In: The Mathematic of Surfaces, edited by J.A. Gregory, 1986
- [Witkin_87]
defmodel Witkin, Andrew and Fleischer, Kurt and Barr, Alan, "Energy Constraints on Parameterized Models", Siggraph 87, Anaheim, July 27-31, and Computer Graphics, vol. 21, no. 4, July, 1987
- [Witkin_88]
defmodel Witkin, Andrew and Kass, Michael, "Spacetime Constraints", ACM, Computer Graphics, vol. 22, no. 4, August 1988
- [Woodward_88]
smooth Woodward, C.D., "Skinning techniques for interactive B-spline surface interpolation", Computer-Aided Design, vol. 20, no. 8, October, 1988
- [Woodwark_87]
CAGD Woodwark, J.R., "Blends in Geometric Modeling", In: Martin R.R., The mathematics of Surfaces II, Clarendon Press, Oxford, pp. 255-287, 1987
- [Wu_77]
CAGD Wu Sheng-Chuan, Abel J.F. and Greenberg, D.P., "An interactive computer graphics approach to surface representation", Communications of the ACM, Vol. 20 No. 10, pp. 703-712, October 1977
- [Zienkiewicz_67]
math Zienkiewicz, The Finite Element Method, third edition, McGraw-Hill Book Co., U.K., 1967

6080-15