Variations of Shape in Industrial Geometric Models

Variations of Shape in Industrial Geometric Models

by

Bastiaan Niels Veelo

A thesis submitted in partial fulfillment
of the requirements for the Norwegian degree of
Doktor Ingeniør
to
the Department of
Engineering Design and Materials
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway

November 2004

Supervised by:

Prof. dr.techn. Sven F. Fjeldaas

Admission committee:

Prof. Dr. Imre Horváth (first opponent)
Delft University of Technology, Delft — Netherlands

Dr. Ing. habil. Volker Bertram (second opponent)
Ecole Nationale Supérieure d'Ingénieurs, Brest — France

Prof. dr.ing. Terje Rølvåg (chair)
Norwegian University of Science and Technology, Trondheim — Norway

Electronically distributed at:
`http://urn.nb.no/URN:NBN:no-7263`
Online bibliographic description:
`http://publications.uu.se/ntnu/theses/abstract.xsql?dbid=240`

Doctoral thesis 2004:106
ISBN 82–471–6431–0 (printed)
ISBN 82–471–6430–2 (electronic)
ISSN 1503–8181
URN:NBN:no-7263

To Nicole.

Abstract

This thesis presents an approach to free-form surface manipulations, which conceptually improves an existing CAD system that constructs surfaces by smoothly interpolating a network of intersecting curves. There are no regularity requirements on the network, which already yields superior modelling capabilities compared to systems that are based on industry-standard NURBS surfaces.

Originally, the shape of such a surface can be modified only locally by manipulating a curve in the network. In this process there is an inherent danger that the curve is being pulled away from intersections that it has with other curves. When this happens, the network is invalidated as a surface representation, and many curves may have to be adjusted to restore network consistency and surface quality. This thesis contributes a method that solves these problems by propagating changes that are made in one curve to curves in its vicinity. How and to what extent curves react to changes is controlled by two parameters that can be varied along the curve that is being manipulated. Any curve may be constrained in one or more degrees of freedom. The integrity of the curve network is implicitly conserved, as well as the geometric continuity of the surface.

The result is a tool for the modification of curve-interpolating surfaces, which can easily be applied to large areas on models with any level of detail. This allows designers to concentrate on the creative process, rather than on planning chains of actions. They can explore different design variations, optimise shapes further, and generally be more productive.

Short Contents

Short Contents · iii

Contents · v

List of Figures · xi

Acknowledgements · xv

Glossary · xvii

Nomenclature · xxi

1 Introduction · 1

2 Where Standard Surface Methods Come Short · 11

3 Approximation of Arbitrarily Connected Points · 31

4 Interpolation of Arbitrarily Intersecting Curves · 49

5 Global Shape Manipulation · 71

6 Manipulation of Shapes in H-rep · 91

7 Evaluation · 127

8 Summary and Future Work · 157

Appendices · 163

A Computer Programming · 165

B Article in "Ship Technology Research" · 177

Bibliography · 189

Index · 201

Index of Citations · 206

Contents

Short Contents **iii**

Contents **v**

List of Figures **xi**

Acknowledgements **xv**

Glossary **xvii**

Nomenclature **xxi**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Focus 4
  - 1.2.1 Research Questions 4
- 1.3 Method 4
  - 1.3.1 Research Approach 6
  - 1.3.2 On the Relevance of Ship Design in a Broader Context 7
- 1.4 Contribution 9
- 1.5 Overview 9

2 Where Standard Surface Methods Come Short **11**
- 2.1 Standard Surface Methods 11
  - 2.1.1 Standards 11
  - 2.1.2 Commercial Software 14
- 2.2 Shortcomings 18
  - 2.2.1 Control versus Quality 18
    - 2.2.1.1 Control 19
    - 2.2.1.2 Quality 19

2.2.1.3 Conflict 20
2.2.2 The World is not Quadrilateral 21
2.2.2.1 When Control is Never Sufficient . . 22
2.2.2.2 Degenerate Surface Patches 24
2.2.2.3 Non-Regular Compositions 26
2.2.3 Change is Badly Supported 27
2.3 Chapter Summary and Look Ahead 28

3 Approximation of Arbitrarily Connected Points 31
3.1 Subdivision Surfaces 32
3.1.1 Catmull-Clark Subdivision Surfaces 33
3.1.2 Doo-Sabin Subdivision Surfaces 34
3.1.3 Interpolation 35
3.1.4 Value for Engineering Applications 36
3.2 Surface Splines 38
3.2.1 Tangent Plane Continuous Surface Splines . . 38
3.2.2 Curvature Continuous Surface Splines 45
3.3 Chapter Summary and Look Ahead 46

4 Interpolation of Arbitrarily Intersecting Curves 49
4.1 Interpolation of Cubic Bézier Curves 50
4.2 Interpolation of Continuous Curves 50
4.2.1 Method Outline 52
4.3 An Application to Conceptual Design 54
4.4 An Application to Ship Hull Design 57
4.4.1 Curves 58
4.4.2 Data Structure 59
4.4.3 Cross-Boundary Derivatives 59
4.4.4 Surfaces 59
4.4.5 Practical Experiences 60
4.5 Integrated Fairing: The H-rep Concept 61
4.5.1 Method Outline 62
4.5.2 Curves 63
4.5.3 Data Structure 66
4.5.4 Cross-Boundary Derivatives 66
4.5.5 Surfaces 66
4.5.6 Fairing 67
4.5.7 Continuity Considerations 68
4.6 Limitations 68
4.7 Chapter Summary and Look Ahead 69

5 Global Shape Manipulation **71**
5.1 Free-Form Deformation 72
5.1.1 Arbitrarily Shaped Lattices 72
5.1.2 Direct Manipulation 73
5.1.3 Impact on Model Representation 74
5.2 Hierarchical Refinement 75
5.2.1 Localised Hierarchy Surface Splines 76
5.3 Superposition 77
5.4 Decay Functions 78
5.5 Deformation Based on Physics 84
5.5.1 Finite Element Method 84
5.5.2 Low Degree of Freedom Models 86
5.5.3 Spring-Force Models 87
5.6 Chapter Summary and Look Ahead 88

6 Manipulation of Shapes in H-rep **91**
6.1 Problem Statement 92
6.1.1 Manipulation of Control Points 92
6.1.2 Manipulation of Data Points 92
6.2 Prospective Methods for Shape Variation 93
6.2.1 Free-Form Deformation 94
6.2.2 Hierarchical Refinement 94
6.2.3 Superposition 94
6.2.4 Decay Functions 94
6.2.5 Deformation Based on Physics 95
6.2.6 Making a Selection 96
6.3 Input to Decay Functions 96
6.4 Euclidean Distance Dependant Variation 97
6.4.1 Preliminary Definitions 98
6.4.2 Test Case 99
6.4.3 Test Evaluation 104
6.4.4 Constraints 104
6.4.5 Selection Fields 107
6.4.6 Variations in Non-Uniform Directions 108
6.4.7 Re-Design of Curves 111
6.4.7.1 Curve Sections as the Basis of Selections 111
6.4.7.2 Arc Length Parameterisation 115
6.4.7.3 The Combination, an Example 117
6.4.8 Unconnected Selections 117
6.4.9 Finishing Up 119
6.5 Surface-Distance Dependant Variation 120
6.5.1 Geodesics 121

6.5.2 Concerns for Surface Fairness 124
6.5.3 Computational Complexity 125
6.6 Dirichlet Parameterisation 126
6.7 Chapter Summary and Look Ahead 126

7 Evaluation **127**
7.1 Practical Value 127
7.1.1 Narrowing the Fore Ship 128
7.1.1.1 Comments 128
7.1.1.2 Alternatives 130
7.1.2 Construction of a Bulbous Bow 131
7.1.2.1 Alternatives 133
7.1.3 Change of the Aft Keel 134
7.1.3.1 Alternatives 135
7.1.4 Increase of Curvature in the Bilge 135
7.1.4.1 Alternatives 136
7.2 Limitations 137
7.2.1 Loss of Constraints 137
7.2.2 Introduction of Discontinuity 139
7.2.2.1 Alternatives 140
7.3 Relevance 141
7.4 Novelty 142
7.4.1 Decay Functions 142
7.4.2 Curve-Based Variations 143
7.4.3 Interaction between Vector Fields 143
7.4.4 De-Selection Fields 144
7.5 Recommendations 144
7.5.1 Data Management 144
7.5.1.1 Advantages 145
7.5.2 Elimination of Artifacts 146
7.5.3 Aspects of Efficiency 148
7.5.4 Open Interface 148
7.6 Two Paradigms Revisited 150
7.6.1 Surface Quality 151
7.6.2 Practical Usability 151
7.6.3 Method Complexity 153
7.6.4 Conclusion 154
7.7 Chapter Summary 155

8 Summary and Future Work **157**
8.1 Future Work 159
8.1.1 Constraints 159
8.1.2 Approximation Once Again 160

Appendices **163**

A Computer Programming **165**
A.1 Modern Software Engineering 165
A.1.1 The Object Model 166
A.1.2 Generic Programming 167
A.1.3 Application 168
A.1.4 Cost 168
A.2 Container Class 170
A.2.1 Class Interface 171
A.2.2 Class Implementation 173

B Article in "Ship Technology Research" **177**

Bibliography **189**

Index **201**

Index of Citations **206**

List of Figures

1.1 Diagram of the structure of this thesis. 6

2.1 Migration of standards towards STEP. 12
2.2 The STEP surface entity hierarchy. 13
2.3 A curve bounded (trimmed) quadrilateral surface can be generated to close an open shell. 18
2.4 The control net of a surface patch 21
2.5 Narrowing of the surface patch. 21
2.6 The control points do not line up 22
2.7 Modelling features individually. 23
2.8 A surface patch with a degenerate side. 25
2.9 A surface patch with a degenerate corner. 25
2.10 Filling n-sided holes with n quadrilateral patches. 26

3.1 The hierarchical levels in Forsey's dragon example. . . . 32
3.2 The Catmull-Clark algorithm. 34
3.3 The Doo-Sabin algorithm. 35
3.4 Subdivision surfaces in the entertainment industry. . . . 37
3.5 Gregory patch in Bézier form. 39
3.6 A 5-sided S-patch. 40
3.7 A surface spline constructed with S-patches. 41
3.8 The Bézier control points of surface splines. 42
3.9 Surface splines: an example. 44

4.1 An exploded view of how a B-rep can be used to support a wire-frame model 53
4.2 The influence of setting weights of NURBS curves. 64
(a) Parabolic arc. 64
(b) Elliptic arc. 64
(c) Hyperbolic arc. 64

5.1 Decay functions defined by Parent [1977]. 79
5.2 Comparison between a cosine and B-spline basis function. 80
5.3 Interference between fields of influence. 82
5.4 C^1 decay function proposed by Singh and Fiume [1998]. . 83

6.1 Schematic illustration of the computation of shift vectors. 98
6.2 A G^2 piecewise polynomial decay function. 99
6.3 The initial shape of the H-rep used in the test case. 100
6.4 Preview of the potential shift of data points for the test case. 101
6.5 The shape of the H-rep from the test case after the operation has completed. 102
6.6 Rendering of the shape variation from the test case. . . . 103
6.7 The result from the test case, intersected with a vertical longitudinal plane. 103
6.8 Symmetry plane constraint implemented by mirroring the selection field as well as the model. 105
6.9 A G^2 decay function with variable fall-off. 109
6.10 The effect of shifting data points in a uniform direction versus shifting data points in the normal direction. 110
(a) Variation with uniform direction. 110
(b) Variation along the local surface normal. 110
(c) At knuckles, the surface normal is undefined. . . 110
6.11 Variation by explicit definition of new control point positions. 111
6.12 Curve based variation. 112
6.13 Illustration of the effect that different curve parameterisations have on the typical shift vector. 112
6.14 Knot values and the support of basis functions drawn out. 114
6.15 The typical shift vector for an explicit variation, defined exclusively on the subsection of the curve that actually changes geometry. 114
6.16 Ordinary parameterisation versus arc length parameterisation. 116
(a) Ordinary parameterisation. 116
(b) Arc length parameterisation. 116
6.17 Arc lengths and unchanged curve sections. 118
6.18 Pseudo-code of an algorithm that only shifts contiguous sets of data points. 120
6.19 Geodesics 121
6.20 The effect of an imprecise shift vector. 125

7.1 Point based variation. 129

7.2 The construction of a bulbous bow. 132
(a) Initial shape. 132
(b) Stem curve re-designed. 132
(c) Middle water line drawn out. 132
(d) Lower water line adjusted. 132
(e) Extra curves remedy data scarcity. 133
7.3 The aft body, adjusted by a single curve-based variation. 134
7.4 Increase of Curvature in the Bilge. 136
7.5 Distortion caused by a change in tangent. 139
7.6 Disregarding limited curve sections can help eliminate artifacts. 147

Acknowledgements

First of all, I thank Professor Sven Fjeldaas for accepting my thesis proposal, which got it all started, and for the enjoyable collaboration that followed. Secondly, I thank the Faculty of Engineering Science and Technology in general and the Department of Engineering Design and Materials in particular (they were called differently when I started) for their financial support.

I am greatly indebted to Dr.ir.ing. Herbert Koelman from SARC BV, the Netherlands, for his interest and trust, suggestions and encouragement. The ability to work with his source code has given an enormous stimulation, a structure on which my ideas could find solid soil and grow to realisation. If it was not for the collaboration with him, I do not know where and when this study would have ended. SARC is also to be credited for the example ship hull model that can be seen at various places in this thesis.

In the later stages of my work, some of my relatives have been particularly supportive. Monika and Uwe, thank you for your care. But most importantly, my strongest supporters have been my wife and best friend Nicole, together with our boy Julian, and I am deeply thankful for your love, patience and understanding. This is to you.

Glossary

2D · two-dimensional.

3D · three-dimensional.

ϵG^2 · almost curvature continuous. The magnitude of dis-continuities in curvature are kept reasonably low due to discrete but closely spaced constraints. See also the definition of G^2 on page xix.

ANSI · American National Standard Institute.

B-rep · boundary representation, a popular scheme to represent solid models of physical objects [Zeid, 1991]. A B-rep defines the solid by means of its boundary, and on which side of this boundary the solid exists.

B-spline · basis spline, a popular approximation technique for curves and surfaces, based on the blending of control vertices. The polynomial degree and local support of the blending functions (basis functions) can be specified independently of the number of vertices. Bézier curves and surfaces are contained as special cases of B-spline curves and surfaces. See also the definition of NURBS on page xx.

C^0 · having parametric continuity of order 0, or based on piecewise polynomials that are not everywhere differentiable. This implies being G^0, as defined on page xix.

C^1 · first order parametrically continuous, or based on piecewise polynomials that are once differentiable everywhere. This implies being G^1, as defined on page xix. See also the definition of C^2 below.

C^2 · second order parametrically continuous, or based on piecewise polynomials that are twice differentiable everywhere. This implies being G^2, as defined on page xix. Parametric continuity is more strict than geometric continuity: a C^2 non-degenerate spline is always also G^2, but a G^2 spline may have a parametric continuity lower than C^2. Nevertheless, the appearance of one is not 'less smooth' than the other.

CACD · computer-aided conceptual design. Use of the computer in the conceptual design phase is still not very successful, because models must be generated with almost the speed of thought in order to support *idea generation*, which is essential in this phase.

CAD · computer-aided design. Honestly though, most design actions take place on paper with quick pencil strokes, and often it is more appropriate to talk of Computer Aided Draughting [Lawson, 1997, page 303]. When the computer helps with more than just draughting, we often speak of computer-aided engineering (CAE). The term of CACD (see the definition of CACD above) is sometimes used to emphasise *idea generation* in the design process.

CAE · computer-aided engineering, meaning that the computer helps predicting the performance of designs. See also the definition of CAD above.

CAGD · computer-aided geometric design, a term used in stead of CAD when an emphasise on geometric aspects is intended.

CAM · computer-aided manufacturing.

CAS · computer algebra system, a system for symbolic computation, such as Maple, Mathematica or GNU Maxima.

CDRS · Conceptual Design and Rendering System, a geometric modelling system developed by Evans & Sutherland Computer Corporation (E&S), and written entirely in Common Lisp.

CFD · computational fluid dynamics.

CSG · constructive solid geometry, a scheme to represent solid models of physical objects, based on boolean operations between primitives. See also B-rep.

D-NURBS · dynamic NURBS, an extension of the NURBS scheme (as defined on page xx) with physical properties for virtual sculpting and other applications.

E&S · Evans & Sutherland Computer Corporation, makers of the Conceptual Design and Rendering System (CDRS). Ivan Sutherland invented the first interactive graphics system, 'Sketchpad', in 1963 [Farin, 2002b].

Fairway · in the context of this thesis: a geometric modeller for the design of (the exterior of) ship hulls, developed by SARC, and written in Extended Pascal. In most other contexts: "The navigable part of a river, bay, etc., through which vessels enter or depart; the part of a harbour or channel which is kept open and un-obstructed for the passage of vessels" [Webster's Revised Unabridged Dictionary, 1913].

FEA · finite element analysis, using the finite element method (FEM).

FEM · finite element method, a method for solving an equation by approximating continuous quantities as a set of quantities at discrete points.

FFD · free-form deformation, a method for manipulation of the global shape of a geometric object, by warping the space in which it is defined.

FoB · flat of bottom, the area at the bottom of a ship hull that is completely planar, typically horizontal. Often present in cargo ships and larger passenger ships.

FoS · flat of side, the area on the side of a ship hull that is completely planar, typically vertical. Often present in cargo ships and larger passenger ships.

FSD · Fast Shape Designer, a geometric modeller for CACD (as defined on page xviii), developed at Delft University of Technology, and written in C++ with *Inventor*.

G^0 · positionally continuous, or having geometric continuity of order 0. Also notated as GC^0, and V^0 or VC^0 for visual continuity.

G^1 · tangentially continuous, or having first order geometric continuity, as well as positional continuity. Also notated as GC^1, and V^1 or VC^1 for visual continuity.

G^2 · curvature continuous, or having second order geometric continuity and all lower orders. Also denoted as GC^2, and V^2 or VC^2 for visual continuity. See also the definition of ϵG^2 on page xvii.

genus · a property of a surface defined as the largest number of non-intersecting simple closed curves that can be drawn on the surface without separating it. Roughly speaking, it is the number of holes in a surface; a sphere has a genus of 0, a torus has a genus of 1.

G^k · arbitrarily continuous, or having geometric continuity of order k and all lower orders. Also denoted as GC^k, and V^k or VC^k for visual continuity.

GP · generic programming, a programming concept that allows the definition of procedures, concepts, structures and algorithms independently from the data type on which they will work.

GUI · graphical user interface, the means of a computer program to communicate with the user through graphical elements such as windows, buttons, sliders, menus etc.

H-rep · hybrid model for ship hull representation, a geometric modelling technique integrating wire-frame, surface and solid representation, see Section 4.5. This should not be confused with other modelling techniques that can be called hybrid, e.g., because they support both the boundary representation (B-rep) and constructive solid geometry (CSG).

IGES · Initial Graphics Exchange Specification, an ANSI standard for the exchange of geometric data between computer programs.

ISO · International Organisation for Standardisation.

lattice · the arrangement of control points in a regular periodic pattern in three dimensions, on which a tri-variate spline volume is defined.

LeSS · localised hierarchy surface splines, surface splines that support manipulation at multiple levels of detail.

LGPL · Lesser General Public License, a license for "copyleft" (as opposed to "copyright") software libraries [Free Software Foundation, 2000].

MCAD · mechanical computer-aided design, CAD with a focus on the design of machines and appliances.

NTNU · Norwegian University of Science and Technology, or *Norges teknisk-naturvitenskapelige universitet.*

NURBS · non-uniform rational B-spline, an extension of the B-spline scheme (as defined on page xvii) that is capable of representing the curves of conic sections and surfaces of revolution, ellipsoids, etc. The term NURBS is rather unfortunate, because it suggests that uniform B-splines are explicitly excluded — which is not the case.

OOD · object-oriented design, the design of software systems in accordance with the object model, which is a paradigm in which problems are abstracted into objects and classes. The OOD is usually followed by implementation using object-oriented programming (OOP).

OOP · object-oriented programming, or programming in accordance with the object model, which is a paradigm in which problems are abstracted into objects and classes. For a successful application of OOP it is important to follow prior object-oriented design (OOD).

patch · a finite surface of n sides, where n is usually 4, described by a single mathematical relation.

PTC · Parametric Technology Corporation, the company behind the CAD package Pro|ENGINEER.

SARC · Naval Architectural Software and Engineering Centre, or *Scheepsbouwkundig Advies en RekenCentrum*, the company behind *Fairway*.

SGI · Silicon Graphics Incorporated, a graphics hardware and software company, having produced the OpenGL high performance graphics language, amongst other things.

STEP · Standard for the Exchange of Product Model Data, ISO 10303

STL · standard template library, a collection of generic algorithms and containers for use with the C++ programming language.

UML · unified modelling language, a graphical language for the notation of object-oriented designs.

valence · property of a node in a graph, equal to the number of edges joined to it. Also called valency or degree. In the context of surface patch assemblies: the number of shared patch sides emanating from a shared patch corner.

VDAFS · *Verband Deutscher Automobilhersteller FlächenSchnittstelle*, a Surface Data Interface format developed by the German Automobile Manufacturers Association. VDAFS is a German national standard published as DIN 66301.

Nomenclature

$\mathbf{U}$	knot vector of a B-spline curve, page 113.
$\mathbf{c}(t)$	a parametric curve, page 109.
$\hat{\mathbf{c}}(\hat{t})$	a parametric curve, variation on $\mathbf{c}(t)$, page 109.
$d_{i,j}$	shortest distance between data point i and the base of selection field j, page 98.
δ	normalised distance, $d_{i,j}/r_j$, see equation (6.2), page 98.
f_j	radial decay function for the intensity of selection field j, page 98.
g_k	radial function belonging to a de-selection field, which scales down the intensity of selection fields, page 105.
i	index of data points on the surface, page 98.
j	index of selection fields, page 98.
k	index of de-selection fields, page 105.
m	mapping from t to $\hat{t}$, page 115.
N	B-spline basis function, page 113.
$\mathbf{P}$	control point of a B-spline curve, also known as a vertex of its control polygon, page 110.
r_j	extent of selection field j, page 98.
$\mathbf{s}_i$	shift vector for data point i, see equation (6.1), page 98.
$\mathbf{S}_j$	typical shift vector, page 98.
t	curve parameter, page 109.

CHAPTER 1

Introduction

In computer-aided design (CAD), or more precisely, computer-aided geometric design (CAGD), there are two different paradigms for the description of surfaces in geometric models, each with an equally long history. One is based on the continuous approximation of discrete points, the other on transfinite interpolation of curves. For an introduction on these surface representations and their difference, the reader is referred to standard texts such as [Rogers and Adams, 1990; Foley *et al.*, 1990; Zeid, 1991; Piegl and Tiller, 1997; Farin, 2002b]. The industry standard for surface description uses so called non-uniform rational B-spline (NURBS) surface patches, which follow the first paradigm and form the basis of most established systems for computer aided design, manufacturing and engineering (CAD/CAM/CAE). This fact puts the second paradigm somewhat in the shade, and very few commercial systems are based on it.

1.1 Motivation

This study was initiated out of frustration, from an engineering and design point of view, with the shortcomings of existing NURBS-based systems. These shortcomings consist of two main problems. Firstly, there is a conflict between the description of detailed surface features and larger-scale surface fairness. Secondly, many engineering problems demand a patch layout or topology that differs from a regular checkerboard. That is, either the number of patches meeting at a patch corner is not necessarily four, or patches have not necessarily four sides, or both. Geometries with this property are usually denoted as having arbitrary topology*. Patches that are not quadrilateral are not covered by data exchange standards and therefore rarely supported by commercial CAD/CAM/CAE systems. In addition, the upholding of tangential and curvature continuity across patch boundaries is usually badly supported, especially around irregularities in the patch layout. Chapter 2 describes these shortcomings in more detail.

Consequently, I engaged in attempting to address these deficiencies. Unfortunately, I was rather ill informed about the state of the art, and I have used a considerable amount of time reinventing wheels and creating an infrastructure that I never used. Eventually, I discovered the appropriate channels of communication, and I learned that the problems had been identified and were being addressed. One particular idea that I had, which I thought could solve some of the detail versus fairness problems and which was the primary motivator for engaging in this study, appeared to have been published only three years earlier [Léon and Trompette, 1995, see Section 5.5.3 in this thesis]. Most contributions are coming from the fields of applied mathematics and computer science, and so the action is taking place in a different scientific arena than where I grew up in. I was foreign to the vocabulary that is being practised there, which explains to some extent why these advances remained outside my horizon initially.

I realised that I was lacking the appropriate background and that the state of the art had advanced beyond the point were I would be able to contribute significantly to the approximation paradigm, within the time remaining for the study. Chapter 3 gives a survey of some of the solutions that have been proposed for the definition of surfaces with arbitrary topology, within the approximation paradigm, and Chapter 5 surveys methods that address the detail versus fairness conflict by means of global manipulation.

The interpolation paradigm advances in a slower pace. I have practically been aware of the state of its art since I studied for my Bachelor degree, when I had the opportunity to visit a company called Naval Architectural Software and Engineering Centre (SARC) to attend a demonstration of a computer program that was being developed there, for the geometric design of ship hulls. Its modelling

*In this thesis, the term "topology" is used to abstract the inherent connectivity of objects while ignoring their detailed form. "Arbitrary topology" thus means that there are no rules on the connectivity; surface elements may connect arbitrarily (as long as they describe a 2-manifold). There is also a formal definition of topology defined in terms of set operations [Weisstein, 2004] where "arbitrary topology" could be interpreted as "no topology at all". That would be a misunderstanding.

methodology is based on transfinite surfaces that interpolate an arbitrary network of intersecting curves, and thus supports geometries with arbitrary topology. The method effectively restores the traditional way of lines plan draughting in a computer method. Although I was impressed with the achievement and could clearly see the advantages of computerisation, I also saw that the biggest limitation inherent to lines plan draughting was not addressed. In particular, it remained the designer's responsibility to make sure that curves that should intersect each other, also really do. This has the effect that the freedom to make changes in the design decreases as the number of curves in the model grows. Because the surface description is easily invalidated, I had dismissed the interpolation paradigm as inferior to the approximation paradigm at that time.

By chance, the interpolation paradigm was brought to my attention again when I stumbled over the PhD thesis of the man behind the above mentioned computer method, Herbert Koelman [1999]. His thesis describes the conception and implementation of said method. This time I was better informed, and I started to doubt the fundamental value of the approximating paradigm for engineering applications. In engineering, precision is often more important than aesthetics. Often a model has to adhere to a prescribed geometry within a tight tolerance. In this regard it is important to observe that a curve is much easier controlled than a surface. Also, many stylists are used to think in terms of so called feature curves. For both these reasons, it probably makes more sense to work with a surface that is defined by curves than the other way around. In addition, when the object being engineered is essentially a plate construction, like a ship, the information required for computer-aided manufacturing (CAM) consists of plate contours. When both the input and output consists of curves, there is reason to argue that the modelling methodology in between, be based on curves as well. This way, the designer gets direct control over the shape of the parts being manufactured. Chapter 4 gives a detailed covering of how the interpolation paradigm has evolved over the last decade, with respect to modelling shapes with arbitrary topology.

Still, the interpolation paradigm is lagging behind the approximation paradigm, as it has not received the same amount of attention to resolve remaining limitations; at least in theory that is, as very few approaches that are presented in chapters 3 and 5 have made it to mainstream implementations yet. For the interpolation paradigm, the main remaining limitation is the decrease in modelling freedom as the design progresses, as mentioned earlier. Section 4.6 discusses the details of this limitation.

1.2 Focus

The interpolation paradigm has good prospects from an engineering point of view, and contrary to the case of approximation I could still see opportunities to contribute. Therefore, I switched my focus of research from the approximation paradigm to the interpolation paradigm. The focus of this thesis is therefore computer-aided geometric design (CAGD) of smooth surfaces without regularity constraints, i.e. with arbitrary topology, in particular by means of interpolation of a network of arbitrarily intersecting curves.

1.2.1 Research Questions

With the above focus, this thesis will seek an answer to the following questions:

1. How can the designer be freed from the responsibility to maintain curve intersections everywhere?
2. How can the designer apply larger-area changes in the geometry without damaging detail or surface fairness?
3. How does the resulting method compare with methods that follow the approximation paradigm?

Questions one and two are answered in Chapter 6 and the third question is answered in Section 7.6.

1.3 Method

In discussing research method, it is important to differentiate between science on the one hand and technology or engineering on the other. Science is concerned with the discovery of truth, while technology is concerned with the application of scientific knowledge for the well-being of mankind. As Carl Mitcham puts it:

> "The questioning of distinctly technological ideas has a different content than the questioning of scientific ideas. The assumption among technologists is not that the technological ideas are true but that they work, and that the works to which they give rise are good or useful." [Mitcham, 1994]

In the search for truth, rigour is essential, and since long has the Positivist doctrine set rules for the demarcation of scientific knowledge from other forms of knowledge and beliefs. There is a desire to impose the same rigour on engineering, but the Positivist view on practice is not as successful as it is on science. "Increasingly we have become aware of the importance to actual practice of phenomena — complexity, uncertainty, instability, uniqueness, and value-conflict — which do not fit the model of Technical Rationality [i.e., the Positivist epistemology of practice]" [Schön, 1983, p. 39]. Apparently, the problem manifests itself not only in the application of scientific knowledge, but also in the making if it. "From [the Positivist] perspective, we tend to see science, after the fact, as a body of established propositions derived from research. [. . .] But we may also consider science before the fact as a process in which scientists grabble with uncertainties and display arts of inquiry akin to the uncertainties and arts of practice" [Schön, 1983, p. 48–49]. This view is shared by Latour [1987], who considers the situation of scientists and engineers while they are doing their job. He finds that their struggles carry great similarities, to the point that science and technology in action, before the fact, can be covered by the single term *technoscience*. Technoscience is compared with an open box of Pandora: there is uncertainty, there are deadlines, passions, decisions, traps and dead-ends. When the science is done and facts have been produced, the box closes and all this danger disappears inside. First then it becomes a black box, a fact of truth or a functional system, which can be accepted without questioning and with which further science can be build.

As a better epistemology of practice, and in search for rigour in practice, Schön [1983] proposes the concept of *reflection-in-action*, which has been summarised as follows:

> [Schön] describes professional work as a *reflective dialogue with the problem situation*, where new designs (solution hypotheses) are continually developed and tested. To attack wicked problems, *problem setting* (framing, analysis, understanding) and *problem solving* must be intertwined. Each new proposal may contribute to solving the current problems, but also to building an understanding of the problem, trigger new perspectives and ideas for solution. Social dialogue, rules of thumb, tacit and personal knowledge are paramount for testing and assessment. [. . .] The technological process is thus less explicit and formal than the scientific method [. . .], and its problem-hypothes[i]s-testing-evaluation cycles are more rapid. [Jørgensen, 2004, app. A.2.1]

1.3.1 Research Approach

As this is an engineering thesis, we will adopt Schön's model of reflection-in-action. After framing the problem, we will find that there is more than one way of solving it; after all, the product of engineering is not a truth value, but a relative answer of variable value. At the same time, resources permit only the exploration of one trail. This section will present a plan for research that is intended to select the right trail that will lead us to a solution of good value. The success of this selection will be evaluated at a conceptual level by comparing the value of the resulting solution with the expected value of other approaches.

The plan is to take three rounds of research, with three different objectives, depicted schematically in Figure 1.1 below. Because this study went through a paradigm shift with regard to surface representation, we will not loose the former out of sight completely.

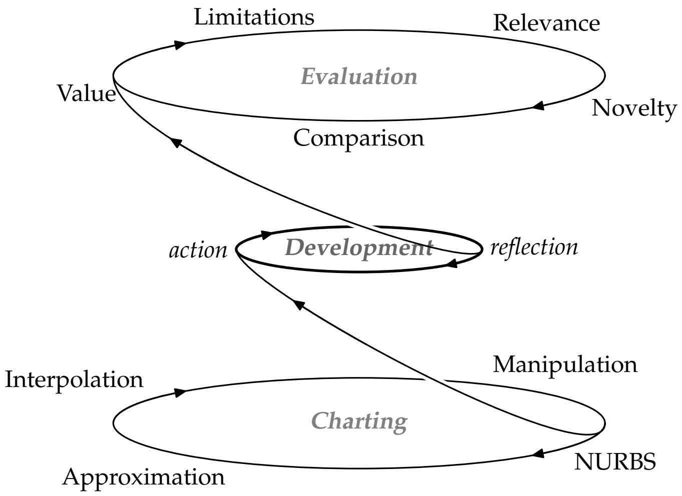


Figure 1.1: Diagram of the structure of this thesis.

Our first round will be of a charting character and have a wide diameter. Our point of departure is the discussion of the industry standard of surface representation, where we will familiarise ourselves with the problem. We will pass through both paradigms of alternative surface representations, making out the state of the art, and end in the coverage of various methods for shape manipulation. This gives us overview, understanding and context of the problem, and it gives us an idea of the existing solution space and sources of inspiration.

In our second round we will tighten our grip and move in smaller circles, zooming in on one surface representation of industrial relevance and one method for shape manipulation. These are combined and adapted to each other to develop an improved modelling methodology. Reflection-in-action [Schön, 1983] will be facilitated by means of computer implementation of experiments for validation and evaluation of hypotheses and ideas. This gives rise to new ideas and inspiration to fuel another round of development. Thanks to a collaboration with SARC, the programming could be done integral to the commercial computer code of Koelman [1999].

When the method has evolved to an effective and useful tool, we will will move on to a higher plane, widening our view. Here we will consider the practical value and limitations of our contribution, and its relevance and novelty. We will make conceptual comparisons with the expected performance of other approaches that were identified in the first round, and we will make an updated comparison between the approximation paradigm and interpolation paradigm in general.

1.3.2 On the Relevance of Ship Design in a Broader Context

The design of ship hulls is a good benchmark for geometric modelling frameworks, due to a number of special requirements:

- The modelling of ship hulls favours methods that allow modelling with arbitrary topology. Systems that use standard parametric surface patches typically exhibit problems in modelling the bow and stern regions.

- In applications of industrial design or the entertainment industry it is generally sufficient that the model "looks good". In ship hull design often much stricter requirements are in effect. An exact shape can be dictated by hydrostatical and hydrodynamical principles, by internal space requirements, or by practical reasons such as the construction of sponsons that must connect seamlessly to the existing hull of a ship that is being modified, to improve its stability.

- Cargo vessels typically have areas on their shell that are completely planar, the so-called flat of side (FoS) and flat of bottom (FoB). The modelling system must not allow deviations from these planes and at the same time provide a fluent transition to the curved parts of the hull.

- Naval architects often require explicit geometric *dis*continuities in the ship hull. Sharp edges (knuckle lines) are an obvious example where tangents are discontinuous. Discontinuities of curvature also exist, e.g., when the transition between the FoS and the FoB at the mid-ship section (the bilge) is designed to be circular. Obviously, a circular bilge has constant curvature, and the FoS and FoB have no curvature at all, so there is a curvature discontinuity where these meet[*]. Curvature discontinuities can also arise implicitly, for example when water lines are designed to be parabolic in the bow region[†]. An optimal design system should allow free orientation of these discontinuities, and allow them to vanish over distance.

- The design of ship hulls put high demands on surface fairness. Here a mathematical guarantee of geometric continuity of some degree is not sufficient. The process of fairing a hull is almost an artistic activity, in which one works out unwanted variations in surface curvature, and at the same time purposely accepts sudden changes or even discontinuities in curvature, as described above. This is a process that is difficult to automate, and a good hull design system should provide means to evaluate and improve surface fairness, and allow a good deal of human control in the process.

- In the maritime industry, there are strict and tight bounds on the availability of time and funding. Before a shipyard can persuade a customer to sign the contract for an order, a precise estimation of the cost and performance of the vessel needs to be made. Because of the complexity and speciality (individuality) of ships, this often involves doing much of the design work, including hull shape design [Abt *et al.*, 2001]. Due to the capital cost of the products, customers of course do price inquiries at more than one place, and shipyards are competing for the same orders world wide. This way, many a hull is designed in return

[*]A circular bilge is a tradition that stems from the time when lines plans were drafted by hand. But a bilge that is curvature continuous with the FoS and FoB can be easier to build, and in this age of CAM, there is no real reason why the bilge should be circular.

[†]Likewise, this is a design trick with a long history, but not one that necessarily produces a bow of great beauty.

for nothing, because the order went somewhere else. The contract itself often includes financial penalties for delayed delivery, which can be so severe that they have the potential to bankrupt the shipyard in case of violation (which is another reason to be precise in the preliminary design).

This stands in contrast with the automotive industry, where series are typically very large and there is more time to perfect the shape of visual surfaces. One can also afford to assign more people to this job. But most importantly, the relations of shape with other design variables are fewer and simpler; e.g., a change in weight or the length centre of gravity does not require changes in shape, unlike in ship building.

1.4 Contribution

The contribution of this thesis consists of a method to preserve curve intersections automatically during curve editing of a network of intersecting curves. When working with a surface that interpolates such a network, this gives the designer the freedom to make design changes independently of the number of curves and curve intersections in the model. With two extra parameters that can be varied along the curve that is being manipulated, the designer is able to control how and to what extent the surface that surrounds the curve follows the changes in the curve.

Larger area shape variations in a model can be accomplished by means of a simple spatial deformation, in which selected features of the model can be constrained in one or more coordinates. Smoothness conditions and shape details that exist in the model are preserved throughout the variation. The method is purely geometric and therefore interactive to a great extent.

1.5 Overview

This thesis has 8 chapters. This introduction has outlined the motivation, focus, method and contribution. Chapters 2–5 provide the background for the work and cover the lower cycle in Figure 1.1. Chapter 6 contains the contribution of this thesis and represents the middle cycle. The upper cycle, validation and evaluation, is contained in Chapter 7.

CHAPTER 2

Where Standard Surface Methods Come Short

In this chapter the reader is introduced to the fundamental problem that motivates this thesis. Occasionally, I will use terms from basic spline theory without prior introduction. Knowledge of their meaning will not be necessary to understand the essence in this chapter. The interested reader is referred to one of the standard texts on curve and surface theory [Rogers and Adams, 1990; Foley *et al.*, 1990; Zeid, 1991; Piegl and Tiller, 1997; Farin, 2002b].

2.1 Standard Surface Methods

Before discussing any shortcomings, we should decide on what we understand as standard surface methods. A good place to start is in the vendor neutral standards for data exchange between geometric modellers.

2.1.1 Standards

There is more than one standard for data exchange between geometric modellers. The Initial Graphics Exchange Specification (IGES) is an American national standard, and widely supported. The German automotive industry has produced the *Verband Deutscher Automobilhersteller FlächenSchnittstelle* (VDAFS), which is a national standard in

Germany. There are several others, and more recently the International Organisation for Standardisation (iso) reached an international standard known as the Standard for the Exchange of Product Model Data (step). As step is meant to cover and replace iges, vdafs and other standards (see Figure 2.1 below) it suffices for us to consider only step.

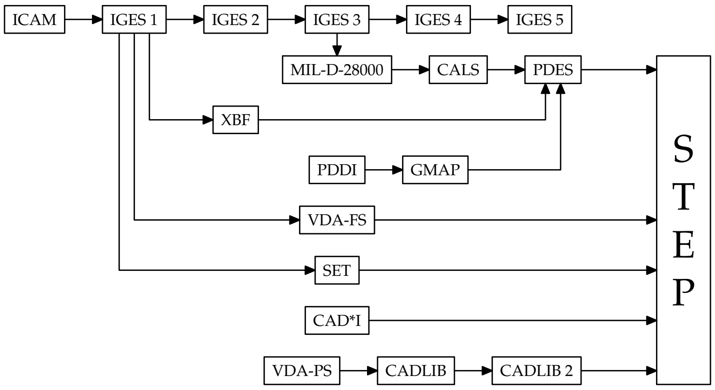


Figure 2.1: Migration of standards towards step. (Source: `www.prostep.org`.)

Figure 2.2 on the next page shows the surface types that are supported by step. An important entity is the bi-parametric basis spline (B-spline) surface and its rational variant. The standard specifies special cases of the B-spline surface explicitly by constraining the knot vector, which allows a more compact description of the model data. Also, implementations may be able to do optimisations based on the specialisation. However, all specialisations are covered by the general (rational) B-spline surface definition. Surfaces of this kind are often popularly denoted as nurbs surfaces[Piegl and Tiller, 1997].

In the standard, the B-spline surface is categorised as a bounded surface, meaning that it can describe only surfaces with a finite area. This as opposed to the elementary surfaces, which are in principle un-bounded. From the other examples of a bounded surface we see that a surface may be trimmed by a curve drawn on the surface, producing a curve bounded surface. This causes a partition of the surface to be hidden, but it is important to realise that the surface data remains in the model and participates in the definition of the visible part of the surface. Another way to trim bi-parametric surfaces is to regard only a sub-domain in either parameter direction, resulting in what the standard calls a rectangular trimmed surface. Surfaces may

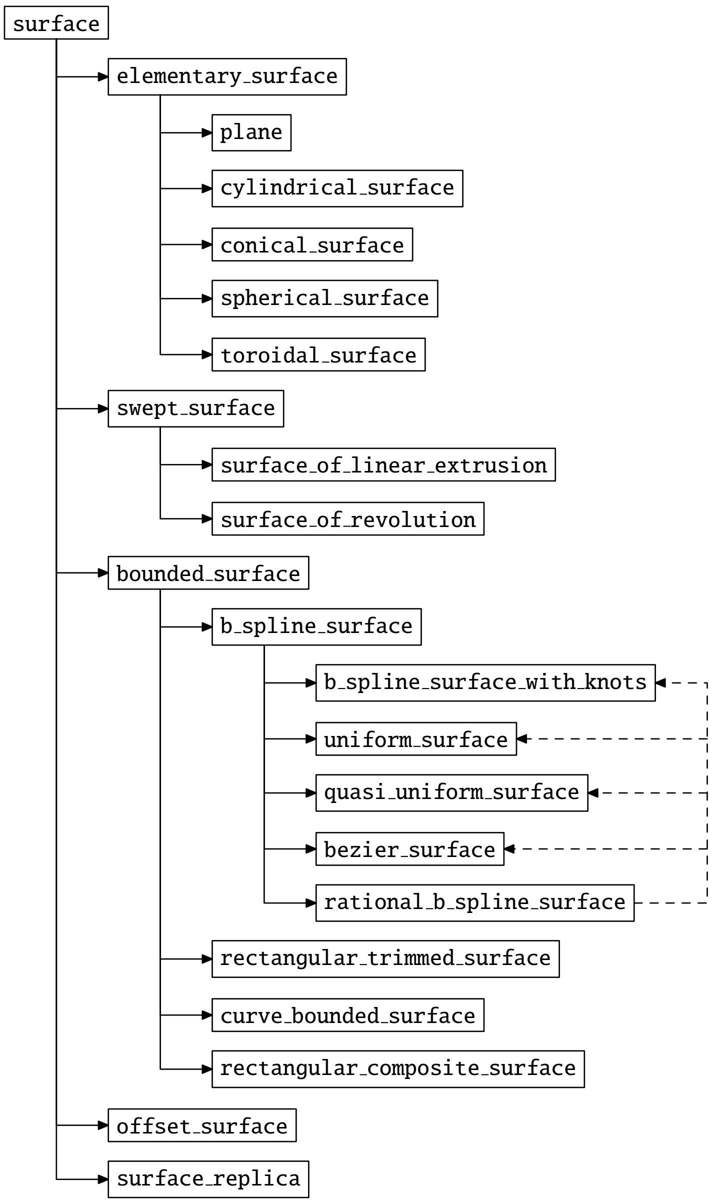


Figure 2.2: The STEP surface entity hierarchy, distilled from online application protocols (`www.steptools.com/support/stdev_docs/express/`). The instance `rational_b_spline_surface` and the dashed reversed arrows mean that the `b_spline_surface` and all its specialisations may also be rational.

also be combined into a complete checkerboard arrangement, where four surfaces meet at internal corners, to form a rectangular composite surface. The latter entity does not contain any actual geometry, it is just a list of references to surfaces and a documentation of the geometric continuity constraints that exist between them.

Summarising the parametric surface capabilities that are covered by the standard, we can say that the standard (only) defines bi-parametric surface patches [*], which are rectangular unless they are degenerate, and (only) regular combinations of them, producing rectangular composites. The term "rectangular", however common, is somewhat misplaced, as these surfaces have little if anything to do with right angles. The point is that they have four sides, which is accurately covered by the term "quadrilateral".

The offset surface and the surface replica entities only contain a reference to another surface. It is possible to represent all elementary surfaces in the standard with a NURBS definition if one allows patches to be degenerate, and because a NURBS curve can describe any supported sweep curve, NURBS surfaces can also represent the swept surfaces in the standard. We conclude that the quadrilateral NURBS surface definition covers all surface entities from the STEP standard, and thus can be regarded a standard surface method. Although triangular NURBS (or should we say "trilateral" NURBS) do exist [Qin and Terzopoulos, 1997], they are not in the standard.

But in order to get an overview of standard surface methods, it is not enough to look at data exchange standards alone. A good standard for data exchange needs to cover the common denominator of established commercial software. But a single implementation may brake loose from the pack and advance further. Therefore we will change perspective and have a quick look at commercial software in the following section, with respect to surface definitions.

2.1.2 Commercial Software

We will focus on the automotive industry because of the strict requirements on geometric continuity of body panels, which is due to their specular finish. In the automotive industry, it is tradition to differentiate between three classes of surfaces, as a specification of their required quality.

[*]The number of surface parameters can be concluded from a further inspection of the standard, omitted here. Triangular parametric surfaces usually are defined on barycentric coordinates, which consist of three coordinates, which are clearly not present in the standard.

Class A · All visible surfaces, be it interior or exterior. No aesthetic defects are tolerated.

Class B · Secondary surfaces that can be seen sometimes, but are less important. Examples are the inside of the glove compartment and the door aperture.

Class C · Surfaces that are not visible during normal use, like the seat mounting under the carpet or the shell of the fuel tank.

Beware that this is a classification of surface requirements, not of surface properties. Much less should the classes be understood as surface definitions.

Most vendors have a history in mechanical computer-aided design (MCAD), and their products reflect that. *Parametric Technology Corporation (PTC)* has Pro|ENGINEER. *Dassault Systèmes* has Catia and SolidWorks. *UGS PLM Solutions* has SolidEdge, I-DEAS and Unigraphics, the latter two of which are in the process of being merged into one product called NX. When looking at the surface technology that they are based on, clearly "NURBS" is the recurring buzz-word. It is hard to back this statement up with facts, but internally, probably all of them use the quadrilateral NURBS surface definition exclusively*.

It is not surprising that NURBS are thriving in MCAD, as most shapes in mechanical engineering are primitive: planar, cylindrical, conical, spherical, etc. They result from the common solid modelling operations such as extrude, sweep, revolve and fillet, without the need for surface manipulation. The NURBS representation supports all these shapes, which is an advantage for the programmer as algorithms are required for one single surface representation only.

Designing free-form shapes†, as are demanded in industrial design, has traditionally been more cumbersome in these systems. The trend is now towards curve based surface editing, which seems to be an improvement over low level control net editing. It is being sold as a new technology, with names as "BlueSurf" in the case of SolidEdge, but in fact it is just a new interface on the standard quadrilateral patches and rectangular composite surfaces. The situation has improved for the designer, but the fundamental limitations remain.

There will be more on this later, but one of the fundamental problems that designers face, sooner or later, is breaking loose from

*If you are thinking right now "but I can choose to work with Bézier surfaces as well", remember that Bézier surfaces can be represented by a NURBS definition as a special case; just like a circle is a special case of an ellipse.

†Synonyms are sculpted shapes and organic shapes.

the checkerboard lay-out of surface patches. SolidWorks has a nice demonstration (Figure 2.3 on page 18) suggesting to have come a long way in this, by matching a curve trimmed surface to four other patches with tangent plane continuity. It is possible to recognise the work of Celniker and Welch [1992] in this. This is a vendor-provided demonstration, so the presented case may well be arranged in favour of the algorithm behind it; it is not difficult to imagine situations where it cannot be expected to perform as well, if at all. In their conclusion, Celniker and Welch [1992] warn that a "discretisation error for curve constraints results in the surface pulling away from the constraint curve when its ability to exactly represent the solution curve is exceeded". However impressive, the transition is still only tangentially continuous (G^1), and only up to a bounded precision; even the positional continuity (G^0) cannot be absolute. Farin [2002a, section 16.9] reminds us that

> Trimmed surfaces should be seen as an "engineering" extension of tensor product surfaces*. That is to say, they are no panacea to all surface problems either. Consider, for example, the problem of joining two trimmed surfaces in a smooth way. If they are to join along trim curves, there is no known method to ensure exact tangent plane continuity between them, as was the case for standard tensor patches. Such smoothness questions must be dealt with on a case-by-case basis, which is clearly not very desirable.

Obviously, the capabilities of main-stream MCAD software do not suffice for the geometric modelling of class A surfaces, at least not for high quality products with a shiny finish. For this job, dedicated software exist. *ICEM* is a company that specialises in class A surfaces, with its Surf product. *Alias*, although having its main focus on the entertainment industry, is nonetheless a strong competitor in this segment with SurfaceStudio. *UGS PLM Solutions* has its own ImageWare, which is sold as a component in NX. Also *Dassault Systèmes* has a dedicated module for Catia, called Automotive Class A (ACA).

By browsing the technical specifications of these products, it appears that the underlying surface mathematics is not different from pure MCAD software. We exclusively encounter the same old quadrilateral surface patches. This software fights the symptoms of the limita-

*The tensor product is a method to generate quadrilateral surfaces using curve methods. NURBS surfaces are an example.

tions of quadrilateral patches rather than to address the cause[*]. Their special features consist generally of the following:

1. Extended surface fitting and reverse engineering capabilities. This is used to create surfaces from the three-dimensional (3D) scans of physical clay models, in which much of the styling still takes place.
2. Thorough surface manipulation techniques and constraint solvers. This is for tuning the surface patches to work out unwanted bulging and attaining a high degree of geometric continuity across patches [†]. This is a time consuming task.
3. An array of analysis and surface inspection tools. The human senses of touch and vision are extremely sensitive to small geometric discontinuities. Typical tolerances are 0.001mm for positional continuity and $0.05°$ for tangential continuity. The tolerance on curvature continuity is dependent on the application, and this continuity is best evaluated visually on screen using different kinds of curvature plots and zebra plots.
4. Photo-realistic rendering. To judge the aesthetic value of the surface model and the design in a realistic setting. This is important, as there is no way back once the molds have been manufactured. The next opportunity for evaluation is the real thing coming out of the factory, in great numbers.

We conclude that NURBS rule the surface definitions in both data exchange and commercial software. Apparently, breaking loose from the pack is not attractive as this will break interoperability with other computer programs. That will not be tolerated by customers. The alternative is to advance the standard as well. But since the cost of making fundamental changes in programs as colossal as geometric modellers is so enormous, and the number of influential parties is significant, the standard is so inert that advancing it with regard to surface definitions cannot be done overnight. Quadrilateral NURBS surface patches are the standard, and they will remain being the standard for many years to come. We should now talk about why that should bother us.

[*]This is understandable, as the class A surface models are passed on to general MCAD systems for further engineering.

[†]If the composite surface is not at least curvature continuous (G^2), sharp transitions will be visible in the reflection of an image in the surface. This is perceived as an aesthetic defect of the surface.

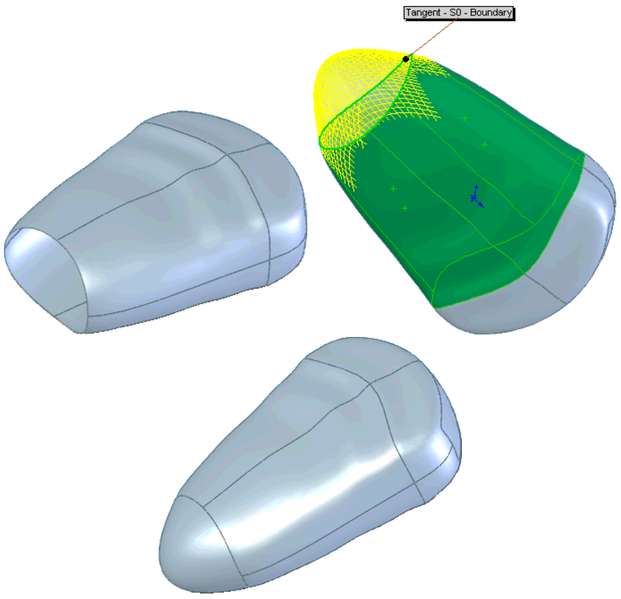


Figure 2.3: A curve bounded (trimmed) quadrilateral surface can be generated to close an open shell, with (approximately) tangent plane continuity. (Source: promotional material, *SolidWorks*.)

2.2 Shortcomings

Although NURBS surfaces are highly versatile at first sight, they are not as versatile as one would like in many practical modelling and design situations. The regular structure of a single patch causes problems in modelling high quality surfaces that do not match this regularity in one way or another. This will be the subject of the next subsection. To reduce these problems, one may try to model the geometry with several smaller patches. This introduces other difficulties, which will be discussed successively.

2.2.1 Control versus Quality

Let us first define control and quality. Later we will see that they are in conflict.

2.2.1.1 Control

The shape of a NURBS surface is controlled by a regular quadrilateral mesh of control points, known as the control net or the polygon net. The shape of the surface loosely and smoothly follows the control points, but in general does not pass through them. The control points are ordered in complete rows and columns in the network, like the elements in a matrix. Each single control point has influence on the shape of a quadrilateral sub-domain of the NURBS surface.

The size of the sub-domain is proportional to the polynomial degree of the surface and the spacing of control points. The larger the degree, the larger the area that each control point has influence on[*]. The degree is a constant that counts for the entire patch. When control points are positioned close to each other, the area of the sub-domain is smaller.

Internal knot spacings also have a say in the size of the sub-domain. But since there is just one knot vector that is shared by all columns and another one that is shared by all rows in the patch, it can only be used to redistribute the size of the sub-domain among the control points in complete rows and columns, not individual control points.

Altogether, it is only the control point spacing that has local control over the size of the individual sub-domain that the control points have influence on.

2.2.1.2 Quality

Surface quality can be defined as a (rather subjective) measure of how closely the shape of a surface resembles the intended shape. In the term "shape" we include the position and several derivatives everywhere on the surface. In naval architecture, a surface of sufficient quality is called a fair surface, and the process of detecting imperfections and smoothing them away is called fairing. Usually, quality includes smoothness, or geometric continuity of several orders. This is a mathematical condition, and can be checked for analytically.

However, geometric continuity in itself is not a sufficient measure of quality. Undesired undulations in a surface are also regarded as poor quality. The easiest example is a region in a surface that is supposed to be planar. The curvature should be zero (infinite radius of curvature), but in practice the surface may undulate slightly above and underneath the intended plane. When control points are close to

[*]A higher degree means also that more sub-domains of neighbouring control points overlap, so that a point on the surface depends on the position of more control points.

each other this will be obvious as ripples in the surface, but otherwise it will be like a weak swell and be less obvious. In the case of a planar region it is not particularly difficult to detect undulations (the curvature will be non-zero and change sign) and neither will be the solution (constrain relevant control points to the plane).

But mostly, imperfections are less obvious. Often, the intended shape is not planar but curved, and not even of constant curvature. Deviations from the intended shape do cause distortions in curvature, but they may be subtle. They are harder to detect because there is no prescribed value of curvature (like zero in the planar example). In general, detection of imperfections requires careful human evaluation, for instance of computer simulated reflections in the surface. Elimination of imperfections generally requires a human expert and is an iterative process.

2.2.1.3 Conflict

High shape control and high surface quality, as it happens to be, are often in conflict. In the case of curves, there is a lower threshold on the number of control points by which a curve can be forged into the desired shape. Every extra control point will cause a deviation of the curve from its ideal shape if it is not positioned exactly on the right spot. The most tranquil shape is obtained with a minimum of control points.

For curves, it is not complicated to reach an ideal state, as the spacing of control points can be varied freely along the curve*. But with quadrilateral NURBS surfaces however, the spacing of control points is not completely free, because control points are organised in complete rows and columns. In one region of a surface patch the shape may require more control points for its definition; but addition of one or more complete rows and/or columns introduces superfluous control points in other regions, where they cause noise and make surface fairing more difficult. See Figure 2.4.

The conflict is not always instantiated due to a local need for extra control. The contours of the surface patch itself can also squeeze control points tighter together than one would prefer, as in Figure 2.5. The number of undulations that are possible is not higher in narrow regions of the patch than elsewhere, but the required accuracy of the positioning of control points is much higher; it is relative to the distance between them. To illustrate the point by exaggeration: where

*In practice, the ratio of successive control point spacings will seldom exceed 2:1, as it can cause unwanted bulging or "over shooting" of the curve.

control points are one metre apart, an error of one millimetre may easily go unnoticed; it is only 1‰ of the spacing. But where control points are only a millimetre apart, the same quantitative error amounts to 100% of the spacing, causing very noticeable turbulence. It requires much more effort to obtain high surface quality in tighter parts of a patch than elsewhere.

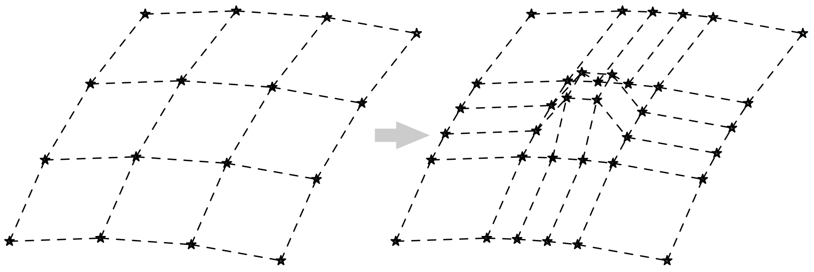


Figure 2.4: The control net of a surface patch. When extra control is needed in some local area of the patch, it causes superfluous control in several other places.

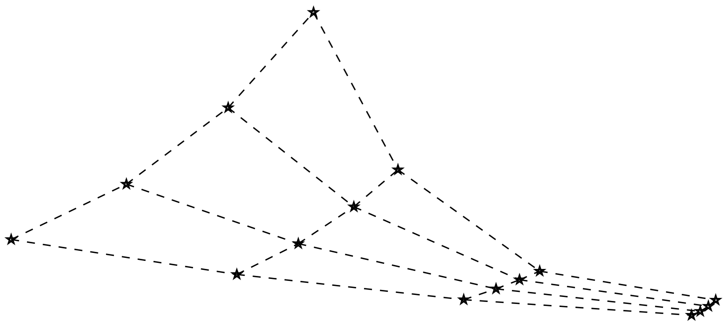


Figure 2.5: Attaining high surface quality is also difficult when control points are packed together due to narrowing of a surface patch.

2.2.2 The World is not Quadrilateral

Due to the common NURBS surface patches being quadrilateral and having a regular control net they are not always flexible enough to fit the desired shape. Geometries that require some irregularity in the patch lay-out are often denoted as having arbitrary topology.

Arbitrary topology means that there are no restrictions on how patches may connect. Although such geometries are very common, modelling them does not go without problems.

2.2.2.1 When Control is Never Sufficient

Suppose the shape that is to be modelled looks like the shaded image in Figure 2.6 below. It has a quadrilateral contour, with which a quadrilateral NURBS patch should be nicely compatible. But the shape contains a semi-torus like feature. The row- and column-wise organisation of control points does not match up (Figure 2.6, left). Every regular quadrilateral control net with a manageable number of control points will only be able to define an approximation of the shape, and it will be a bumpy one. The bumps can be decreased by increasing the number of control points beyond being manageable (Figure 2.6, right), for instance for a computer-generated surface fit, but the limitation is fundamental and will not go away.

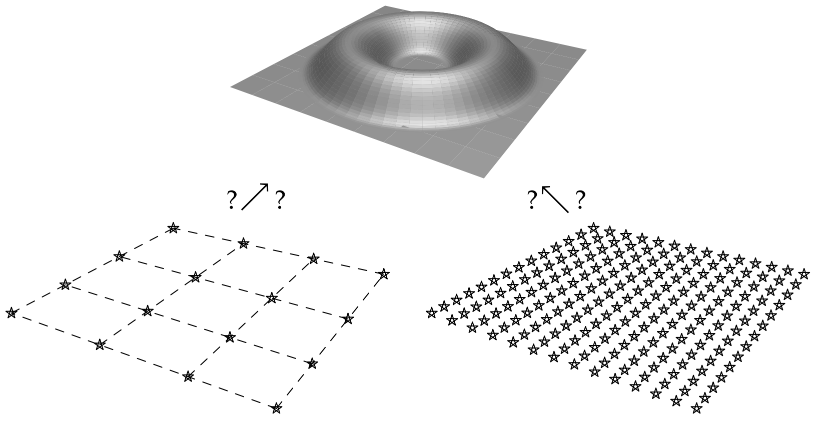


Figure 2.6: The control points do not line up with the geometry that is to be modelled.

This is a constructed problem, but there are many practical situations in which the structure of control points does not suit the geometry that is to be modelled. A work-around is to try and find out whether individual surface features can be modelled by individual patches, then to trim away the pieces that overlap and stitch the collection together.

Figure 2.7 below shows the work-around for our example. The toroidal feature can be modelled by bending a control net four times, like a snake biting its tail. The base patch is modelled separately and can be kept simple in this case. The contour lines of the toroidal feature are projected onto the base patch to form bounding curves for trimming away (disregarding) the surplus surface.

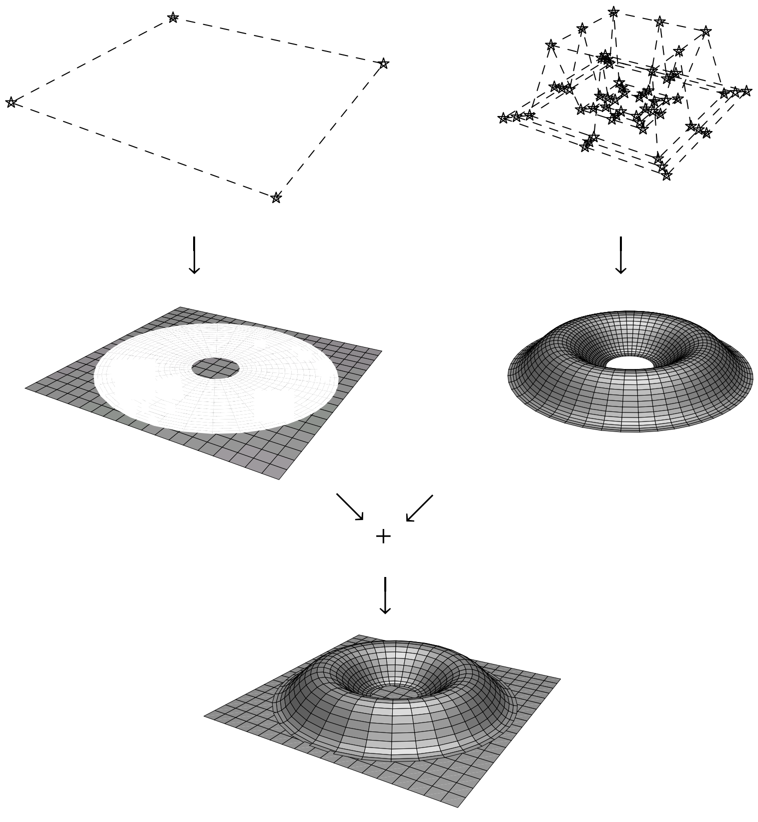


Figure 2.7: Modelling features individually is a work-around for the case when control points do not line up correctly.

A problem with trimmed surfaces is that obtaining geometric continuity is non-trivial. Even positional continuity is a difficult matter because the patches do not share boundary control points. This means that associativity between the surfaces, i.e., the property that one surface remains compatible with its neighbour during changes, is not available *per se*. In the case of Figure 2.3 on page 18, associativity may be implemented by rebuilding the trimmed surface whenever a boundary surface is modified, based on the model history. However, this may affect the correctness of any references made to the trimmed surface.

2.2.2.2 Degenerate Surface Patches

Many of the regularity problems that are encountered in practice could be solved if triangular surface patches would be available. A classical example is the nose of a zeppelin, where triangular surfaces meet in a fan formation around a shared corner.Triangular Bézier patches have been around from the beginning, and the corresponding de Casteljau algorithm is arguably more elegant than the tensor product definition of quadrilateral patches [Farin, 2002a, chapter 17]. Even the triangular equivalent of NURBS patches have been developed [Qin and Terzopoulos, 1997]. But as we have concluded earlier, they are rarely used in the industry. Instead, a triangular patch is often simulated with a degenerate quadrilateral patch.

There are two ways in which a patch can be degenerate. One is by having a collapsed side, called the degenerate side (Figure 2.8). The other is to have a degenerate corner, which is a corner that is positioned so that its corner feature vanishes, i.e., its adjacent edges are flush (Figure 2.9).

Degenerate patches introduce a number of complications. A degenerate corner involves constraints on some of the control points to preserve the degeneracy throughout surface manipulations. On the other hand, degenerate sides tend to clutter control points near them (similarly to Figure 2.5 on page 21) with the adverse consequences for surface fairing that were discussed in Section 2.2.1.3. Neither solution is symmetric.

Most importantly, degenerate patches are the cause of numerical problems. The tangent vectors to a bi-parametric surface are defined as the first derivative of the two surface parameters. At a degenerate corner, these are collinear by definition, along the patch boundary. Generally, the normal vector is defined as the normalised cross product of the two tangent vectors. In this definition the normal vector degenerates into an expression of the form $0/0$ because the tangent vectors are linearly dependent. Fortunately, we are saved by the fact that the closest neighbouring control points to the degenerate corner are coplanar with the latter, and the plane in which they lie happens to be the tangent plane at the degenerate corner. So the tangent and normal information can be derived, but by another route.

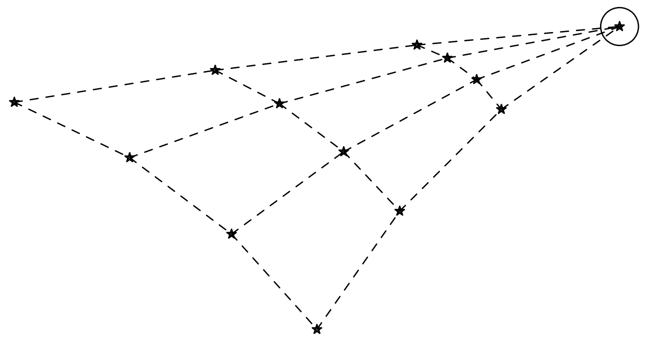


Figure 2.8: A surface patch with a degenerate side (circled).

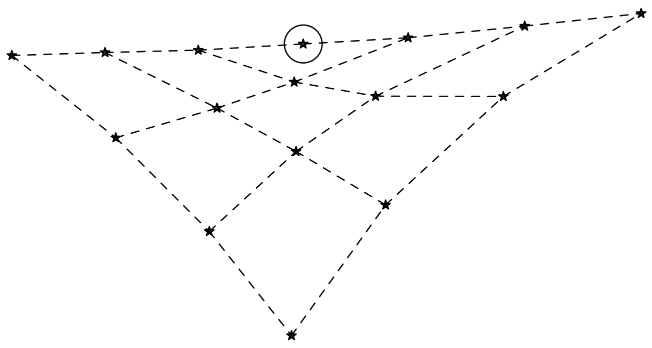


Figure 2.9: A surface patch with a degenerate corner (circled).

The situation is a little worse with patches with a degenerate side, because one of the tangent vectors is singular. Here too, the normal vector can be obtained in an alternative way, but only if all control points in the column next to the degenerate side are coplanar with the latter. This is not a natural consequence like in the case of a degenerate corner, so this condition has to be provided for explicitly.

With respect to radius of curvature, the situation of degenerate patches is much more implicate, and as far as I know it is not yet fully resolved. For more information the reader is referred to [Wolter and Tuohy, 1992] and [Yamaguchi, 2000].

Missing surface derivatives cause problems for common algorithms for tasks like the computation of intersections. Undefined normal vectors cause problems for rendering algorithms that are used for the generation of shaded or photo-realistic images.

2.2.2.3 Non-Regular Compositions

There is a way to eliminate the need for triangular patches, and with it degenerate patches, by transferring the non-regularity from the number of patch sides to the valence of corners, i.e., to the number of patches meeting at a shared corner. A sphere-like geometry can be modelled with six quadrilateral patches where three patches meet at every corner — like blowing up a dice so that it turns into a marble.

Irregular holes with n sides in a composition of quadrilateral patches can be filled with n quadrilateral patches that meet half-way along the sides of the hole, as in Figure 2.10 on the current page. Note that n may be three, by which we have an alternative for a three-sided patch that, unlike a degenerate patch, is symmetric.

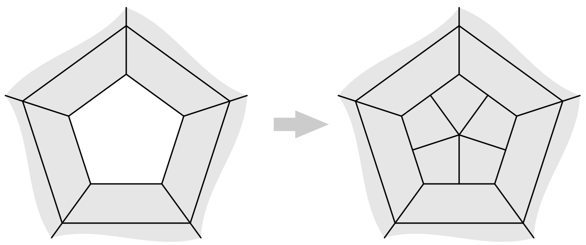


Figure 2.10: Filling n-sided holes with n quadrilateral patches. In this example, $n = 5$.

The downside of corners with a valence different from four is that there are no simple rules to obtain geometric continuity at and around such corners. In the case of filling a hole, constraints can be computed automatically, although support for this is not in the mainstream geometric modellers[*]. But mostly, surfaces are custom constructed and geometric continuity has to be worked for; as one designer expressed himself on an Internet forum for automotive body engineering,

> "It is pretty tricky sometimes just to get up to curvature continuity in a patch corner. You can chase yourself around a corner for days, until it is right."[†]

[*] *ICEM Surf* supports automatic filling of holes with up to twelve edge curves.
[†] `www.eng-tips.com`, thread "Mathematic criteri[on] of CAD Class-A surfaces".

2.2.3 Change is Badly Supported

In the shape of a surface, we differentiate between global shape features and local shape features. Global shape features are features that can be recognised from a distance, as, for example, the slightly curved shape of body panels in the side of a car. For fairness, the defining data should be sparse. Local shape features are details, like, in the same example, the impression that makes room for the handle in the door of a car. To define this, a higher density of surface data is needed. A surface that does not appear to have local features may still need dense defining data, to forge it into some prescribed shape within tight tolerances. In such a case, one may see the generally fair shape as the global feature, and the compliance with the tolerance as local features.

This is a manifestation of the fairness *versus* control conflict discussed earlier, and a fundamental problem with any method that defines a continuous surface based on discontinuous data*. Unfortunately, the NURBS method amplifies this fundamental problem, because extra shape data is not only introduced due to local shape features, but due to its regular topology as well. This is why some publications refer to NURBS surfaces as being rigid, i.e., discouraging changes.

The presence of both global and local shape features has implications for the ability to change the design at a global level. Mainstream MCAD systems provide one or more of the following means to change a NURBS surface:

- Change the position of a single control point.
- Change the position of a selection of control points, all in the same direction over the same distance.
- Direct surface manipulation. The user "grabs" an arbitrary point on the surface and drags it around, and the system computes new positions for relevant control points [Fowler and Bartels, 1993].
- Change the shape of a single curve in the set of defining curves of a lofted, ruled or swept surface.

Suppose one needs to modify the global shape of a surface containing also local features. As soon as the first single control point is manipulated, the fairness of the global shape is lost. When a selection of control points is moved, this may preserve the shape within the selection, but is likely to introduce defects along the border of the selection.

*This includes interpolation of curves, see Section 4.6.

Direct manipulation is only slightly less damaging than single control point manipulation, because a select number of control points is moved simultaneously. But if the defining data is moderately dense, global fairness is lost all the same. In addition, direct manipulation can cause unexpected bulging near the grabbed point. Ruled and swept surfaces are incapable of directly describing local shape features, and if a lofted surface has local features, it is usually by means of closer placed curves*. Global fairness, if it has been obtained at all, is lost when the first curve is manipulated.

Alternatively, the local feature may be modelled as a separate surface that fills a trimmed-out hole in a larger surface that covers the global feature, like in Figure 2.7 on page 23. This does not make things any easier, because when the larger surface is changed by control point manipulation, the connection between the features is likely to be lost and the local feature be left behind. When trying to move the local feature back in place, one may find that the hole in which it fitted, has been deformed by the global change.

2.3 Chapter Summary and Look Ahead

The standard NURBS surface method comes short in the fact that it is so complicated to produce high quality surface models of any but the trivial geometries. The process is characterised on the one hand by finding a balance in the position of superfluous control points, caused by the regularity of the control net, and on the other hand by reducing geometric discontinuities of several orders to within the tolerances. This task requires special software and expert operators.

Besides quality being expensive to attain, both in equipment and man-hours, quality does not persist throughout design changes in the general case†. Therefore, striving for surface quality has to be postponed until after the design has evolved to its final state, or improvement of the design is hindered by the chance of loosing the investment in quality. Especially because the act of geometric designing is about evolving concepts into new and better ones, the described limitations of NURBS surface modelling should make you wary of the value of computer-aided design, with an emphasis on *design.*

Briefly, I have the following two demands on CAGD:

*Sometimes, it is possible to bring local features into these surfaces by manipulating the control net that results from a ruling, sweep or loft. These are however lost completely when the surfaces are rebuilt upon a change in one of their defining curves.

†Special deformation techniques exist that try to preserve surface quality.

1. the freedom to design the shape that I want, and
2. the freedom to change my mind and my design.

These demands are not addressed to my satisfaction by the current state of the industry, at least not both of them in the same product. The art, i.e., literature, has fortunately advanced somewhat further. Chapters 3 and 4 review how our first demand is addressed using the paradigms of approximation and interpolation respectively. Chapter 5 reviews solutions for our second demand. We will see that most solutions are tailored to the approximation paradigm. Chapter 6 considers a commercial implementation that adheres to the interpolation paradigm, and that at least satisfies our first demand. It reports upon research on how this implementation can be extended to satisfy our second demand as well.

CHAPTER 3

Approximation of Arbitrarily Connected Points

This chapter gives an overview of some of the methods that have been suggested in the literature to remove the regularity requirement of control nets, by which sculpted shapes with arbitrary topology can be modelled. We can identify two camps in this endeavour.

On the one hand there are subdivision schemes, inspired by the observation that in the limit of repeated knot insertion, the control polygon of a uniform B-spline curve converges towards the curve itself. Subdivision schemes generalise this principle to sculpted surfaces with arbitrary topology by repeatedly cutting the corners and edges off of arbitrary polyhedra. A surface constructed by this scheme is always a faceted approximation of the limit surface.

On the other hand there is the camp that tries to achieve a truly continuous parametric description of sculpted shapes with arbitrary topology, preferably using standard surface patches at a lower level. These constructed surfaces are usually called surface splines, not to be confused with spline surfaces. Synonyms are G-splines and geometric continuous patch complexes.

The methods of both camps are fairly good at describing sculpted shapes that are tangentially continuous (G^1). However, attaining higher orders of geometric continuity appears to be much more complex.

Before looking more closely at these two approaches we should mention the hierarchical B-spline surfaces proposed by Forsey and Bartels [1988]. They do not really fit into this chapter because they do not solve the arbitrary topology problem, but they do address the control vs. fairness issue to a great extent and therefore deserve to be mentioned. Hierarchical B-spline surfaces allow the designer to add control points locally without having to add complete rows or columns of them, although they remain structured. As a bonus, the higher levels of detail follow shape manipulations on lower levels, preserving consistency. Therefore, the designer can switch between the design of local features and global shape at will. These surfaces are a multi-resolution extension of the standard B-spline surfaces and allow practical design of complex yet fair geometries, exemplified as shown in Figure 3.1 below. Clearly, one can get a long way even without completely arbitrarily connected control points.

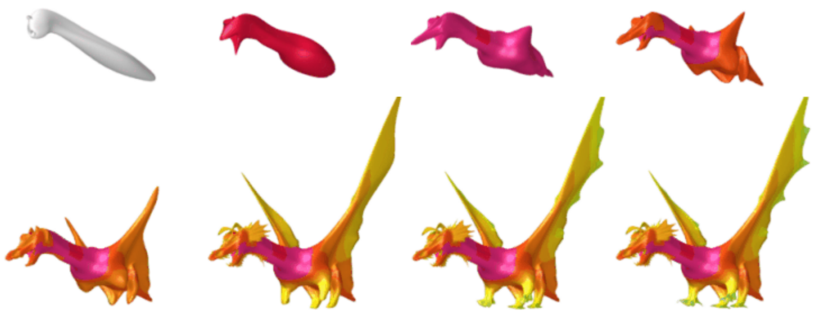


Figure 3.1: The hierarchical levels in Forsey's dragon example (source: `www.cs.ubc.ca/nest/imager/`). The head and the body are modelled with one hierarchical B-spline surface each and pasted together.

3.1 Subdivision Surfaces

There are many different subdivision schemes. We will limit ourselves to the discussion of two well known schemes, namely Doo-Sabin and Catmull-Clark. They are also the first subdivision schemes for polyhedra of arbitrary topology, and were jointly introduced as early as 1978.

3.1.1 Catmull-Clark Subdivision Surfaces

Catmull and Clark [1978] have proposed subdivision rules for both the generalisation of bi-quadratic and bi-cubic B-spline surfaces to nets of arbitrarily connected control points. But since Doo and Sabin [1978] in their evaluation of these schemes presented better rules for the bi-quadratic case, only the bi-cubic form is referred to as the Catmull-Clark subdivision scheme.

Initially, a polyhedron is considered, formed by the control points and their connections. At each refinement step, every n-sided face of the polyhedron is subdivided into n smaller faces. For this, one new vertex is computed for each old face, each old edge and each old vertex. The rules are as follows:

1. New face points are formed by finding the centroid of the vertices of the old face.
2. New edge points are formed by taking the average of the end points of the old edge and the new face points on either side of the edge.
3. New vertex points are computed by taking the following average:
$$\frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n} \tag{3.1}$$
where n is the valence of the old vertex, Q is the average of the new face points of all faces adjacent to the old vertex, R is the average of the midpoints of all old edges incident on the old vertex and S is the old vertex.

Figure 3.2 on the following page illustrates one refinement step. Note that after the first step, all faces are quadrilateral. The vertices that have a valence different from four after the first refinement step represent the irregularities in an otherwise regular mesh of quadrilateral cells. These irregularities will persist through all further refinements and are therefore called *extraordinary points*.

As mentioned before, the Catmull-Clark algorithm generalises regular uniform cubic B-spline surfaces, which in the regular case are curvature continuous (G^2). Indeed, Catmull-Clark surfaces are G^2 everywhere — except at extraordinary points. Catmull and Clark report that a saddle surface suffers from an imperfection at the extraordinary point if the valence is high. The analysis of the geometric continuity at and near extraordinary points performed by Doo and Sabin [1978] reveals that the curvature is actually infinite at this point.

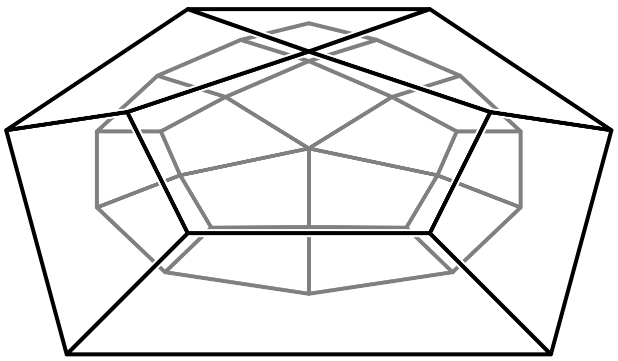


Figure 3.2: The Catmull-Clark algorithm: There is a new vertex for every old vertex, edge and face.

3.1.2 Doo-Sabin Subdivision Surfaces

Regarding the generalisation of bi-quadratic B-spline surfaces to nets of arbitrarily connected control points, Doo and Sabin [1978] report better behaviour of the scheme presented by Catmull and Clark [1978] than they did regarding the bi-cubic case. At the same time, they present even better rules that produce ideally behaved surfaces for all valences. Hence these surfaces are called Doo-Sabin subdivision surfaces.

Refer to the example in Figure 3.3 on the next page. At each refinement step, every n-sided face of the polyhedron is replaced by a new and smaller face with the same number of sides that is embedded in the same plane as the old face. New faces also appear in the place of old vertices and old edges, by connecting the vertices of the face-faces across the old edges, and around the old vertices. Note that after the first step, all vertices have valence four.

The vertices of the refined polyhedron are computed as follows: For each n-sided old face with vertices $\mathbf{v}_i$, new vertices $\mathbf{v}_i^{(1)}$ are defined as a weighted sum of the old vertices,

$$\mathbf{v}_i^{(1)} \equiv \sum_{j=1}^{n} \alpha_{ij} \mathbf{v}_j$$

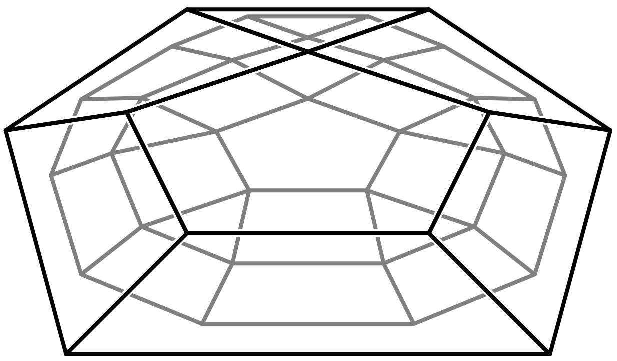


Figure 3.3: The Doo-Sabin algorithm: there is a new face for every old face, edge and vertex.

with weight factors defined as

$$\alpha_{ij} \equiv \begin{cases} \frac{n+5}{4n} & \text{when } i = j, \\ \frac{3+2\cos\frac{2\pi(i-j)}{n}}{4n} & \text{when } i \neq j. \end{cases}$$

Doo-Sabin subdivision surfaces are tangentially continuous (G^1) everywhere, just as regular bi-quadratic surfaces. Extraordinary points appear at the centres of faces that are not quadrilateral and near control points that have a valence different from four. After the first refinement step, when all vertices have valence four, the final number and position of the extraordinary points are therefore known — they are invariant across any further refinement.

3.1.3 Interpolation

Just like the control polygon can be found for which a B-spline curve interpolates a given set of points, the surfaces that were discussed above can be forced to interpolate selected control points. To top that, Nasri [1997] describes recursive subdivision surfaces that interpolate B-spline curves, where the control polygons of the interpolated curves are identified as sequences of edges in the original control polyhedron. Interestingly, with this we have almost tumbled into the subject of Chapter 4. Zorin *et al.* [1996] propose the "modified Butterfly" subdivision scheme, which is interpolating by design, and produces C^1 surfaces of arbitrary topology.

3.1.4 Value for Engineering Applications

Two limitations of subdivision surfaces that could prevent incorporation in engineering applications are, firstly, the inherent difficulty of designing surface features such as cusps, creases, and darts, and secondly, poor evaluation efficiency. The first limitation has been eliminated by the introduction of non-uniform recursive subdivision surfaces by Sederberg *et al.* [1998]. Regarding the evaluation issue, Stam [1998] presents a means of exact evaluation of Catmull-Clark subdivision surfaces, including all surface derivatives, at arbitrary parameter values without subdividing. The direct availability of surface derivatives is very important, e.g., for the visualisation of curvatures, for finding intersections, for detection of interference and for generation of numerical control (NC) codes for computer-aided manufacturing (CAM). The technique initiated by Stam has been extended by Wang *et al.* [2000] to include the non-uniform scheme of Sederberg *et al.* Fortunately, much has changed on the hardware front since the inception of subdivision surfaces, which has also much to say for interactive applications using subdivision surfaces. Bolz and Schröder [2002] show how Catmull-Clark surfaces can be rapidly evaluated by optimising for modern computer hardware.

Even without these improvements, subdivision surfaces have become popular in the entertainment industry; Figure 3.4 on the facing page illustrates their success. But not so in Engineering. Peters [2003] observes

> "Yet, at present, no one seems to be ready to base a full car body design or the crafting of special purpose lenses on this paradigm."

The reason is that there are much higher demands on surface quality in Engineering and Industrial Design than there are in the entertainment industry. The demand for G^2 surfaces is relatively low in the entertainment industry, which is primarily concerned with the modelling of phantasy characters. Yet G^2 surfaces are an absolute requirement for the design of high quality class A surfaces. Therefore a scheme is needed that generalises at least cubic B-spline surfaces, such as the Catmull-Clark scheme.

By recursive subdivision, n-sided holes at the irregularities in the control polyhedron are recursively filled with ever thinner rings of regular polynomial pieces. The problems related to the irregularities in the control polyhedron of a Catmull-Clark surface are thereby swept further and further towards the extraordinary points. In video games, artifacts[*] may be pushed beyond the relatively coarse display resolution before they can become noticeable, or they are just accepted. In motion pictures, artifacts may be retouched away if necessary. In Engineering and Design however, artifacts are unacceptable.

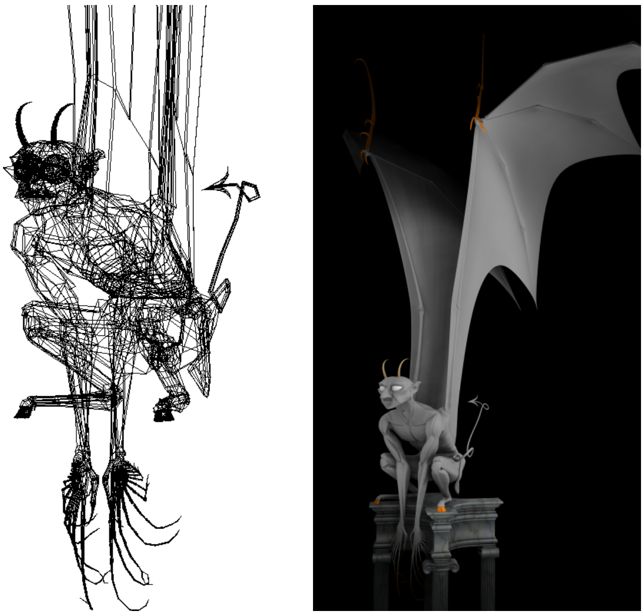


Figure 3.4: Subdivision surfaces in the entertainment industry; detail of the control polyhedron (left) and the final rendering (right). (Source: *Blender Documentation*, `download.blender.org/documentation/`)

Peters [2003] gives several illustrations of curvature misbehaviour of Catmull-Clark surfaces around extraordinary points. When valence is moderately high, curvature may change sign and increase without bound closer to the extraordinary point. This means that the surface can show to be concave, even when the control polyhedron is convex. Attempts to bound the curvature exist, but these result in oscillations of curvature and thus ripples in the surface.

Sabin [2002, sec. 12.7] warns for "first step artifacts", in which "the original topology of the [control] polyhedron shines through" on high end visualisation devices, "in the form of ripples whose spatial frequency is that of the original [control points]". Presumably, Sabin is

[*]In CAGD, the term *artifact* is often used to denote a visual imperfection or a shape deficiency, like an unintended side-effect of an algorithm.

speaking of the above described curvature problem at extraordinary points, but it is not exactly clear.

An inconvenience when modelling with Catmull-Clark subdivision surfaces is caused by unequal contraction of the surface depending on the valence rather than the geometry of a vertex. This is due to the extra dependency on the valence of the S-term in (3.1) and the effect can be observed already in Figure 3.2 on page 34. Although there is a high interest in subdivision surfaces and although the remaining problems are being worked on, they will not be supported by the mainstream MCAD systems any time soon.

3.2 Surface Splines

The surface spline presents itself to the designer, not as a set of individual patches that each approximate a different regular control net, but as a continuous surface, approximating one large irregular control net. The surface spline can be of arbitrary topology, and it may be closed and therefore function as the boundary of a solid[*].

3.2.1 Tangent Plane Continuous Surface Splines

With regards to surface splines, the earliest reference that I found is to Chiyokura and Kimura [1983]. They describe a modelling system called MODIF, which later led to the Japanese CAD/CAM system DESIGNBASE [Farin, 2002b], owned by Ricoh. Their approach is actually one of curve interpolation, which is the subject of Chapter 4. It is interesting to note that the two paradigms regarding modelling with arbitrary topology, namely approximation and interpolation, share roots this way.

Based on a B-rep in which each edge is represented by a single cubic Bézier curve, Chiyokura and Kimura produce a parametric surface for each face, interpolating the cubic boundary curves. Faces with $n \neq 4$ sides are split into n quadrilateral sections, each filled with one quadrilateral surface patch as in Figure 2.10 on page 26. In order to achieve tangent plane continuity at vertices with a valence different from four, they apply a specially constructed surface patch, which they call the Gregory patch. It is an application of the twist compatibility correction that Gregory performed on Coons patches, to the Bézier

[*]One could count such a closed surface spline as a boundary representation (B-rep), but since the designer manipulates the control points of the surface and not the nodes and edges of the traditional B-rep, it is not quite the same.

form. The resulting patch has twenty control points (see Figure 3.5 below) and is rational. Written in rational Bézier form, the patch is bi-septic (degree 7×7) and the corner weights are zero, which results in singularities [Farin, 2002a, section 22.6].

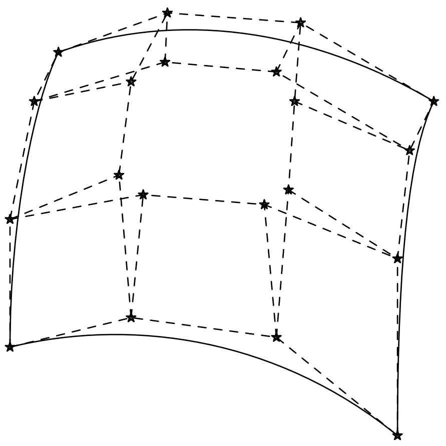


Figure 3.5: Gregory patch in Bézier form, with 20 control points.

Van Wijk [1986] shows that ordinary bi-cubic patches can be used to model some cases of geometries with irregular topology. The control polyhedron is restricted to have only quadrilateral faces, and the valence of vertices must either be odd, or be a mix of three and four. The composition is G^1 across the patch boundaries. Although the degree of the patches is modest, the supported topology is not completely arbitrary.

Sarranga [1987, 1989] allows more topologic diversity at the cost of higher order polynomials. Using bi-sextic (degree 6×6) Bézier patches, he allows three, four or five patches to meet at a corner. The computation of the constraints on many of the control points is performed by a computer algebra system (CAS), by which the patches are made G^1 around the corners.

Algebraic conditions for arbitrarily continuous (G^k) patch complexes with arbitrary valence are formulated in Hahn [1989], based on the theory of differential topology and differential geometry. This publication is accompanied by an application, namely the development of a second order parametrically continuous (C^2) n-sided interpolant [Gregory and Hahn, 1989] devised to fill n-sided holes within otherwise regular C^2 patch complexes. In practice, local constraint systems have to be solved to enforce patch to patch smoothness, which make it difficult to predict the shape of the resulting surface. It seems you have to be a mathematician to appreciate this method.

Loop and DeRose [1989] also propose an n-sided patch, but with a geometrically comprehensible definition. It is a generalisation of Bézier surfaces to patches with an arbitrary number of sides, based on the idea of restricting Bézier simplexes to embedded surfaces, which they call S-patches. See Figure 3.6 below for an example. S-patches

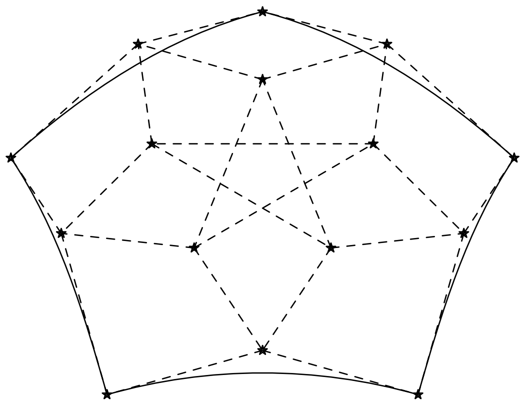


Figure 3.6: An S-patch with five sides and a "depth" of two.

can be geometrically constructed with a de Casteljau inspired algorithm. In [Loop and DeRose, 1990, see also Loop, 1992] the authors use S-patches to propose a generalisation of B-spline surfaces to arbitrary topology. Two schemes are presented, but neither allows the control polygon to be completely arbitrary. The bi-quadratic scheme requires the control polyhedron to have exclusively quadrilateral faces, but the valence of vertices is unrestricted. In the resulting surface, one patch appears for every vertex. The bi-cubic scheme produces one patch for every face in the control polyhedron, and therefore the faces need not be quadrilateral. In this case however, the valence of vertices is restricted to four. The composite surface is G^1 across patch boundaries, see Figure 3.7 on the next page for an example. Stoddart *et al.* [1994] apply the bi-quadratic scheme to surface reconstruction over an unstructured point set.

Apart from the fact that S-patches are non-standard surface definitions, they have the unfortunate property that their computational complexity is an exponential function of the number of sides. For example, a six sided patch has already 56 control points in the cubic scheme [Loop, 1992].

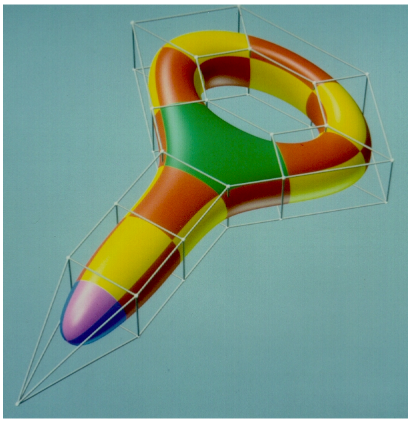


Figure 3.7: A surface spline constructed with S-patches. (Source: [Loop, 1992].)

Zheng and Ball [1997] provide a different generalisation of Bézier patches to 3, 5 and 6 sides of arbitrary degree, with fewer control points. The control points are organised in a flat pattern, i.e., the connections between control points do not cross as in S-patches. For comparison, their six sided patch with cubic boundary curves has only 24 control points. These patches can connect to surrounding quadrilateral Bézier patches to form a first order parametrically continuous (C^1) composite surface. In [Zheng, 2001] the corner twist constraint is removed for patches with an arbitrary number of sides, analogously to the modification of quadrilateral Bézier patches by Chiyokura and Kimura [1983], see Figure 3.5 on page 39. This simplifies the construction of surfaces with arbitrary topology.

Peters [1994] discovered that by using two Doo-Sabin refinements on an arbitrary control polyhedron, the irregularities are topologically sufficiently separated from each other, by which the problem of constructing a smooth patch complex with standard non-rational quadrilateral patches becomes simpler. In addition, the refined control net can be used to determine the position of most of the control points of the patches, so that no systems of equations need to be solved*. Where the control polyhedron is regular, bi-quadratic B-spline surfaces are

*In order to guarantee tangent plane continuity also at the irregularities in the control polyhedron, a perturbation has to be applied to the points in the refined mesh that are edge-adjacent to corners of irregular cells that have a number of sides that is produced, and irregular faces are covered with bi-cubic patches, without singularities. The faces of the control polyhedron need not be planar.

We can give a simplified illustration of the surface spline construction, following [Farin, 2002a, section 21.8]*. Figure 3.8 below shows how one of the n bi-cubic patches, enumerated by i, is positioned to cover part of an n sided cell in the refined mesh. The nodes of the refined mesh are indicated by squares; the solid squares surround the n-sided cell (partly shown). The corner that is shared by all patches that cover the cell, i.e., Bézier point $\mathbf{b}_{33}$, is simply the average of all solid squares

$$\mathbf{b}_{33}^{(i)} \equiv \frac{1}{n}\sum_{j=1}^{n} \mathbf{d}^{(j)}$$

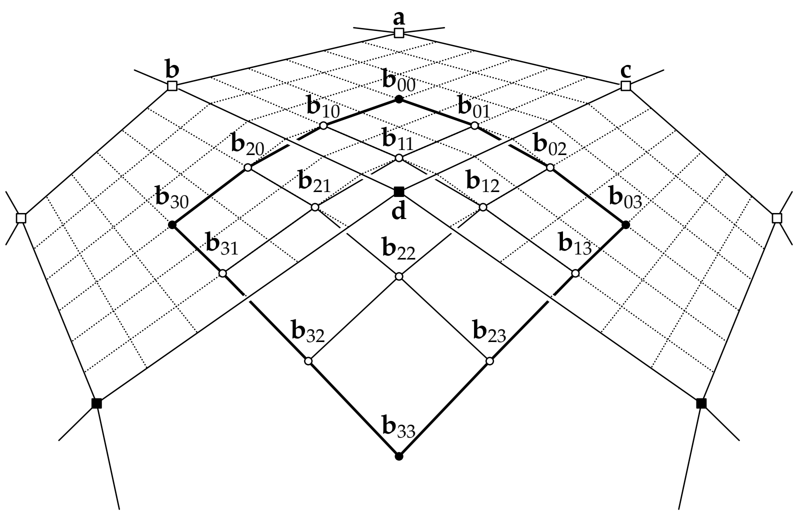


Figure 3.8: Surface splines: how Bézier control points of a cubic patch i are derived from refined mesh nodes.

If we disregard an extra dependency on n for simplicity, the "outer" two rows and columns of Bézier points lie on bi-linear patches (visu-

both even and greater than four. This perturbation is the cause why this construction needs two refinement steps, so perturbations of different irregularities do not conflict with each other.

*The simplification leaves out the perturbation discussed in footnote * on the page before. Some errors that were found in [Farin, 2002a] have been corrected.

alised by dotted lines) that interpolate the refined mesh. For the four points in the top corner in the figure, their computation amounts to

$$\begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} \\ \mathbf{b}_{10} & \mathbf{b}_{11} \end{bmatrix} \equiv \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{6} & \frac{5}{6} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} & \mathbf{b} \\ \mathbf{c} & \mathbf{d} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & \frac{5}{6} \end{bmatrix}.$$

The Bézier points $\mathbf{b}_{30}$, $\mathbf{b}_{20}$, $\mathbf{b}_{31}$, $\mathbf{b}_{21}$ and $\mathbf{b}_{03}$, $\mathbf{b}_{13}$, $\mathbf{b}_{02}$, $\mathbf{b}_{12}$ are found analogously.

The three remaining Bézier points are determined as follows:

$$\mathbf{b}_{32}^{(i)} \equiv \mathbf{b}_{23}^{(i+1)} \equiv \mathbf{b}_{33}^{(i)} + \frac{4}{3n} \sum_{j=1}^{n} \cos\left(j\frac{2\pi}{n}\right) \frac{\mathbf{d}^{i+j} + \mathbf{d}^{i+j+1}}{2},$$

and

$$\mathbf{b}_{22}^{(i)} \equiv \begin{cases} -\sum_{j=1}^{n}(-1)^j \mathbf{e}_{i+j-1} & \text{if } n \text{ is odd,} \\ \frac{-2}{n}\sum_{j=1}^{n}(n-j)(-1)^j \mathbf{e}_{i+j-1} & \text{if } n \text{ is even}^*. \end{cases}$$

where $\mathbf{e}_i \equiv (1-c)\mathbf{b}_{32}^{(i)} + c\mathbf{b}_{31}^{(i)}$ and $c \equiv \frac{2}{3}\cos(2\pi/n)$.

In its standard form, the composed surface interpolates the centroid of the faces of the control polyhedron, but the algorithm can be extended to interpolate the control points instead. Conic blends (e.g., circular roundings) can be produced with rational patches. The composed surface is G^1 across the patch boundaries and follows the control polyhedron in a predictable way. By varying blend ratios in the refinement steps, the roundness of the surface can be controlled, so it can be made to follow the control polyhedron closer, or even interpolate it with a crease. Farin [2002a, section 21.8] notes that the composite surface is not equivalent to the Doo-Sabin limit surface. Figure 3.9 on the next page gives an example of the modelling capabilities of G^1 surface splines.

Loop [1994] shows that one Doo-Sabin refinement step suffices when using three-sided patches [†], at the cost of higher degrees (at most quartic patches, of degree 4). Over regular regions of the mesh, a biquadratic Bézier patch may be used in place of four quartic triangular patches.

Peters [1995b] gives an improved version of his original method, which reduces the number of patches by a factor four and uses less

[*] Both [Peters, 1994] and [Farin, 2002a] subscript $\mathbf{e}$ as $\mathbf{e}_{i+j}$, which seems to be an error.

[†] This is because no perturbation is needed.

complicated formulas. This method also needs just one mesh refinement step, made possible by using a mix of four-sided patches and three-sided patches. Optimally, bi-quadratic four-sided patches are used where the mesh is regular and cubic three-sided patches are used where the mesh is irregular. The resulting surface is G^1 and lies within the local convex hull. Upon user request, the surface spline may also be built up exclusively of three-sided patches or four-sided patches, but in the latter case a deficiency in tangent plane continuity may arise and the convex hull property does not hold for all blend ratios.

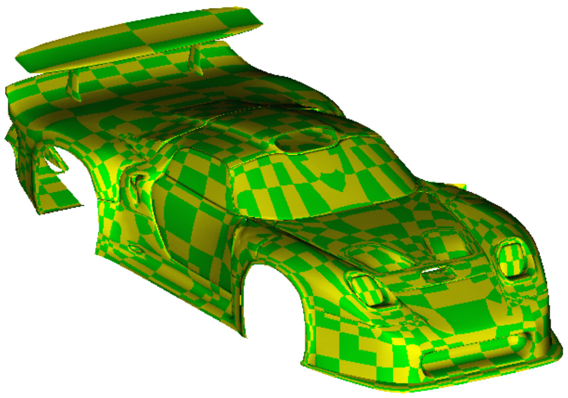


Figure 3.9: Surface splines: an example. (Source: Peters, `www.cise.ufl.edu/research/SurfLab/surf_spl/`)

As we have seen, three-sided patches are non-standard and rarely supported in established CAD/CAM systems. Therefore, a third variant is given in [Peters, 1995a], which improves continuity and shape properties when the user chooses to model entirely with four-sided patches. Also the convex hull property is guaranteed in nearly all cases. In this variant, patches are bi-quartic (of degree 4×4).

As an aside, we can mention that Gonzalez-Ochoa [1997] demonstrates that surface splines are suitable for interactive modelling. A multi-resolution approach is implemented in the localised-hierarchy surface splines proposed in [Gonzalez-Ochoa and Peters, 1999] to represent level of details without fragmentation. Eck and Hoppe [1996] demonstrate the use of surface splines for surface reconstruction based on scattered point data, obtained from laser range scanners.

3.2.2 Curvature Continuous Surface Splines

So far, only tangentially continuous surface splines have been discussed. Obtaining higher orders of continuity is more difficult. Since Catmull-Clark subdivision is G^2 wherever the control polyhedron is regular, it may be worthwhile to try and transfer the technique discussed so far to the Catmull-Clark subdivision scheme. Indeed, Peters [2000] shows how this can be done. This method uses just one Catmull-Clark subdivision step, by which the mesh cells are all quadrilateral. These cells correspond with one bi-cubic NURBS patch each, which join C^2 wherever the control polyhedron is regular. Here the composed surface is equivalent to the Catmull-Clark limit surface. At the extraordinary points the surface may be only G^1, but with finite curvature, which is better than the Catmull-Clark limit surface.

In the quest of completely G^2 surface splines, we note that Peters [2002b] explains

> "if we are not concerned about the degree, it is straightforward to generate G^k free-form surface splines for any k."

We are however concerned about the degree, because higher polynomial degrees can cause numerical errors*, require more storage and most importantly, they increase the cost of computations, say of intersections. What we also shy away from is to solve global constraint systems.

It seems that G^2 surface splines cannot be constructed without having to do at least *some* constraint solving. Peters [1996] uses subdivision to reduce the number of unknowns, and the remaining local constraint system is left to a computer algebra system (CAS) to be solved. The resulting surface spline consists of 8 quartic or octic (degree 4 or 8) three-sided patches for every edge in the control polyhedron. Alternatively, the surface can be represented with quadrilateral patches exclusively, which are then bi-sextic (degree 6×6), or with a mix of bi-quartic and bi-cubic (degrees 4×4 and 3×3) quadrilateral patches where the mesh is regular, and octic three-sided patches elsewhere.

Gregory and Zhou [1999] prove that C^2 surface splines can be constructed from bi-quintic patches. However, Peters [2002a, 2003] reports shape deficiencies caused by the use of quadratic boundary curves.

*Due to the high powers in splines of high degrees, the effect of round-off errors, caused by the limited precision in which computers can handle real numbers, is magnified [Press *et al.*, 1992, Section 1.3]. In unlucky cases, these defects can become visual.

A completely different approach is introduced by Grimm and Hughes [1995], who use the theory of manifolds to construct surfaces of arbitrary topology with guaranteed continuity of any degree (G^k). Instead of matching up patches by their edges, they let surface pieces overlap substantially. The composed surface is compared with an atlas of charts or maps; when navigating, there is another map you can switch to before travelling off your current map. Thus there are no discontinuities in parameterisation across surface pieces, which is advantageous for texture mapping and the definition of smooth paths on the surface. The efficiency is improved by Cotrina Navau and Pla Garcia [2000], by reducing the number of charts. The construction of the manifold is computationally intensive, but it needs only be rebuilt after changes in the topology of the control polyhedron, so that the effects of control point manipulation are quickly incorporated. However, in the G^2 case, the degree of the patches is already 9×9 or higher [Peters, 2002a, 2003].

Inspired by the use of maps, Peters [2002a] manages to reduce the degree of patches considerably, at least algebraically [Peters, 2002b]. In this case, the control polyhedron is subdivided by one Catmull-Clark refinement step. Using a re-parameterisation to a triangular map that is trimmed to four sides, G^2 surface splines are constructed consisting of bi-cubic patches (degree 3×3) and some strips of patches with degree 3×5. The method can be generalised to G^k surface splines. One critique of this approach is that the alignment of iso-parametric lines is distorted when the valence of irregular nodes in the refined mesh becomes large [Peters, 2003]. It is envisioned that to hybridise these techniques with the ones in [Gregory and Zhou, 1999] will be a promising approach, in terms of low degree and lack of algorithm-induced shape deficiencies.

3.3 Chapter Summary and Look Ahead

In this chapter we have seen two different approaches for the definition of surfaces that approximate arbitrarily connected points. Subdivision surfaces are popular in the entertainment industry, but not very well suited for engineering applications. The representation is unsupported by standards for the exchange of model data and suffers from shape deficiencies or limited geometric continuity.

The other approach is based on the assembly of parametric surface patches, some of which are supported by standards for the exchange of model data. They are a better match for engineering applications, with respect to rapid evaluation and compatibility with established algorithms and concepts such as constructive solid geometry (CSG). However, G^2 surfaces still require considerable computational efforts because of high polynomial orders and constraint solving.

As we have seen, there is active development to improve the situation for both approaches. Progressions herein deserve to be monitored.

The next chapter will take a similar look on the competing paradigm of surface construction by interpolation of arbitrarily intersecting curves.

CHAPTER 4

Interpolation of Arbitrarily Intersecting Curves

The methods for the construction of surfaces with arbitrary topology that were discussed in the previous chapter, are based on approximation of control points. They are attempts to generalise the standard surface method of NURBS to surfaces of arbitrary topology, and can thereby be called the leading paradigm.

Although approximating algorithms arguably produce more pleasing surfaces than interpolating algorithms [Peters, 1995a], control point manipulation is not the natural way for surface stylists to interact with their design. This is confirmed by the fact that software producers start making interfaces to surface manipulation based on curves, as indicated on page 15.

This chapter however, is about a different paradigm that is based on curves all along. The design of a sculpted shape, regardless of its application, usually starts with a few characteristic pencil strokes on a piece of paper. As the design progresses, more of these 'feature curves' are added to define higher detail or to remove ambiguity. A trained brain can interpret the resulting 'wire-frame' sketch as the representation of a surface or a solid. But although wire-frame modelling has been supported in systems for CAD from the very beginning, to turn a wire-frame model into a complete surface model is not a trivial task.

A first step towards such a surface model is of course the possibility to 'fill in' the curve-bounded cells of the wire-frame with surface patches. The transfinite interpolants such as Coons patches and variants [Farin, 2002a, chapter 22] were obviously designed with this application in mind, but although these patches stem from the advent of CAGD, it has taken long before they were successfully used to model surfaces with arbitrary topology.

This chapter describes the history and background of the technology on which the experiments of Chapter 6 on page 91 are based, and therefore we will evaluate contributions in a little more detail than we did in the previous chapter.

4.1 Interpolation of Cubic Bézier Curves

Let us take the same point of departure as we did in the discussion of surface splines in Section 3.2, with the publication by Chiyokura and Kimura [1983]. For details the reader is referred to that prior discussion. Of relevance here is that although they do support the modelling of solids with a G^1 boundary and arbitrary topology, they put strict requirements on the curves defining the wire-frame; each curve must be a cubic Bézier curve and extend over one edge of one mesh cell in the wire-frame, no longer and no shorter. This makes it relatively easy to obtain tangent continuity across patch boundaries, but it makes it harder for the designer to create fair feature curves.

4.2 Interpolation of Continuous Curves

Jensen *et al.* [1991] come with a break-through on the computation of cross-boundary derivative data, by which curves may be arbitrarily defined and extend to any length over the surface. They describe the internals of a geometric modeller developed by Evans & Sutherland Computer Corporation (E&S), called the Conceptual Design and Rendering System (CDRS)[*]. As this system is meant to be a tool for computer-aided conceptual design (CACD), it provides an algorithm to fit a cubic interpolating spline curve with quadratic end conditions (i.e., being $C^{2\dagger}$) to a sketched curve. The sketched curve is obtained

[*]The CDRS was acquired by Parametric Technology Corporation (PTC) in 1995, and involved E&S employees joined PTC. It is unclear whether PTC incorporated the curve interpolation technology into its own product Pro|ENGINEER, or whether they in fact just bought expertise.

[†]On the subject of geometric vs. parametric continuity, the authors mention the following important point:

by sampling the cursor position as the designer draws the shape in two orthographic windows, and is thus piecewise linear. Curves that collectively are to define a smooth surface are allowed to have 'tails' that extend outside the surface boundary, and any number of curves may meet at a common point.

It appears that Jensen *et al.* support two types of surface patches: a triangular and a rectangular patch. By constructing multiple patches on the same network of curves, surfaces with non-regular topology can be modelled, although the curve network cannot be completely arbitrary.

The triangular surface patch is bounded by three intersecting curves that are interpolated by a triangular Coons-type interpolant, more precisely, the C^1 approach by Nielson (for this and other triangular Coons patches, see Farin [2002a, chapter 22.9]). The triangular surface cannot have internal curves.

The rectangular surface patch, bounded by four intersecting curves, can have internal curves but they must subdivide it into quadrilateral cells, i.e., the internal topology must be regular. This set is then interpolated by a spline-blended Gordon surface, modified to adhere to cross-boundary derivative information. The surface is also *compatibility corrected* with regard to corner twists. A Gordon surface imposes parameterisation constraints on the curves: a curve that intersects a number of other curves, must do so in a way that the parameter values for these curves at the intersection points are all equal. These constraints are in this system unlikely to be satisfied, firstly because 'feature curves' must be allowed to be placed freely, and secondly, curves may extend outside the surface to participate in other parts of the model. The authors adopt a re-parameterisation method for the curves, such that the constraints can be met without modifying the shape. The resulting quadrilateral surface is C^2 internally, and G^1 across its boundary.

> [. . .] which is not widely recognised is that when transfinite surfaces are applied to curves which have geometric, but not parametric, continuity to some level, the surface does not inherit that level of continuity. For example, a Coons patch constructed from curves that are G^1, but only C^0, will in general be both G^0 and C^0.

See also the definition of C^2 in the Glossary on page xvii.

4.2.1 Method Outline

When enough curves are sketched to form a wire-frame representation, surface patches can be generated which smoothly interpolate the given curve data. This involves the following steps:

1. A set of curves is selected, in between which the surface is to interpolate: three boundary curves for a three-sided surface, or four boundary curves with optional internal curves for a four-sided surface, according to the previously mentioned constraints. As we understand, this is the designer's responsibility.
2. Data incompatibilities are resolved. Curves that are supposed to intersect, but do not, are made to. The authors are unclear whether this step is performed automatically or with manual intervention.
3. A topological data structure is built automatically, inspired by the B-rep; see Figure 4.1 on the next page. The use of B-reps is common in solid modelling, but vital as well for this method of surface modelling. Data directly related to the surface consists of references to the curves in the selection, and for each of the curves, references to the intersection points that mark the start and end of the curve section that is of interest to the surface. For each curve in the selection, an ordered list of intersection points is maintained. And for each intersection point, a record is maintained of incident curves and surfaces.
4. If a Gordon surface is being constructed, a re-parameterisation of the curves is computed to satisfy its parameterisation constraints.
5. With the use of the topological information established in step 3 above, cross-boundary derivative information is computed. This information has to be presented in the re-parameterised form, and the authors find it necessary to produce this via a vector-valued spline function that produces normalised vectors in the tangent plane that are orthogonal to the boundary curve. This orthonormal vector and the normalised tangent vector along the curve are scaled, and their vector sum yields the necessary derivative information. The scaling factors are defined by two individual scalar-valued spline functions, with the property that they produce a cross-boundary derivative vector that is equal to the tangent vector along the crossing curve at the intersection point, disregarding the sign. For details see [Jensen *et al.*, 1991].

6. With the re-parameterised curves and the cross-boundary derivative information, an interpolating surface can be generated.

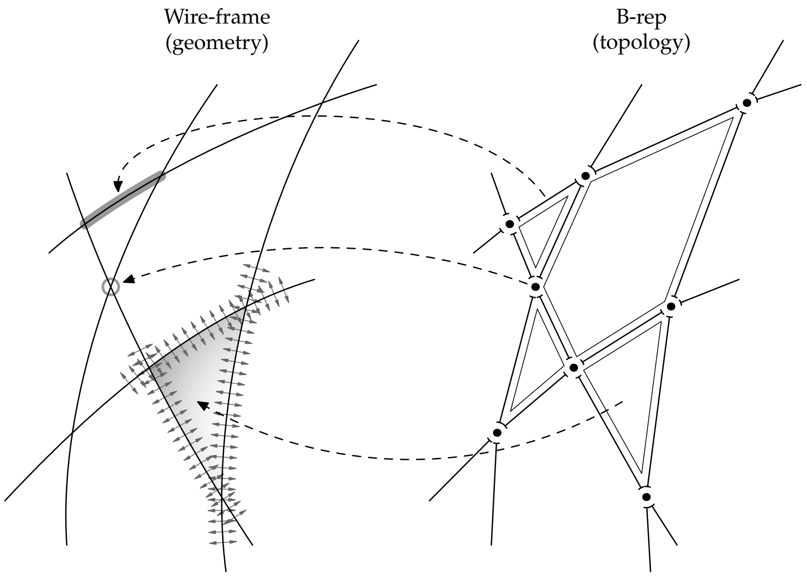


Figure 4.1: An exploded view of how a B-rep can be used to support a wire-frame model, for the construction of a surface with arbitrary topology. On the right side the topologic elements of *node*, *edge* and *face* are shown, i.e., a common B-rep. Dashed arrows indicate references from the topologic elements to their geometric representations, which are derived from a wire-frame representation, displayed on the left. Partial tangent ribbons are also shown, used for constructing the surface representation of faces. Note that each curve in the wire-frame is shadowed by a string of edges in the B-rep. Mesh cells with more than four sides, as displayed here, were not supported in the implementation by Jensen *et al.* [1991], but were made possible in later implementations by Michelsen [1995] (see Section 4.4) and Koelman [1999] (see Section 4.5).

The surface patches so constructed, collectively interpolate the wire-frame. The resulting surface is C^2 where possible and G^1 elsewhere, or just positionally continuous (G^0) where knuckles are intended.

A blind spot in this article concerns step 2 on the preceding page, which we would have liked to know more about. In a realistic design, curves are likely to have a large number of points where they are supposed to intersect with other curves. In general, it will be by chance if two such curves really do intersect. To resolve all incompatibilities in a curve network is a difficult task, whether it is done automatically or by hand, and likely to be iterative in nature. It will often require shape modifications, which has consequences for the smoothness of the curves in particular and for the aesthetics of the design as a whole.

4.3 An Application to Conceptual Design

Van Dijk [1994] describes the conception and implementation of a similar design tool, called Fast Shape Designer (FSD), at the Faculty of Industrial Design Engineering, Delft University of Technology. His research aims exclusively at providing computer support for the conceptual design phase of an industrial design task, i.e., CACD. This is fundamentally difficult, because the user should be able to quickly sketch his ideas, upon which the system should present back the rendering of a 3D model, which may then spark new ideas in the designer. A prerequisite for *idea generation* to take place is that this cycle can be iterated through quickly enough, and without too much technical attention from the designer, such that his creative mind can work without interruption or distraction.

The effort displayed by van Dijk suggests that he is of the opinion that the Conceptual Design and Rendering System (CDRS), discussed in the previous section, does not support idea generation well enough, but he mentions successful use of that system for concept evaluation. Similarly to CDRS, the FSD provides a curve sketching interface. But instead of drawing projections of the space curve in two orthogonal views, the curve is sketched in perspective directly on a freely positioned plane. The algorithm allows adaptive sketching, in which the shape of the curve is altered by additional strokes. This of course only produces plane curves. When true space curves are needed, the author proposes to sketch on a curved surface, but we note that this does not provide the same level of freedom in sketching, and this two-stage procedure cannot be quick enough to support idea generation that involves space curves.

To display shaded images of the evolving model in real-time, the FSD applies hardware accelerated rendering with the *Open Inventor* toolkit, which builds on the graphics language *OpenGL*, both developed by Silicon Graphics Incorporated (SGI). This toolkit has only direct support for sculpted surfaces in the NURBS format[*], which has

[*]It would likely have been possible to display non-NURBS as well, using lower level elements. But that would put a significant extra burden on the programmer, and

led to the requirement that the FSD surfaces must be, or be convertible to, rectangular NURBS surfaces. Part of the design philosophy for the FSD was that the user would be offered to sketch design curves (i.e., feature curves) and that the modelling technique would be direct. A direct modelling technique allows manipulation of surfaces directly, versus indirect techniques by which a change of shape is effected by manipulating control points that lay in the neighbourhood of the surface. Therefore straight application of the NURBS scheme was ruled out, and transfinite interpolation was adopted.

Being a system for conceptual design, it seems that the FSD is meant to produce surfaces with a minimum of defining curves. Thus the interior area of transfinite patches, for which per definition no shape control handles exist, can be quite large. Practice has shown that the shapes that are produced by the system can differ from what the designer expected. Therefore, van Dijk [1994] offers the designer a choice between three different patch types, all of them G^1 across their boundaries, that produce different internal shapes*. They all have roots in Coons' transfinite patch, and algorithms exist to convert them to (a set of) NURBS patches. But a requirement of these algorithms are that the type of curves which the patches are defined on, are semi-uniform cubic B-splines.

In order to model with arbitrary topology, at least a three-sided patch has to be provided. Although there are no obvious technical hinders in extending the applied techniques to three-sided interpolants, visualising them with hardware acceleration is another matter. Again because Open Inventor has built-in support for rectangular NURBS patches, it was decided to represent a three-sided patch with a degenerate four-sided patch, by inserting an extra corner half-way one of its boundary curves. This can be repeated to produce a two-sided patch and even a one-sided patch. Not all the transfinite patch types behave satisfactorily when they are degenerated though, and the OpenGL rendering only succeeds courtesy a discretisation into small triangular facets at the lowest level. See also Section 2.2.2.2 on degenerate patches.

would possibly lead to slower execution because more has to be done in software. In addition, sticking with standard NURBS gives better prospects for data exchange with other systems.

*We note the following. Because of van Dijk's dislike of indirect surface schemes, he loses control of the internal shape. A lot of effort is put in regaining that loss by offering multiple transfinite interpolation schemes, and even inventing new ones. But the kind of control this results in, cannot be called *direct*, and is considerably more restrictive than control point manipulation.

To produce a feature curve of the required type, the sketch data is fitted with a uniform cubic B-spline by means of a basic least squares method. The number of control points of the curve is 15 by default, but can be adjusted interactively by the user, who thus determines the closeness of the fit and the fairness of the curve.

Sketched curves can be assembled into a curve network for interpolation, which involves the construction of a B-rep data structure. This is initiated by the user, who indicates which curves are supposed to intersect each other. Automatically, a *link* object is generated for each (supposed) intersection, which serves as a vertex in the data structure. Contrary to Jensen *et al.* [1991], van Dijk [1994] is clear about what is done when curves that are meant to intersect each other, do not. Using data from the *link* object, curves can be *snapped* on to each other, to make them intersect each other precisely. Like with the CDRS, making a large network consistent in this way can be a task with no end — which is why the FSD accepts inconsistent networks just as well. When design curves do not intersect each other, adjacent surfaces will deviate from them at their corners, at the cost of geometric continuity.

This seems like a good design decision for a CACD system. Paper sketches, which support ideation well, are fast but imprecise. A computer system that is to replace paper sketching, should accept the same imprecise input and make the best of it, rather than produce correct models and slow down the user by requesting high precision input. The difference is only that inconsistency is so much more obvious in computer renderings than in artist renderings. Designers will have to develop a tolerance against these defects, to not let it distract them, and continue with the creative process.

An interpolating patch is constructed by selecting three or four design curves and pressing a button. This causes *edge* objects to be created along the design curves, in between the *link* objects. An *edge* object owns two NURBS curves, defined in global coordinates. One of them traces the design curve in between the parameter values contained in the two *link* objects, but possibly deviates from it at the ends if the *link* object corresponds to a crossing and not an exact intersection. This NURBS curve defines one of the boundaries of the interpolating patch. The other NURBS curve is positioned right next to it, and their difference defines the tangent vector across the boundary. Its control points are derived from tangent ribbons along the design curves. These were constructed in the form of vector valued cubic Bézier curves, successively converted to B-spline curves by reparameterisation and multiple knot insertion, to match them with the design curves[*]. The NURBS curves on opposite sides of the patch are given equal knot vectors and an equal number of control points by knot insertion.

With this information, an interpolating patch can be generated. A corresponding *patch* object is created, which takes on the function of a *face* element in the B-rep data structure. This *patch* object contains the NURBS surfaces to which the patch is converted, so that they can be sent to the graphics library for visualisation, or be exported to standard data formats.

4.4 An Application to Ship Hull Design

The use of transfinite surfaces that interpolate an arbitrary network of curves is not only advantageous in conceptual design. This is illustrated in the PhD thesis by Michelsen [1995], who describes an application to ship hull design. His work is implemented as a set of Fortran-77 archives for the program package *I-ship*, developed at the Department of Ocean Engineering of the Technical University of Denmark[†].

Michelsen envisions the following sequence of operations:

1. Preliminary shape design with a set of unconnected curves, e.g., representing intersection curves in parallel planes, like the station curves in a lines plan.
2. Definition of connecting curves in other planes, like the water lines in a lines plan, while fairing the existing set of curves.
3. Establishment of topology information. There is an attempt to automate this step, but manual inspection is necessary, followed by manual correction in ambiguous cases.
4. Building a patch-work by filling in the mesh cells with transfinite surfaces.

[*]It seems that this is simpler than the approach of Jensen *et al.* [1991], probably because it does not need to be as general since the curves are known to be B-splines instead of completely arbitrary.

[†]Michelsen was dictated in his choice of programming language by interfacing with the pre-existing *I-ship* system. It was not a happy choice, as he reportedly had difficulties implementing efficient data structures in that language.

4.4.1 Curves

Michelsen does not go as far as to implement a sketching interface — after all, the focus is not on *idea generation* but rather on modelling precision — but he is also of the opinion that approximating splines like the B-spline variants are not intuitive enough for non-mathematicians to work with. He chooses a cubic interpolating spline, so curves can be manipulated with "control" points *on* the curve, here called *offset points*. These splines are equivalent to the collection of short cubic Bézier curves, here called curve segments, one between each pair of successive offset points. The location of the free Bézier control points (the ones that determine the tangents at the ends, known as anchors in the graphic world) is determined to produce (by default) an overall G^2 curve by solving a linear system of equations*. In addition, several kinds of tangent constraints (including knuckles) can be specified at each individual offset point. It is for example possible to have a master/slave relation between the tangents of the two segments of a curve that meet at a common offset point, in which the tangent of one segment automatically takes on the value of the tangent of the other segment.

Whether an interpolating spline is indeed more intuitive to work with than an approximating spline, is subject to debate. As noted by Jensen *et al.* [1991], the choice of the number and placement of offset points is critical: a curve with few offset points can bulge-out unexpectedly when it is manipulated, and a curve with many offset points is hard to fair. Other disadvantages in comparison to the B-spline is that control is global, i.e., manipulation of a single offset point changes the entire curve, and there is danger of oscillation, which makes it hard to model straight sections in a curve. The bulging and oscillation can be reduced by changing the way in which the curve is parameterised, but this also makes the curve affine variable† and, as noted by Michelsen [1995], there is not one parameterisation that suits all situations.

*Note that although the B-spline curve is approximative, direct manipulation of B-spline curves is perfectly possible, as shown by Fowler and Bartels [1993] (also discussed in [Farin, 2002a, section 9.3]). In fact, this method is even more direct than the interpolating spline, as one is free to "grab" the curve anywhere along it, not just at an offset point. In addition, the linear system of equations that is to be solved for direct manipulation of B-splines is generally smaller than for the interpolating spline, since the number of control points that need to be moved is equal to only the degree of the curve+1. This implementation might have been better off with a common B-spline curve definition.

†which means that the system of equations must be solved again after any affine transformation of the offset points.

As an additional aid against oscillations and excessive bulging, Michelsen allows curves to be split into shorter sub-curves. Each sub-curve is represented by an individual interpolating spline, with a G^1 (or less) connection to its neighbours. This effectively allows a discontinuity of curvature at the joint between sub-curves.

For display, the Bézier segments are efficiently discretised by adaptive subdivision using de Casteljau´s algorithm, which reduces the number of curve evaluations in regions of low curvature, while producing high detail in regions of high curvature.

One important advantage of using interpolating splines for wire-frame modelling is that intersections between curves are absolutely accurate. Offset points are stored directly, and two intersecting curves share the same offset point at their intersection. Thus even round-off errors are non-existent at the intersection.

4.4.2 Data Structure

Michelsen chooses Baumgart´s "winged edge" data structure to implement a B-rep, essential for the creation of the patch-work surface. The existence of sub-curves requires a simple additional data structure that administers the relations between offset points, sub-curves and curves.

4.4.3 Cross-Boundary Derivatives

As usual, cross-boundary derivative information is needed in order to obtain a G^1 connection across patch boundaries. For this, tangent ribbons are constructed in a way similar to the approach of Jensen *et al.* [1991], discussed in step 5 on page 52.

4.4.4 Surfaces

This is the first implementation in this series that supports wire-frame cells with more than four sides. These cells are filled with an n-sided transfinite patch defined by Gregory [1983]. A patch of this kind is composed of a blend between n interpolants that simultaneously match two adjacent edges for each corner of the patch. The blending functions, defined on barycentric coordinates, suppress the contribution of interpolants as the opposite side of the polygonal boundary is approached. This n-sided patch formulation applies to $n > 4$, but formulations for $n = 4$ and $n = 3$ are similar. Explicit formulations for $n < 3$ are not provided in this implementation, 2-sided cells are chosen to be filled with a degenerate 3-sided surface.

Surfaces constructed with these patches are C^2 within patch boundaries, and G^1 across patch boundaries. As Michelsen notes in his thesis, whether tangent plane continuity is sufficient in all design situations, is an open question. But since the boundary curves themselves are C^2, the discontinuity of the curvature across the boundary is limited and can be decreased by adding more curves, such that patches become smaller. Thus "almost curvature continuity" can be achieved, or ϵG^2. The approach of Jensen *et al.* [1991], discussed in Section 4.2, performs even better in this regard, as regular quadrilateral regions of the network are truly C^2, independent of their size, by the use of Gordon surfaces.

Support for n-sided patches removes constraints on the regularity of the wire-frame, which is of great value. The price is incompatibility with standards for loss-less data exchange with other CAD systems.

4.4.5 Practical Experiences

Michelsen's thesis offers a full supply of examples, and experiences with this modelling method are well documented. The system seems to perform generally well, but it is worth noting that not every consistent and fair wire-frame will result in a satisfactory surface model. For details the reader should refer to Michelsen [1995, section 6.3], but in short the following problems can be expected:

- The aesthetic quality of patches degrades with high variance in the boundary data, such as interior corner angles, edge lengths, or derivative information.
- Wire-frame cells with fewer than three sides can occur in practice, for which surfaces are ill described.
- When patch corners appear inside the convex body, the surface will extend outside its boundary, a situation that is surely unwanted.
- When curves intersect at small angles, the surface normal at that point becomes unstable. Thus the surface is sensitive to small changes in these curves.

These problems can be resolved by changing the wire-frame. This is not a fortunate solution for ship hull design applications, as naval architects are used to work with planar curves embedded in standard intersecting planes, i.e., the curves of a traditional lines plan. An extra space curve here and there is annoying at the least and might be confusing. The real problem though, is that the model needs to be inspected in detail to identify defects of this kind, after which fixes need to be devised manually.

It is also noted that some situations, like the connection of the hull of a modern yacht with its keel, would gain from practices known from CSG, like boolean combinations of individual wire-frames, and the definition of surface features such as edge blends and chamfers. The use of Euler operators is recommended for future work, to maintain a consistent topology during changes in the wire-frame.

We conclude that the system has much potential, but a wide commercial application seems to be lacking; probably due to the academic setting in which *I-ship* is developed, maintained and supported.

4.5 Integrated Fairing: The H-rep Concept

A Dutch company by the name of SARC has commercial success with its system for ship hull design called *Fairway*. *Fairway* supports both *ab initio* design of new hulls, digitalisation of existing lines plans and even digitalisation of full-scale ship hulls by means of photogrammetry. Its conception and implementation is described in the PhD thesis by Koelman [1999]. He denotes the underlying technology as "hybrid model for ship hull representation (H-rep)", to indicate the mergence of wire-frame modelling with solid modelling. Its hybrid nature is clearly visible in Figure 4.1 on page 53. Overviews of the H-rep are also published in Koelman *et al.* [2001] and Koelman [2003].

Instead of asking the user to work through a sequence of steps in order to generate a surface with arbitrary topology, Koelman gives the user a flying start: every new project starts with a model of the starboard side of a minimal initial ship hull, based on specified main dimensions. This initial shape is defined by a contour line (in the XZ-plane at $y = 0$, which is the plane of symmetry), a station curve (embedded in a vertical transverse plane) at half the ships length and deck-line (a space curve). This set of curves is enough for the definition of a complete solid model, and although parts of it can be made invisible (such as the centre plane and the deck), the model always stays a valid solid, by the application of Euler operators — as suggested earlier by Michelsen [1995]. Thus the B-rep data structure serves both the administration of transfinite surfaces and the integrity of the solid.

An important advantage of using a complete initial model and the application of Euler operators, is that users are never bothered with patches or curve selections. They are given the impression of working with a wire-frame model, very much in the tradition of lines plan draughting, with the power of solid modelling. Another valuable improvement is the addition of a combined curve fairing and fitting algorithm. Thus, in an iterative working procedure, the curves in the wire-frame can be made consistent with each other (i.e., curves that should intersect can be made to intersect, at least up to a given accuracy) while giving the hull an overall fairness of production quality*.

4.5.1 Method Outline

The *Fairway* system offers an alpha-numerical interface and a graphical interface. The graphical interface consists of one or more projections of the wire-frame model, in which curves can be manipulated one at a time. Surface patches are not visualised, in fact they are not even computed until strictly necessary. The complete hull surface can be visualised in a separate rendering mode†, but this mode does not support shape manipulation.

Starting with the initial hull, one first brings the existing curves to ones likings, by control point manipulation. If the resulting hull is rendered at this stage, the surface shape will likely not be to ones satisfaction at larger distances from the defining lines. The solution is to add more lines by which the shape can be manipulated, effectively subdividing the patches. This can be formulated as projecting a curve onto the hull, or as intersecting the shape with a plane. Most of the time, the wanted curve is one of the standard lines plan curves, like station curve (embedded in a vertical transverse plane), buttock (embedded in a vertical longitudinal plane), water line (embedded in a horizontal plane) or diagonal (embedded in an oblique longitudinal plane), such that only type and offset need to be specified.

While adjusting a curve, it might happen that it is pulled away from the intersection points that it had with other curves, effectively rendering the wire-frame inconsistent as a surface or solid representation. There is no mechanism that prevents this, or otherwise keeps the surface sound. If this happens, the other curves must be faired through this curve, something that is the responsibility of the designer, but in which the fairing/fitting algorithm is of valuable help. Since curves

*See page 8 for a description of the process of fairing.

†Computer rendering is done by discretisation of the patches and low-level calls into the OpenGL library.

can only be manipulated one at a time, this process is iterative in nature.

This is the way in which changes are made in a design, and it helps when as little curves as possible are present, so the number of curves that are affected by a possible inconsistent region is small. This methodology is completely analogous to traditional manual lines plan draughting, by which ship hulls have been designed for centuries.

When enough curves are present to give the hull its desired shape, and the surface is fair and consistent, the surface itself is finished. Curves that are necessary for computer aided manufacturing can be projected onto the hull, like the profiles of frames, girders and bulkheads, as well as butts and seams of the shell plating.

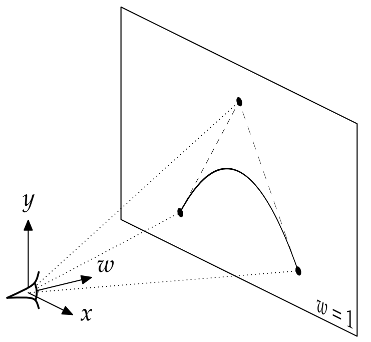
4.5.2 Curves

The curves used in Koelman's implementation are all NURBS curves, because of their ability to represent exact conic sections. The type of curve is specified by the user, with the choice of straight line, circular, parabolic, elliptic, hyperbolic and general 3D B-spline curve.

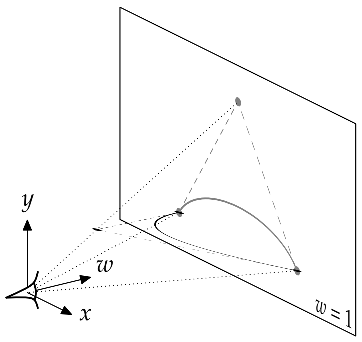
Though the weight factors of the individual control points can be used as shape modifiers of the rational curve, they are less intuitive to work with than just to move control points around. The extra freedom can even be confusing to designers. Therefore it was in this implementation decided, to hide the weight factor from the user in almost all cases. The only weight factor that can be specified by the user, belongs to the middle control point of a quadratic NURBS curve with three control points, by which an elliptic arc ($w < 1$, Figure 4.2(b)) or a hyperbolic arc ($w > 1$, Figure 4.2(c)) can be produced. When a circular arc is requested, which is a special case of an elliptic arc, the appropriate weight factor is computed automatically. When a parabolic arc is requested, all weight factors are kept at 1 (Figure 4.2(a)). The weight factors of all other control points of these and all other curves are permanently fixed at 1.

The degree of the curve is also hidden from the user, and automatically kept as low as possible, to maintain a high level of local control[*]. Straight line segments are of first degree. Conics are of second degree. Free-form curves are given degree three to five, depending on the constraints put on their end-points.

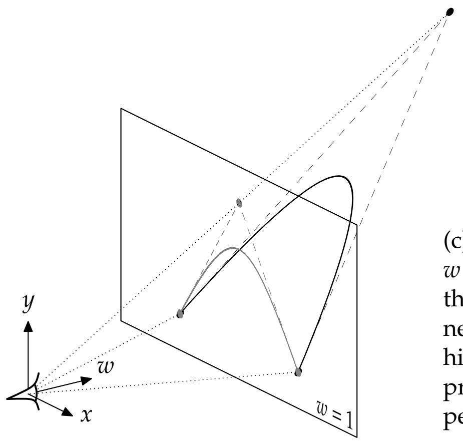
[*]Recall that basis functions have a support over knot intervals that is proportional to their degree (actually, their support is equal to their order) so control points that correspond to basis functions with a large support, control a large portion of the curve.



(a) All control points have $w = 1$, meaning that their homogeneous coordinates fall in the projection plane, which essentially produces a non-rational curve; in this case a parabolic arc.



(b) The middle control point has $w < 1$, positioning it in front of the projection plane in homogeneous coordinates. Consequently, the higher-dimensional curve is also in front of the projection plane. When this curve is projected perspectively, it produces an elliptic arc.



(c) The middle control point has $w > 1$, positioning it behind the projection plane in homogeneous coordinates, and with it the higher-dimensional curve. When projected, this curve produces a hyperbolic arc.

Figure 4.2: The influence of setting weights of non-uniform rational B-spline (NURBS) curves, illustrated on the construction of 2D conic sections, after [Nowacki *et al.*, 1995]. An n-dimensional NURBS curve is constructed by the perspective projection of an $n + 1$ dimensional B-spline curve. If the NURBS control points are defined by coordinates (x, y) and weight factor w, then the higher dimensional B-spline control points are defined as (wx, wy, w), the so-called homogeneous coordinates of (x, y). The term *rational* stems from the fact that the projection (x, y) is produced by the ratio $(wx, wy)/w$.

Like Michelsen [1995], Koelman allows the curves in the wire-frame to be partitioned into curve sections; only he calls the sections just *curves* and the collection of sections a *poly-curve*. But Michelsen had to offer this as a fix to counter oscillation in his interpolating splines, and oscillation is not an issue when using NURBS curves. In this implementation, poly-curves play a functional role: each part of the poly-curve can be assigned an individual curve type, as discussed on page 63. In a similar fashion to Michelsen [1995], constraints on the tangent and curvature at the end-points of the curve sections can be specified by the user, or inherited from the adjacent section in the poly-curve in a *master/slave* relationship. In this implementation, master/slave relations are extended to include unconnected curves, so one curve can be defined as the offset of a master curve, or any combination of its length, breadth and height coordinates made dependent on these values of a master curve. An example of where this is valuable is to make the height of the deck where it crosses the plane of symmetry dependent on the height of the deck at the side, based on a constant transverse deck radius [SARC, 2004].

Although the invention of poly-curves may not be a dramatic achievement scientifically, it is of great practical value to the designer. Not so much in the field of computer animation, where the focus is mostly on the freedom in topology — after all, a model of a dinosaur needs to *look like one*, it does not need to be a cast of a real one— but it is important all the more in engineering fields like naval architecture, where precision and correctness are just as important as topology. Now the sections of curves that lie inside flat regions of the hull (like the FoS and FoB) can be made a straight line by definition[*]. Likewise, the bilge can be defined circular, and water-lines in the bow region can be defined parabolic. The length, place and shape of these curve sections may change, e.g., because of a change in an adjacent section, but their types remain fixed.

Thus, the type of certain shape features in the surface can be fixed during shape manipulation. We note that this property is not surprising to someone accustomed to another modelling methodology that is very common nowadays, and is called feature based design using CSG. There, models are constructed by successively applying different features[†] to a model, like boolean combinations, edge blends, chamfers,

[*]As a comparison, in some other modelling systems, flat regions in the surface can only be achieved by carefully placing a substantial number of control points in the same plane. Deviations from that plane can occur through errors in the manual input, and can go easily unnoticed in the coarse images on a computer monitor.

[†]These operations are recorded in a CSG tree or history tree, by which the model can

taper angles *etcetera*, so features naturally play a centric role. Then again, CSG is not particularly well suited to the design of sculpted shapes, and has great difficulties with sculpted shapes with arbitrary topology. So, the poly-curves of Koelman bring advantages of these two completely different methodologies slightly closer together.

4.5.3 Data Structure

Koelman extends the B-rep data structure in a similar way as Michelsen [1995], to support poly-lines. But instead of using the winged edge data structure to implement a B-rep, he uses the half-edge data structure*, which has a smaller memory foot print and smaller overhead.

4.5.4 Cross-Boundary Derivatives

Tangent ribbons are constructed on demand, after Jensen *et al.* [1991]. For details please see the thesis [Koelman, 1999].

4.5.5 Surfaces

Four sided wire-frame cells are filled with common Coons patches, with Gregory's correction to counter corner twist incompatibility. For n sided cells with $n > 4$, a boolean sum of n corner patches [Gregory, 1983] is used, just as Michelsen [1995] did. The same patch is used if $n = 3$. If $n = 2$, one edge is split to simulate a three sided cell.

A central functionality in this implementation is the interpolation of new curves, and the projection of space curves onto the hull. In effect, this is the problem of intersecting the surface patches with planar and ruled surfaces. Theoretically, this is a complex problem for two reasons. Firstly, the intersection line can consist of multiple elements that can be open or closed loops, possibly with cusps. Singularities can also occur. And secondly, an exact representation of the intersection

be regenerated from scratch. The order of operations may be changed inside the tree (as far as dependencies tolerate) and one can roll-back (and forth) different parts of the tree to insert operations back in time, or change parameters and dimensions in existing features.

*For the half-edge data structure one is usually referred to the standard work of Mäntylä [1988]. It can be shown that this structure is equivalent to a data structure developed by Fjeldaas [1985], which was meant as a relief of the administrative burden of Baumgart's winged edge structure. A modern implementation of Fjeldaas' structure that I produced, turned out to be practically the same as a modern half-edge implementation, like the Half-edge Data Structure Template Library [Brönnimann, 2001].

The only significant difference is that Mäntylä evaluates a pointer where Fjeldaas evaluates a boolean.

line(s) will have an unmanageable high degree. As a reference: the intersection of two bi-cubic Bézier patches is a space curve of degree 324 [Manocha and Canny, 1991].

The approach taken here is approximative, by calculating a manageable number of intersection points per patch , and fitting a NURBS curve through them by means of the integrated fairing/fitting algorithm. In practice, this leads to satisfactory results. This may seem crude to surface experts, but it is important to note that most important information for CAM is in the curves, at least for hulls that will be build in steel, aluminium, or wood. In addition, in a production-ready model, the curve network is so dense that the information covered by the patches has little to add. So the purpose that the patches serve is primarily to hint the shape of newly added curves, and secondarily for rendering and physical model manufacturing.

4.5.6 Fairing

The ability to fit crossing curves to each other such that they intersect, is essential in bringing the model to a description of a sound solid. Curve fairing, of course, is essential for the construction of production quality surfaces, in which unwanted inflections and variations in curvature are removed. Both fitting and fairing are combined in an algorithm that is due to Dierckx [1993].

The curve is fitted to a collection of data points, which consists of patch intersection points as described above, and/or the nearest points on crossing curves. Each of these points can be assigned a weight factor, by which the mean deviation is to be distributed*. The maximum tolerable mean deviation can be specified by the user: if it is 0, the curve is a true fit. The larger the tolerated deviation, the better the fairing and the lesser the fit.

Given a small initial number of control points, the algorithm finds the NURBS curve that firstly has a minimised square of the jumps in the second order derivatives (which is a measure of (un)fairness), and secondly fits the data points best according to the weight factors. If this curve still deviates more from the data points than the user-specified tolerance, one knot is inserted (a control point added) and the optimisation repeated. This process is iterated until the fit is within the tolerance, which results in a fair curve with minimal number of control points.

*This weight factor is not to be confused with the weight factor of control points of NURBS curves.

4.5.7 Continuity Considerations

Whether or not tangent plane continuity, as offered by the applied patch types, is sufficient in all design situations, was left an open question in Section 4.4.4. Of course, in a finished model, the curve mesh is rather dense, and ϵG^2 is well achieved. In addition, as noted earlier, curves are almost all that is needed for production, and curves have no continuity deficiencies.

Nevertheless, Koelman [1999] reports on a practical experiment, to get a better understanding of the importance of cross-boundary curvature continuity. Earlier versions of *Fairway* were fitted with an automated procedure to recognise sets of rectangular patches in a regular formation. These sets were then interpolated by an extended Gordon surface, in the spirit of Jensen *et al.* [1991], to produce larger regions that were G^2 across boundaries. The experiment was to refrain from the extended analysis needed for Gordon surfaces, and to fill the respective cells with common Coons patches. The only noticeable difference was an increase in speed. Conclusion: tangent continuity across patch boundaries is sufficient.

This is not surprising, as the main purpose of patches in this implementation is to hint the shape of newly added curves. The intersection points, on which this hinting is based, are sampled too far apart to reflect the difference of whether there is curvature continuity across the patch boundary or not. The difference in their position is microscopic, and the new curve that is faired through the points is G^2 regardless.

4.6 Limitations

The discussed systems for curve interpolation share an important limitation. As the design progresses and more curves are added to the model, more of its shape gets rigidly defined. The more curves are present, the smaller the surface patches, and the more local shape manipulations get. Since only one curve can be manipulated at a time, and since this manipulation can damage the consistency and fairness of the model, this has serious implications for the modifiability of the model. Even if the desired variation is small in magnitude but covers more than just a few curves, only two expensive alternatives exist: either throw away part of the work already done, or make a mayor investment in curve fairing. In practice, there is a limit on the surface area that can be afforded to be manipulated.

Let us look at a practical example to illustrate the implications of this limitation on the quality of designs. Naval architects designing ship hulls need to worry about a wide range of aspects in which their design must perform as required. Some of them can be predicted analytically, but for others they only have empiric formulae and their experience to rely on. Their educated guesses cannot be verified until in a late stage of the design, some of them by means of model tests or simulation with computational fluid dynamics (CFD). If the results of these tests are unsatisfactory, one *has* to go back and pay for the mistakes. But testing more varieties just to see if the results can be even better, e.g., with different shapes of the bulbous bow, a less profound shoulder or a different inflow to the propeller, is not affordable in most cases. There is time nor money, even with the tests themselves getting better, computing machinery running faster and hardware costs decreasing.

4.7 Chapter Summary and Look Ahead

In this chapter the state of the art was presented, in modelling shapes with arbitrary topology based on the interpolation of curves, including the preceding innovations that inspired it.

The important observation is that the interpolation paradigm has the following limitation, which it shares with the approximation paradigm in a comparable form (see Section 2.2.3). Shape manipulation that affects a larger area of the model is practically only possible in early stages of the design, and shape manipulation in later stages is about local details. Thus, one could say that modelling with these systems is a one-dimensional process, because the design can only evolve in one direction. If it is the wrong direction, it may be best to (almost) start over.

A method that by-passes this problem is called for, so that it will be possible to warp from one shape variation to the other, by which — in our way of speaking — the design process will become multi-dimensional. This is the subject of the next chapter, Chapter 5, which reviews methods for global shape manipulation. As we will see, most of these methods are directed at the approximation paradigm. That is why Chapter 6 focuses on the interpolation paradigm and how said limitation can be addressed in the implementation of Koelman specifically.

CHAPTER 5

Global Shape Manipulation

This chapter seeks an answer in literature to the problems that were described in Section 2.2.3 and Section 4.6. In short, the objective is to make changes in global shape properties of a design, without adversely affecting surface quality and other features. These are changes for which a change in a single element of the shape defining data (control point or curve) is not sufficient or even counterproductive. What we need is a method to change a collection of data, without destroying continuity, consistency and qualitative relations, within the collection itself, as well as between the collection and data that is not changed. In general, this involves each data element in the collection to be changed, moved or shifted by individual values, in a single operation.

One term that is often encountered regarding proposed methods is *deformation*, which makes sense in the field of computer animation, from which many of the methods originate. Computer animation is important in the entertainment industry. In the application of geometric design however, the term is somewhat misplaced, as it would be natural to understand *deformation* as a degradation of surface quality or distortion of the design, not as an improvement. More appropriate would probably have been *formation* or *variation*. Another term is *sculpting*, which is more fortunate, and usually pertains to shape manipulation by means of virtual tools or haptic interfaces.

Earlier reviews are given by Gibson and Mirtich [1997], who survey deformable modelling in computer graphics with an emphasis on finite element models, and Montagnat *et al.* [2001], who focus on deforming a surface towards an existing data set.

5.1 Free-Form Deformation

Bézier introduced the idea of globally deforming a shape through an $\mathbb{R}^n \rightarrow \mathbb{R}^n$ mapping implemented as a free-form n-variate spline [Bézier, 1978]. Sederberg and Parry [1986] popularised this concept in the graphics literature for the case where $n = 3$ to deform geometric objects, and introduced the term free-form deformation (FFD). free-form deformation is a form of "spatial deformation" (another form of spatial deformation is presented in Section 5.4). Simply put, the process can be imagined as defining a 3D shape inside a block of jelly. Then, by flexing the jelly, the shape inside is deformed with it. But FFD is more powerful than the jelly analogy, because the embedding space is infinitely flexible, does not care about volume preservation or suffer from gravity.

The control polygon of the embedding space is now a 3D lattice, which is parallelepipedical in its simplest form. The process of deforming a shape consists of the following steps:

1. Cartesian coordinates of the model are mapped to coordinates in the parametric space of the embedding tri-variate spline volume, or hyper patch.
2. The shape of the embedding volume is changed by manipulation of the control points in the lattice.
3. The spline volume is evaluated at the parametric coordinates determined in step 1 to give new Cartesian coordinates of the deformed model.

Naturally, the model does not need to be embedded completely in the volume, although it will only be able to deform for the parts that are.

5.1.1 Arbitrarily Shaped Lattices

The kind of deformations that parallelepipedical lattices allow are limited, which inspired Coquillart [1990] to deform the lattice also prior to step 1 above. She proposes to assemble several tri-cubic Bézier volumes by constraining some of the external control points of adjacent volumes to each other, and also to allow them to be degenerate. Embedding volumes can even be created from surfaces in the same way as surfaces are defined from curves (loft, sweep, extrusion etc.). Thus a wide variety in the shape of embedding volumes is possible, and in the structure of the (possibly) composite lattice, and thereby in the possible deformations.

With parallelepipedical lattices, step 1 on the preceding page consists of a straight forward linear mapping. The trade-off imposed by Coquillart's approach is that no such mapping exists, and calculating the volume parameters that correspond to the Cartesian coordinates requires numerical iteration.

MacCracken and Joy [1996] propose to define arbitrarily shaped embedding volumes by means of recursive subdivision [Catmull and Clark, 1978] as discussed in Section 3.1.1, generalised to volumes. This does away with the constraints on lattice control points that Coquillart needs for geometric continuity. Instead of turning to numerical methods to establish the correspondence between Cartesian coordinates and positions in the deformable volume, it turns out that the subdivision procedure itself can be used for this task. Nevertheless, the process is costly compared to a linear mapping, which means that the number of object points that can be deformed interactively may be too small for the application at hand.

5.1.2 Direct Manipulation

For larger lattices, it can be difficult to see how the lattice control points are ordered. The lattice tends to clutter the screen and obscure the object being deformed, and some control points may be hidden within the object. Also, the bewildering degrees of freedom can make it hard to determine how the lattice must be manipulated to obtain a particular deformation in the model. Direct manipulation has been proposed as a solution to this problem, in which an intended deformation is indicated on the embedded model, which is then translated to shifts in the control points of the lattice.

Hsu *et al.* [1992] show how the lattice alteration can be calculated, based on an indicated shift of one or more object points. The system is usually under-determined, as there may be many deformations that contain the indicated shifts; but an over-determined system occurs when the lattice is too coarse to obtain the indicated deformation. They use a least-squares approach which performs well in both the under- and over-determined cases.

Terzopoulos and Qin [1994, section 7.7] suggest a different approach, of embedding the model in a tri-variate NURBS volume that is extended with physical properties; the so-called dynamic NURBS (d-nurbs) described in Section 5.5.2. This volume will deform in accordance with applied forces, which may act on any point within the volume.

Gain [2000, chapter 4] extends the direct manipulation principle to include control of the tangent plane, twist and scaling in addition the position of an object point. As a further improvement of the usability of ffd, Gain [2000, chapter 6] proposes to control the lattice by means of curve manipulation, through a process of curve sampling or functional composition and degree reduction. Given a source curve contained in the deformation volume, the idea is to compute how its lattice should be deformed so that the curve approximates the shape of a given target curve.

5.1.3 Impact on Model Representation

One thing that is often said about free-form deformation is that the method is independent of the representation of the model that is being deformed. Although that is true, it does not mean that the representation is not of concern. Especially in animation, where ffd is frequently used, polygonal model representations are popular. The reason is that the data density of polygonal models is usually high in comparison to the density of control points in the lattice, so the tri-variate spline volume is sampled dense enough to transfer the smoothness in the deformation to the model — in the general case.

But it is not unusual that regions with little curvature in the original model, where polygons may be larger, have a much higher curvature after the deformation, and actually need a lot more smaller polygons in order to give a smooth appearance. Therefore, refinement of the polygon mesh may be necessary prior to deformation. The reverse can also be true, leading to polygon saturation in the deformed model. Gain [2000] gives methods for mesh refinement and decimation to address both these issues.

When the model is represented with spline curves and surfaces, as is the case in most engineering applications, data density is a lot thinner. At first one may think that it suffices to deform the control polygons and nets of the curves and surfaces in the model. But although that will give *a* deformation, it is not *the* deformation described by the embedding volume. In particular, this practice does not preserve continuity conditions that were present in the model.

One solution is to keep both the original model and the deformed lattice, and evaluate both the original representation and the deformation for every model point that is needed. Besides the extra toll in size and evaluation time, this is not friendly towards repeated application of the procedure. But the biggest problem is that all algorithms for rendering, intersection calculations, derivatives, fairness interrogation, and so forth, must be adapted to evaluate multiple nested structures. And they need to take into account the sampling aspect indicated above.

A more realistic approach is to embed the curve and surface definitions in the volume definition by tri-variate composition [Bézier, 1978; Gain, 2000]. The problem there is that this increases the polynomial degree beyond every practical value; if a curve of degree p is embedded in a tri-variate spline of degree $l \times m \times n$, tri-variate composition yields a curve of degree $p(l + m + n)$. Bézier [1978] reports degrees of 75 to 150. Using tri-variate B-splines for the embedding volume can bring down that number significantly, but not within the comfort range. Higher degrees give rise to ill-conditioning and increased computation burden [Gain, 2000]. In addition, unless all needed deformations can be performed with the same lattice, the increase in degree makes the process unfit to be applied repeatedly. It may be possible to reduce the degree after the deformation, but that is an approximative procedure which introduces a deviation, and explicit care must be taken that continuity is not affected.

Tri-variate composition can also be used on polygonal representations, as Feng *et al.* [1998] propose, as an alternative to mesh refinement. Their method takes a polygonal model as input and yields a curved model described by triangular Bézier patches as output — which eliminates the sample problem of polygonal models under FFD.

5.2 Hierarchical Refinement

Hierarchical B-spline refinement [Forsey and Bartels, 1988] was briefly demonstrated before, on page 31. This principle builds on the understanding that knot refinement, which increases the number of control points in a curve or surface, is an exact operation. That is, the surface is identical before and after the refinement. As we know, ordinary refinement of a surface introduces complete rows and columns of control points, which can be a problem, as discussed in Section 2.2.1.3. With hierarchical refinement, the original surface definition is kept, and the refined surface is only used for the quadrilateral region where the refinement was intentional. This smaller patch with finer control is used in place of the original surface where they overlap, and elsewhere, the original surface is used. In order to conserve consistency between the original surface and the refined surface, only internal control points of the refined patch may be moved.

The control points of the refined surface are expressed relative to the positions and surface normals of points on the original surface, corresponding to the surface parameters at which the basis functions of these control points have their maximum. This way, manipulations on the underlying surface carry over to the refined surface in a global sense.

Refinement may be repeated in a hierarchical manner, and the surface representations at different levels of detail can be stored in a tree structure. Composite surfaces of this kind have a highly complicated structure of control points. As the user cannot be expected to make sense of the maze of the control graph, Forsey and Bartels [1988] propose a simplified form of direct manipulation. Every control point corresponds to a point on the surface over which it has maximal influence, which is positioned by the parameter values for which its basis function has a maximum. A change in such a surface point can be converted to a change in the one corresponding control point. So the user is given to pick any such surface point at the level in the hierarchy that corresponds to the intended domain of the manipulation, and drag it to change the shape of the model accordingly.

5.2.1 Localised Hierarchy Surface Splines

By building on the tensor-product B-spline basis, the above method inherits the limitation, regarding the modelling with arbitrary topology, of not addressing the problem of smoothly joining more or less than four patches at a patch corner. This led Gonzalez-Ochoa and Peters [1999] to apply the hierarchical refinement idea to C^1 surface splines [Peters, 1995b], which do address the matter of arbitrary topology, as discussed on page 43. Their system of localised hierarchy surface splines (LeSS) allows connecting arbitrary sub-meshes by which a change of genus can be accomplished. This gives complete freedom of modelling in a topological sense, like the ability to create bridges and punch holes.

Whereas models using hierarchical B-spline refinement [Forsey and Bartels, 1988] are defined as surface patches offset from ancestor patches, in LeSS, finer-level patches *replace* coarser-level patches. So the evaluation of the spline patches does not require traversal of the hierarchy and is numerically stable.

Refinement happens at the mesh, as a partial Doo-Sabin-like subdivision. The nodes of the refined mesh fragment are expressed as offsets of coarser-level nodes, which forms the mechanism by which surface details follow editing on coarser levels. The surface itself is represented by small cubic triangular Bézier patches. Optionally, four of these patches can be converted into one linearly trimmed (diamond-shape) bi-cubic NURBS patch. Even though surface patches are not overlaid as in [Forsey and Bartels, 1988], all levels of refinement in the mesh remain present and active, so the structure of control points becomes just as complex. Therefore, LeSS also supports direct manipulation.

5.3 Superposition

When modelling with plasticine and confronted with the task of adding a bulb to some sculpted model, one approach is to form the bulb separately and stick it to the model. It would be natural to investigate an analogy to this in CAGD, which can be formulated as superimposing a displacement function on a curve or surface.

This is the approach taken by Ishida [1997], with the motivation to provide direct manipulation of the global shape of curves and surfaces. His method is developed on curves, but does not extend quite so well to surfaces. Given some constraints, such as a prescribed shift of a point on the curve, some fixed points and some prescribed tangents, a displacement function is constructed by means of a general B-spline interpolation method. The parameterisation of the displacement function is compatible with the original curve, so that superposition into a new curve consists of the following steps:

1. Adjust the degree of the original curve and the displacement function to the higher of the two degrees, by degree elevation.
2. Calculate the union of the knot vectors and represent the curves over the unified knot vector, by knot insertion.
3. Add the resulting control points of the original curve and the displacement function, by vector summation.

As a consequence, the result may end up with a non-uniform knot vector and additional control points. This is unfortunate, as repeated application of this procedure will pollute the curve with superfluous data.

For surfaces, constraints can be given in the form of points or curves. The limitation is that these points and curves must line up with the control point grid and iso-parametric lines respectively. This requirement may only be an inconvenience as long as we are dealing with a single surface patch or a regular checkerboard arrangement of patches. However, when dealing with models with arbitrary topology, this requirement cannot be met and Ishida's approach cannot be used for global shape manipulation.

5.4 Decay Functions

A polygonal mesh is a low level geometric representation, to which higher level representations such as parametric surfaces are frequently converted for fast computer rendering. Apart from FFD, discussed in Section 5.1, other methods exist to manipulate these polygonal representations directly. As the data density in a polygonal representation is high, the need for global modification quickly becomes apparent.

Our first reference dates back to 1977, when Parent published a description of an animation system implemented on a PDP-11. Parent [1977] allows pulling on one point in the mesh and proposes an interpolation routine that pulls neighbouring points proportionally. The neighbouring is based on the mesh connectivity, namely the adjacency counted by the number of edges which must be traversed from the point in question to the point that is actively being pulled at. This involves only integer arithmetics, which was probably an important performance question at the time.

The proposed scaling of point movements is based on a simple polynomial formulation of integer order $k \in \mathbb{Z}$. Let n be a specified maximum adjacency count, beyond which points are unaffected by the pull, and let i be the adjacency count for the point for which a pull needs to be computed. When $\mathbf{d}$ is the movement of the point that is actively being pulled at, then the movement of adjacent points $\mathbf{d}_i \equiv f\mathbf{d}$, where the *decay function* f is defined as

$$f \equiv \begin{cases} 1 - \left(\frac{i}{n+1}\right)^k & \text{for } k > 0 \\ 1 - \frac{i}{n+1} & \text{for } k = 0 \\ \left(1 - \frac{i}{n+1}\right)^{-k} & \text{for } k < 0. \end{cases} \tag{5.1}$$

The shape of this function, plotted in Figure 5.1, is linear whenever $k \in \{1, 0, -1\}$, semi concave when $k > 1$ and semi convex when $k < -1$.

The same approach is taken by Allan *et al.* [1989]. Having the advantage of better hardware, they propose to take the Euclidean distance as the input of the decay function, for a more intuitive deformation. With this, the method becomes a spatial deformation method and can operate on any model representation, like the FFD. Allan *et al.* [1989] also give the option of expanding and contracting an object, by moving points in individual directions calculated as the average of the normals to their incident faces.

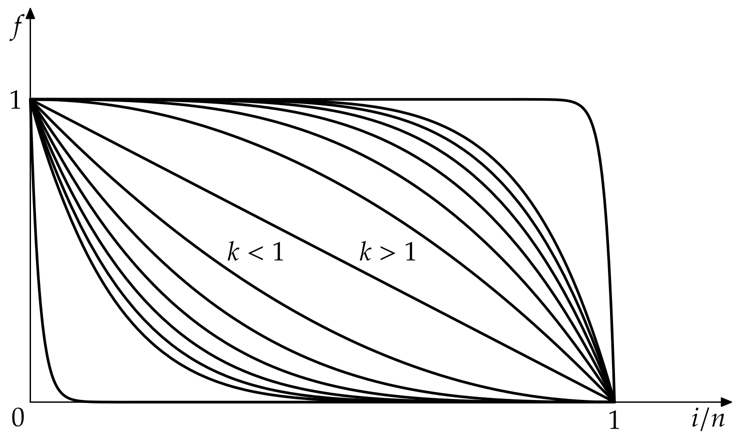


Figure 5.1: Decay functions defined by Parent [1977], see (5.1) on the preceding page, displayed for $-7 \leq k \leq 7$, as well as $k = \pm 64$, which they propose as a maximum.

Obviously, Allan *et al.* have great fun proposing various decay functions. They give the choice between

1. a constant value of 1.0 — which probably makes more sense than taking the power of a polynomial as high as you can, as Parent does above,
2. a linear function, producing a cone, like $k = 0$ in Figure 5.1,
3. a quadratic function, producing a cusp, like $k = -2$ in Figure 5.1,
4. a bell-shaped function, produced by a cosine over the interval $[0, \frac{\pi}{2}]$, see Figure 5.2 on the following page,
5. a wave effect, produced by a sinusoid, and
6. noise, produced by a randomising function.

Bill [1994] combines the decay function approach with adaptive mesh refinement and smoothing to build a system for sculpting polygonal models with virtual tools.

The next contribution is by Borrel and Rappoport [1994], who use a B-spline basis function as the decay function*. Actually, they

*According to a reference in [Marsan *et al.*, 2001], this was proposed already in 1991 themselves do not refer to the works presented in this subsection so far, and see their work as a continuation of the research on FFD. Their objective is to formulate an efficient directly manipulated spatial deformation, and in the process they loose the hyper patch and the lattice altogether. As a consequence, their approach may be argued to have more in common with the approaches centred around decay functions, although the borders begin to fade.

Previously, the cosine was the only proposed decay function that results in a smooth deformation. It is interesting to note that there is one particular basis function that resembles the cosine rather well, and when written in power form ($f = 1 + d^2(2d - 3)$) evaluates five to six times faster than the cosine ($f = 1 + \cos(d\pi)$), according to the relative costs for floating point operations as given by [Gain, 2000]*. Figure 5.2 below shows these two functions in one plot.

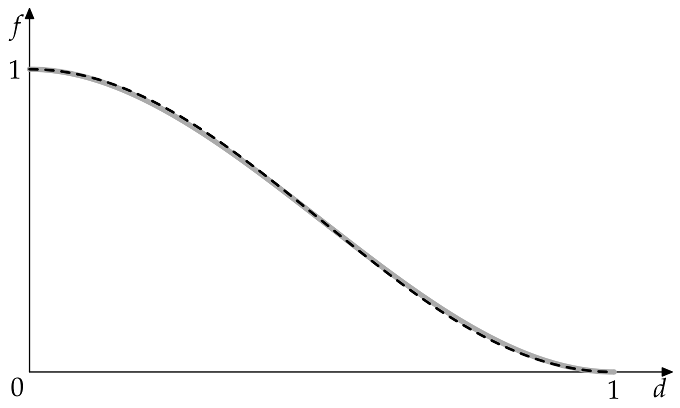


Figure 5.2: Comparison between a cosine over half its interval ($f = 1+\cos(d\pi)$, dashed) and B-spline basis function ($f = 1 + d^2(2d - 3)$, grey).

An advantage of B-spline basis functions is that their shape can be varied by changing their knot vector, as in Figure 6.9 on page 109. However, the general basis functions involve a division, by which they loose their edge in efficiency to the cosine.

One additional variation on decay functions is suggested, namely to keep the function value at unity for some time, before starting

by Paul J. Stewart in his PhD thesis *Direct Shape Control of Free-Form Curves and Surfaces with Generalized Basis Functions* (The University of Michigan, Ann Arbor). However, this could not be checked due to unavailability.

*Addition, subtraction: 1.0; multiplication: 1.614; division: 16.068; square root: 31.094; sine, cosine: 26.687; measured on an SGI Octane 195 MHz R10000.

the smooth decay towards zero. This has the effect of offsetting the space in a particular region, while providing a smooth transition to the undeformed space.

Borrel and Rappoport describe their deformation approach as follows:

> "The user defines a set of constraint points, giving a desired displacement and radius of influence for each. Each constraint point determines a local B-spline basis function centred at the constraint point, falling to zero for points beyond the radius. The displacement of a point is a blend of these basis functions, obtained by a linear combination that insures that all constraints are satisfied." [Borrel and Rappoport, 1994]

Care is taken to satisfy all constraints simultaneously and exactly, for which a system of equations needs to be solved. This happens as follows. When fields of influence are disjoint (Figure 5.3(a) on the next page), constraints do not interact and points are only displaced based on their distance to a single constraint point and its displacement*. When fields of influence partly overlap but their constraint points are still separated further apart than the size of their radii (Figure 5.3(b)), points in the overlapping region are displaced by the vector sum of the displacements obtained as if each constraint was acting alone. But if the field of influence of one constraint point includes an other (Figure 5.3(c)), just taking the vector sum as before would overshoot the constraint displacement of the contained constraint point. In this case, the solution of the system of equations will have produced a different displacement vector for the contained constraint point, one that compensates for the influence of the neighbouring constraint, by which all constraint displacements are met.

This procedure can lead to unexpected results. When two constraint points are placed closely together, with their fields of influence almost coinciding (Figure 5.3(d)) but with constraint displacement vectors that point in completely different directions, the solution of the constraint system may produce displacement vectors that are much longer than the original constraints. This will produce a spatial defor-

*For the sake of consistency with Borrel and Rappoport's text, we will stick with the term "displacement" for the moment. It is not the preferred term in this thesis, because like "deformation" it suggests a degradation in quality. In some other texts, the verb "to move" is used, which is not any better because there is no motion involved; the change in position is instantaneous. In later chapters I will use the term "shift" instead of these other terms, to denote a change in position.

mation that is not well behaved in which displacement vectors can vary violently, a phenomenon called *space tearing*.

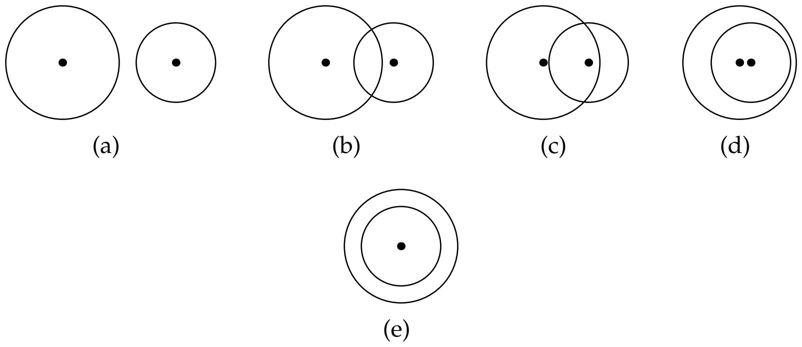


Figure 5.3: Interference between fields of influence.

When two constraint points coincide exactly, the system of equations is over-determined. This is actually allowed, and the least squares solution to the system may be computed by means of a pseudo-inverse. Naturally, the result will not meet the conflicting constraints, but they will be optimally approximated. The authors find that duplicated constraints with different radii of influence (Figure 5.3(e)) can be used to reduce space tearing.

Lazarus *et al.* [1994] propose axial deformations for modelling and animation, in which object points follow changes in a curve defined on or near the object. For this, the shortest distance of an object point to the curve is considered, and its position relative to an orthogonal frame defined at the closest point on the curve. A minimum and maximum range of influence is defined along the curve. Object points outside the maximum range are unaffected. Points inside the minimum range are displaced according to their position in the frame that has been transformed by a change in the curve, which may include rotation. The frame is positioned on the curve at the same curve parameter value as before. For points in between the minimum and maximum range, the displacement is weighted.

Singh and Fiume [1998] improve the interface of axial deformations by replacing the coordinate frame with explicit control over scaling, rotation and translation of data points. They have special interest in the interaction between several deforming curves (which they call a wires), and propose to prevent superposition of deformation by averaging the influence of the deformations that are locally active.

They also propose domain curves to demarcate the deformation for anisotropic directional control of the deformation, and they give a heuristic by which the domain curve can be used to control the extent of the deforming area on one "side" of the wire, relative to the surface. This however involves computing one more shortest distance, which is an expensive operation for curves. Singh and Fiume [1998] use a C^1 polynomial decay function, plotted in Figure 5.4 below.

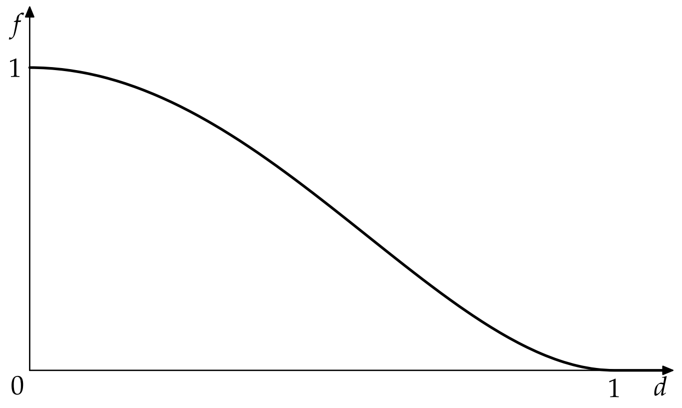


Figure 5.4: C^1 decay function proposed by Singh and Fiume [1998] ($f = \{(d^2-1)^2 \mid 0 < d \leq 1, f = 0 \mid 1 < d\}$).

Basis functions are also used by Marsan *et al.* [2001], who focus on surface feature design. Their contribution is in the input of the basis function; instead of taking the Euclidean distance between points (which they call "radial parameterisation") or the topological distance of a polygonal mesh, they propose a parameterisation modelled after heat flow in a plate, which they call Dirichlet parameterisation. This involves solving a two-dimensional Laplace equation, or potential equation, which they do numerically, using the finite element method (FEM). There will be more on FEM in Section 5.5.1 on the next page.

Marsan *et al.* [2001] allow the domain of the feature to be marked on the surface by a closed periodic NURBS curve. One or more influence centres, at which the surface feature will have a maximum, are marked inside the domain, as points, open curves, or closed curves. Then the surface, together with the markings, is mapped on to a plane, e.g. by parallel projection, and the domain is meshed into triangular elements. A linear system of equations is set up, consisting of a conduction matrix for the element nodes, boundary conditions and the unknown parameter values that will be the input for the basis function. The system is solved for these parameter values by means of basic linear algebra, and the values are then mapped back onto the surface, where a basis function produces a scaling factor for a displacement vector as usual. In-between the nodes of the finite elements, displacement is obtained by linear interpolation.

The advantage of this approach is the freedom and the control in the definition of the domain of the feature, which may be non-convex, and the shape and number of influence centres. However, the examples presented by Marsan *et al.* [2001] all use very fine meshes, and one may wonder whether the use of the topological distance to influence borders and centres would not yield acceptable results as well. This involves an exhaustive graph search, but it still may perform faster than doing a full blown finite element analysis (FEA).

5.5 Deformation Based on Physics

So far we have seen purely geometric methods for global shape manipulation (although the heat transfer analogy from above was inspired by physics, it is still a geometric deformation method). In an attempt to achieve a more intuitive method and realistic sculpting functionality, it has been tried to simulate the behaviour of real materials by integrating structural analysis according to the laws of physics.

5.5.1 Finite Element Method

In general, the finite element method (FEM) is a method for solving an equation by approximating continuous quantities as a set of quantities ("finite elements") at discrete points ("nodes"). Because finite element methods can be adapted to problems of great complexity and unusual geometry, they are an extremely powerful tool in the solution of important problems in heat transfer, fluid mechanics and mechanical systems, which are intractable using analytical methods.

The application of FEM to shape manipulation is mainly inspired by its use in mechanical engineering, where it is primarily applied for checking the stresses in the material of a given structure under a given load. An FEA of this kind results in the stresses in the elements and displacements of the nodes — the former follows from the latter. In mechanical engineering, the deformation is typically of small magnitude and usually regarded as by-product of the analysis. When applied to shape manipulation however, the only interest is in deformation, i.e., the change in the position of the nodes.

It goes beyond the scope of this thesis to go into any depth regarding the mathematics of FEM. Besides a survey of the application of the method to deformable modelling, Gibson and Mirtich [1997] give a good introduction to FEM, as well as references to more detailed coverage.

One of the important things to remember from a mechanical engineering course on FEM is that the method assumes small deformations. In deformable modelling, that assumption is not going to hold. If one is after physical correctness, an expensive re-evaluation of the force vectors and mass and stiffness matrices is required as the object deforms, at sufficiently short time intervals so the assumption of small deformations is valid. Therefore, the method is not a natural fit for interactive global shape manipulation, and applications are sparse.

Celniker and Gossard [1991] describe special purpose elements for smooth curve and surface modelling. They propose a three-step algorithm for the design of free-form shapes. Firstly, curves are set up where the surface is going to have a tangential discontinuity, such as at edges and along creases. The object is then skinned with a deformable surface, consisting of triangular surface elements that minimise both curvature and surface area. The result is comparable with the membranes that span the openings in a physical wire-frame that has been dipped in a soap solution. In the third step, these surfaces (and the curves) can be deformed by applying external forces such as point loads and distributed pressure. The smooth surface elements connect C^1, and are able to describe shapes with arbitrary topology. The shape can be constrained with prescribed position and surface normal at a point on the surface, or along a curve, be it an edge or internal within the surface.

Mandal *et al.* [2000] present a unified approach to represent the smooth limit surface of subdivision surface schemes, using a collection of a single type of finite elements. The method allows direct surface manipulation by means of the application of synthesised forces, is however computationally expensive.

McDonnell and Qin [2001, 2004] apply FEM to subdivision solids, the generalisation of subdivision surfaces to solids that MacCracken and Joy [1996] proposed for FFD lattices of arbitrary topology. For complex models, the FEM-based method does not perform in real time. Alternatively, accuracy is traded for efficiency [McDonnell *et al.*, 2001; McDonnell and Qin, 2002] by simplifying the dynamic model to a mass-spring lattice. The result is an interactive haptics-based deformable modelling technique [see also McDonnell, 2003].

To alleviate the computational burden of an FEA, in order to obtain interactive responsiveness in FEM-based shape modelling, Kang and Kak [1996] propose to do the analysis at two resolutions. Initially, a coarse volumetric analysis is performed to calculate gross deformations at a set of points in the object. The result is then used as the boundary condition for a more detailed analysis of the object surface using plate elements. Again, accuracy and physical correctness are thus traded for efficiency.

Other applications of FEM focus mainly on the simulation of human tissue, for applications where high realism is a requirement such as in surgical simulation and for animation in the entertainment industry.

5.5.2 Low Degree of Freedom Models

Gibson and Mirtich [1997] review several low degree of freedom models. As opposed to the FEM approach, these models sacrifice physical generality for speed. Among these, minimal energy surfaces are the most interesting to us, since they add physical behaviour to traditional geometric modelling primitives, particularly parametric surface patches.

Celniker and Welch [1992] present deformable B-spline tensor product surfaces with linear constraints. Their method preserves a set of geometric constraints, such as interpolated points and curves and prescribed surface normals, while interactively sculpting a free-form B-spline surface by means of virtual forces. The surface seeks a fair shape by minimising an appropriate global energy function, based on resistance to stretching and bending. Thus the surface area and surface curvature are minimised. Welch and Witkin [1992] extended this to trimmed hierarchical B-splines [Forsey and Bartels, 1988], discussed on page 31, to reduce the error bound. This results in surfaces that appear infinitely malleable, by defining points and curves that the surface interpolates. Zheng and Zhang [2002] use linear constraints [Celniker and Welch, 1992] on models of arbitrary topology, using the non-quadrilateral patches introduced by Zheng and Ball [1997], as discussed on page 41.

The extension from deformable B-splines to NURBSs is provided by Terzopoulos and Qin [1994], resulting in so-called dynamic NURBS (D-NURBS) [see also Qin and Terzopoulos, 1996]. Continuous functions distribute mass, stiffness and damping throughout the NURBS surface. The dynamic behaviour of these surfaces results from the numerical integration of a set of non-linear differential equations that automatically yield the control points and weights in response to applied forces and constraints. To derive these equations, Lagrangian mechanics and a "finite element"-like discretisation is employed. The D-NURBS surfaces are evaluated at small time increments, providing visual feedback of the evolving state of the dynamic model. The method transfers to triangular D-NURBS surfaces [Qin and Terzopoulos, 1997], which facilitate the modelling of arbitrary topology, as well as to hierarchical D-NURBS surfaces Zhang and Qin [2001], which facilitate modelling at several levels of detail.

The existence of weights makes the incorporation of dynamics in NURBS geometry substantially more challenging than doing the same in B-spline geometry. Since the NURBS rational basis functions are functionally dependent on the weights, D-NURBS dynamics are generally non-linear, and the mass, damping and stiffness matrices must be recomputed at each simulation time step. Ironically, Terzopoulos and Qin [1994] report that the weights tend to move toward zero (reducing the influence of the corresponding control point accordingly) due to the minimisation of surface energy. We note that weight editing is rarely applied as a shape parameter by designers; weights are usually computed only to represent conic sections accurately. Therefore one might question the value of including these degrees of freedom in a framework for deformable surfaces, especially because this makes the analysis an order of magnitude more costly [Terzopoulos and Qin, 1994]. By making the weights constant, the surface reduces to a dynamic B-spline.

Guan *et al.* [1997] apply physics-based deformable curves and surfaces to the construction of *n*-sided surfaces, smooth surface joining and surface fairing.

5.5.3 Spring-Force Models

Dachille IX *et al.* [2001] describe a haptics-based interface to dynamic sculpting of a single B-spline patch, with force-feedback. They derive a mechanical model by a discretisation of the patch into a grid at a user-defined resolution. In-between the grid points, spring forces are defined, and mass, damping and stiffness properties can be "painted" on the surface by spray-box analogy. Constraints can be defined on position, tangent and curvature. The dynamic system is solved by means of the finite difference method.

Thingvold and Cohen [1990] propose to use elasto-plastic mass-spring-hinge models on the B-spline control points for interactive design and animation.

Léon and Trompette [1995] achieve a very efficient system by considering the static equilibrium of a force-spring system over the control polyhedron of a B-spline surface. Given the initial state of the control polyhedron of a B-spline surface patch, spring elements are defined between the control points, with a connectivity that mirrors the topology of the control polyhedron. These elements are of equal rest length and are assigned a certain spring constant (the authors consider the ratio between the element length and the internal tension directly, which they call "force density"). External forces are defined, acting on unconstrained B-spline control points, such that the static equilibrium between the forces on the control points and the tension in the spring elements resembles the initial state of the control polyhedron. The computation of the external forces involves a straight forward solution of a linear system of equations.

Given this initial equilibrium, the surface can be stretched, inflated, tweaked, etc., by changing external forces and/or spring constants. For this the computation is reversed, solving for the control point positions that correspond to the new static equilibrium of the changed mechanical system.

Boundary conditions, i.e., the specification of free and constrained control points, can be specified either directly, or indirectly by indicating an area on the surface and considering control points that control the shape in that area. The authors report interactive modelling performance for up to seven or eight hundred free control points on a common work station. In contrast to dynamic systems, this is a static method (mass, damping, acceleration and velocity are unconsidered) so the change in shape is instantaneous. Before the designer arrives at the intended shape, several alternative load cases may have to be considered.

Whereas Welch and Witkin [1992] integrate an energy functional over a deformable surface to make it interpolate points and curves, Guillet and Léon [1998] have the same objective but use the simpler mechanical approach of Léon and Trompette [1995]. They allow the surface area under consideration to consist of several connected B-spline surface patches, and G^1 patch transitions are preserved by constraining relative positions of control points. The method enables a composite surface adapt to interpolate a given point and tangent plane, and allows large-scale deformations.

5.6 Chapter Summary and Look Ahead

With the conclusion of this chapter, we have provided us with an overview of the solutions proposed in the literature, to reduce the degrees of freedom for the process of forming the shape of surfaces. In general, the proposals either apply to polygonal discretisations or to parametric surfaces. In other words, there is a bias towards the approximation paradigm. Some consider only one single surface patch, others are capable of handling assemblies of a large number of patches. Some invent a non-standard surface patch, others can readily be superimposed on industry standard surface representations.

We recognise that the interpolation paradigm is under-represented in the context of shape variation (or deformation), and it is the focus of the following chapter to correct that.

CHAPTER 6

Manipulation of Shapes in H-rep

In this chapter we will address the problem that a high density of data in the surface representation limits the freedom of the designer to apply larger-area changes in the shape. Although this phenomenon exists in the approximation paradigm (Section 2.2.3, page 27) as well as in the interpolation paradigm (Section 4.6, page 68), the problem is more severe in the latter because the surface representation may be invalidated as a consequence. Our focus is therefore on the interpolation paradigm, in particular on the first two research questions as posed in Section 1.2.1.

The H-rep concept is heading the evolution of the interpolation paradigm, and consequently we will focus our discussion on Koelman's implementation. Courtesy collaboration with SARC in the Netherlands, proofs of concepts could be implemented in their commercial system for ship hull shape design, called *Fairway* [Koelman, 2004], allowing for experiments and testing of the proposed method. This has been a time-consuming process due to the size and complexity of the programme [Koelman, 1999].

Excerpts of this chapter appeared in Ship Technology Research [Veelo, 2004a], which is a scientific journal with a review policy (reprinted in Appendix B). In addition, presentations were given at two international conferences, namely the 3rd International EuroConference on Computer Applications and Information Technology in the Maritime

Industries, or COMPIT'04 [Veelo, 2004b], and the 8th International Design Conference, or DESIGN 2004 [Veelo, 2004c].

6.1 Problem Statement

The problem is, given a certain region on the surface that interpolates a network of curves, to manipulate all curves in that region simultaneously, in a way that does not destroy the consistency of the network and does not introduce unwanted geometric discontinuities.

6.1.1 Manipulation of Control Points

Initially, one might reason that the surface is defined by the curves in the network, and that the curves are defined by their control points, and thus to manipulate the surface one is to manipulate the control points.

But the control points of the individual curves in the network lay in a thick cloud around it, and the position of each control point inside that cloud has been carefully determined in order to make the curves intersect each other. There is no rule that describes their correlation, they are just suspended in air in a delicate balance that was built up in small incremental steps. How this cloud can be transformed in one operation without breaking that balance is not well understood.

6.1.2 Manipulation of Data Points

As concluded in the previous section, instant generation of a surface variation by manipulation of curve control points will be problematic. A better handle on the surface is needed.

We note that the control points of the curves were determined automatically by the fitting/fairing algorithm described in Section 4.5.6, based on data points. Curves that, according to the topological information, intersect each other*, share a data point. These data points are persistent, as they are used by the fitting/fairing algorithm to restore network consistency when a curve was pulled away from the curves that it is supposed to intersect. As a consequence, data points are distributed over the entire surface.

If we take a fair and consistent H-rep as point of departure, then intersecting curves correlate through their shared data point. And

*If the network is consistent as a surface representation, curves intersect in practice; i.e., topologically as well as geometrically. If the network is (temporarily) inconsistent, they may cross.

curves can be regenerated from the data points from which they originated, courtesy the fitting/fairing algorithm. So with data points we have a direct handle on the surface, and by manipulating them we can manipulate the surface.

This is true as long as the number of data points on a curve is sufficient to hint the shape of the curve. This may become a problem when a curve is intensively manipulated in a terse network, probably by adding control points, without successively adding new intersecting curves by which extra data points are generated on the curve under consideration. One could try to refine the curve fairing/fitting algorithm to let it maintain existing surface features, such as the number of inflections. But apart from the theoretical challenges, there are practical ones: how can a fairing algorithm differentiate between wanted and unwanted inflections? Even the term *inflection* becomes less objective, as curves that were planar before, are likely to be pulled from their plane of definition and become space curves. They turn from two-dimensional (2D) curves into 3D curves, which makes it much harder to describe their features.

To rule out that surface features vanish after applying a surface variation, the following heuristic can help: check if the number of data points on a curve is somewhat in relation to the number of control points, and if they are sufficiently spaced. If they are not, additional data points can be inserted, to better hint the shape of the curve. This can be done completely automatic and invisible to the user, at negligible cost.

We can therefore rephrase the problem statement as

> Given a set of points on the surface of a consistent H-rep model, shift a contiguous selection of them in a way that when the surface is updated to interpolate these new positions, this does not damage surface details or fairness.

Remains to select a method that is able to accomplish this.

6.2 Prospective Methods for Shape Variation

We will now systematically revisit relevant methods for global shape manipulation that were discussed in Chapter 5, and select the one with the best prospects to see if it can be applied to the interpolation paradigm, and possibly developed further.

6.2.1 Free-Form Deformation

free-form deformation (Section 5.1) of the data points on the surface is a candidate. Because curves will be fitted anew through the updated positions of data points, the method will not affect curve continuity or degree, as otherwise discussed in Section 5.1.3. But the method requires the administration of a lattice, and although direct manipulation helps, it is not a very intuitive interface because the extent and character of the deformation depends of the size and orientation of the lattice. Arbitrarily shaped lattices may improve on the character of the deformation, but it adds an extra modelling burden on the designer. For a computer programme where the user is shielded for almost all mathematics behind the method, FFD is maybe not an optimal solution.

6.2.2 Hierarchical Refinement

Hierarchical refinement (Section 5.2) is a fix in the approximation paradigm that does not transfer to the interpolation paradigm. Even when applied to surface splines for modelling with arbitrary topology, the value for global manipulation is questionable. Levels of hierarchy need to be designed and there may well be manipulation requirements that were not foreseen in that design. It is mainly a feature for animation.

6.2.3 Superposition

Although superposition (Section 5.3) was developed on curves, the method considers one curve only, and not a network of mutually intersecting curves. The extension of the method to surfaces follows the approximation paradigm.

6.2.4 Decay Functions

Having neighbouring data points move proportionally when one of them is dragged to a new position, as accomplished with decay functions (Section 5.4), does seem to meet our requirements, and there is no lattice to be bothered with. This is a valuable candidate, and we will look into this further in later sections.

6.2.5 Deformation Based on Physics

There are three possible approaches to deform a set of data points by means of physical analysis. The first is a full FEA. This would involve meshing techniques and the application of plate elements, as stiffness against bending is essential for a deformation that has a smooth character. If the network cells are large, it may not be sufficient to define elements in between existing data points, and surface patches may have to be meshed at a finer resolution to arrive at a high enough accuracy. As an FEA is generally costly, it may not be possible to achieve interactive performance.

The second approach would be to incorporate physical properties in the transfinitely interpolating patches, in the style of Celniker and Gossard [1991]; Celniker and Welch [1992] and Terzopoulos and Qin [1994]. Whether this is at all possible is not certain. In the approximation paradigm, patches can be considered very much in isolation since they are controlled by an isolated set of control points. In the interpolation paradigm, a patch is controlled by its bounding curves, which may extend far beyond the patch. Each curve control point has influence on at least two patches (one on either side) but quite possibly more (if control points are spaced far apart in comparison to the number of intersecting curves). The correlation between surfaces and curve control points is thus anything but structured, and the problem is hardly comparable with the situation of the approximation paradigm.

In addition, the H-rep uses ordinary Coons patches as well as n-sided interpolating patches [Koelman, 1999]. This further complicates the matter. If a general formulation cannot be found, we would have to consider the case for every value of n separately. Even so, the literature only reports interactive modelling capability on rather small models. The approach may not scale well to the number of patches typically present in complete H-rep models.

The third approach may be to use a D-NURBS curve formulation [Terzopoulos and Qin, 1994] for the curves in the network. Although the curves would deform under the application of loads, there would be nothing to make them maintain their mutual intersections. These would have to be defined as constraints. However, these constraints are moving targets as the intersection points themselves are free to move (which is the objective) and may travel along the curve, i.e., are not fixed to one curve parameter value. Due to the non-linearity of the constraint problem and the number of intersections in a typical H-rep model, this is not a prospective approach for an interactive solution.

Spring-force formulations in between data points are not applicable as they will not guarantee geometric continuity. The elements will hinge on the constrained points, causing a sudden change in the tangent.

6.2.6 Making a Selection

From the above discussion, we conclude that only FFD and an approach based on decay functions do not pose obvious hindrances in their application to the interpolation paradigm. Due to its simplicity, the latter approach is selected as favourite candidate. In the following, we will investigate the value of decay functions for the manipulation of H-rep geometric models.

In the evaluation (Section 7.1) of the method that we will develop, we will compare its performance with the expected performance of FFD. Although the mathematics behind FFD is not complicated and in itself straight-forward to implement, the user interface of *Fairway* is not designed to handle 3D control lattices. To implement support for the interactive modelling with FFD in the graphical user interface (GUI) of *Fairway* is an undertaking so time consuming, that this was not pursued within this study. However, with a good understanding of spline theory [see e.g. Rogers and Adams, 1990; Foley *et al.*, 1990; Zeid, 1991; Piegl and Tiller, 1997; Farin, 2002b], the performance of FFD is predictable, which will have to justify a hypothetical evaluation and comparison with the method developed below.

6.3 Input to Decay Functions

As we have learned in Section 5.4, there is more than one input that we can compute decay values for. There is the topological distance, based on the number of edges that must be traversed from one node to another. This is a stepwise parameterisation and thus discrete. A continuous parameterisation results from taking the Euclidean distance, i.e., the length of a straight line through 3D space between two points. This does not consider the shape of the surface at all; for that, one should measure the distance over the surface. We have also seen the use of Dirichlet parameterisation that allows the specification of complex surface areas for variation.

Due to its discontinuous nature, topological distance will not be considered. The remaining types of input will be discussed subsequently in the following sections.

6.4 Euclidean Distance Dependant Variation

In this section we will explore the possibilities of decay functions, and start with an experiment considering the simplest possible case.

We will measure the proximity of data points as the Euclidean distance, i.e., in a straight line through space, disregarding the topology of the surface. We will shift data points in a common direction, but with an individual magnitude. We will make that magnitude dependent on the distance through space from the data point in question to a defined selection centre, and we will limit the variation to be effective within a specified radius from that centre. Thus, the field of influence of a shift operation can be seen as a sphere. We want the magnitude of the shift to be at a certain specified maximum at the centre of this sphere, then decrease smoothly further out, and eventually fade away at the boundary of the sphere. This decaying influence can be achieved by scaling the magnitude with a factor that is a function of the distance to the selection centre — a decay function.

The algorithm behind this experiment consists of the following sequence of operations:

1. Select a data point on the surface to function as the selection centre.
2. Compute the surface normal at that point, to function as the direction in which data points are going to be shifted.
3. Collect all data points in the model in a random access container with dynamic bounds*.
4. Sort all data points in this container according to their distance to the selection centre.
5. Acquire input from the user regarding the extent of the selection and the magnitude of the shift.
6. Starting with the data point closest to the centre of the selection, compute a scaling factor for every data point, by means of the decay function. Terminate when points fall outside the selection sphere, i.e., when the scaling factor becomes 0.
7. Display the resulting shift per data point, as a preview of the potential shape variation. Loop back to step 5 above until the user signals acceptance.
8. Move data points accordingly.
9. Re-fit affected curves.

The iteration over steps 5 through 7 can be executed sufficiently fast to support interactive operation. Interactive acquisition of the two

*This container was implemented for the occasion. See Appendix A.2 on page 170.

input parameters simultaneously is accomplished by binding them to movements of the mouse. Horizontal movements extend or decrease the selection area, vertical movements determine the magnitude of the variation.

6.4.1 Preliminary Definitions

Let us declare $\mathbf{s}_i$ to be the *shift vector* for a data point i, i.e., the difference between the position of that point after and before the shape modification. Also, let $d_{i,j}$ be the distance through space between data point i and the centre (or *base*) of a selection field j, as in Figure 6.1 below. At the base, we will explicitly define a shift vector, the so called *typical shift vector*, denoted by $\mathbf{S}_j$. The radius r_j of the selection sphere, or *extent* of the selection field, is also defined by the user. Outside this sphere, the shape will not be varied. The shift vector $\mathbf{s}_i$ can be expressed as a scaling of $\mathbf{S}_j$, according to a decay function f_j:

$$\mathbf{s}_i \equiv f_j \mathbf{S}_j. \tag{6.1}$$

We want the magnitude of the shift to be as specified at the selection centre, then decay as a function of the distance $d_{i,j}$ relative to the radius r_j, to zero outside the sphere. Thus $f(d_{i,j}/r_j)$ should return a value of 1 when $d_{i,j} = 0$ and 0 when $d_{i,j} \geq r_j$. To simplify its definition, we may normalise to r:

$$f(d_{i,j}/r_j) \equiv f(\delta) \quad \text{where } \delta \equiv d_{i,j}/r_j. \tag{6.2}$$

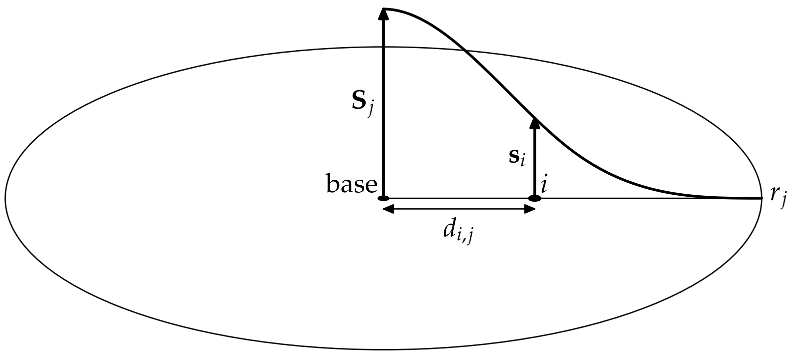


Figure 6.1: Schematic illustration of the computation of shift vectors.

Now f is defined on $[0, \infty]$, with a top of 1 at $d = \delta = 0$. It should start out horizontally for increasing d, then decrease smoothly and flat out at 0 when $d = r$, i.e., $\delta = 1$, where its support ends. In this experiment, its definition was based on half a basis function of a cubic periodic B-spline, which is G^2, scaled and translated according to the above requirements. Its definition, notated in power basis, is given in equation (6.3) and its plot in Figure 6.2 below.

$$f(\delta) \equiv \begin{cases} 6\delta^3 - 6\delta^2 \qquad + 1 & \text{if} \quad 0 \le \delta < 0.5 \\ -2\delta^3 + 6\delta^2 - 6\delta + 2 & \text{if} \quad 0.5 \le \delta < 1 \\ \qquad\qquad\qquad 0 & \text{if} \quad 1 \le \delta. \end{cases} \tag{6.3}$$

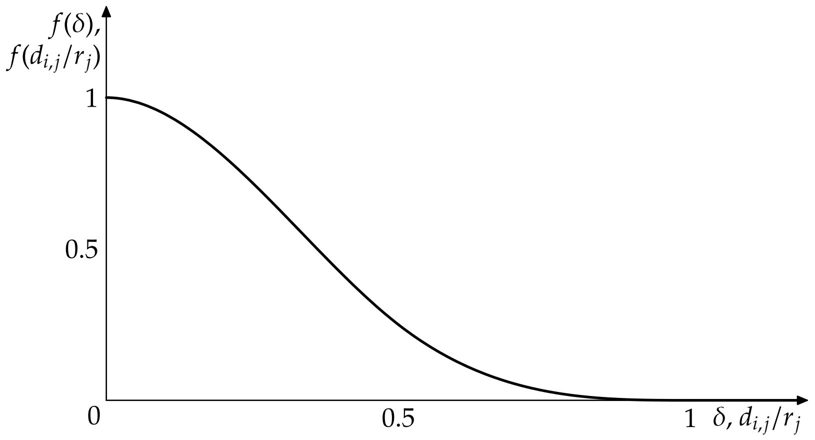


Figure 6.2: The plot of the G^2 piecewise polynomial decay function from equation (6.3) above.

6.4.2 Test Case

We will now evaluate the method on a planar surface, so that the effect of the operation can be easily observed. Figure 6.3 shows the initial shape of the H-rep. After the middle data point has been selected for the selection centre, bars dance out of the data points according to the movements of the mouse. As illustrated in Figure 6.4 on page 101, they indicate the shift that the data points are about to make. At the next press of the mouse button, the shape looks accordingly, as shown in Figure 6.5 on page 102 and Figure 6.6 on page 103.

As an additional evaluation of the produced shape, we can intersect it with a vertical longitudinal plane. We expect the resulting

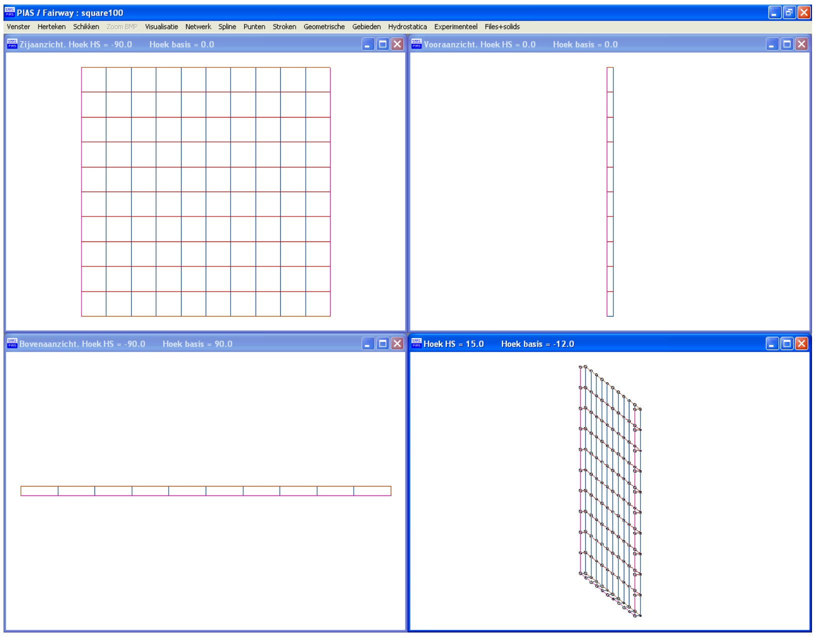

PIAS / Fairway : square100
Venster
Herteken
Schikken
Zoom BMP
Visualisatie
Netwerk
Spline
Punten
Stroken
Geometrische
Gebieden
Hydrostatica
Experimenteel
Files+solids
Zijaanzicht. Hoek HS = -90.0 Hoek basis = 0.0
Vooraanzicht. Hoek HS = 0.0 Hoek basis = 0.0
Bovenaanzicht. Hoek HS = -90.0 Hoek basis = 90.0
Hoek HS = 15.0 Hoek basis = -12.0

Figure 6.3: Screen shot of the graphical user interface (GUI) of *Fairway* showing the initial shape of the H-rep used in the test case. The upper left window contains the view from the side, the lower left window the top view and the upper right the front view. The lower right window displays an isometric projection. The model is actually a narrow box, of which only half is shown, just to provide a planar surface on which the effect of a surface variation is best observed. The lines in the centre plane of the box are printed pink, general vertical lines are blue and general horizontal lines red. The top of the box is omitted. The data points existing in the model are plotted in the iso-parametric projection. Note that their number is minimal: only at the intersection between curves are data points present.

intersection curve to be circular when viewed from the side, because the initial shape was planar and we used a spherical selection field. Indeed, that curve *looks* circular (Figure 6.7 on page 103), but a plot of its curvature shows small deviations and even a discontinuity, whereas a perfect circle has constant curvature.

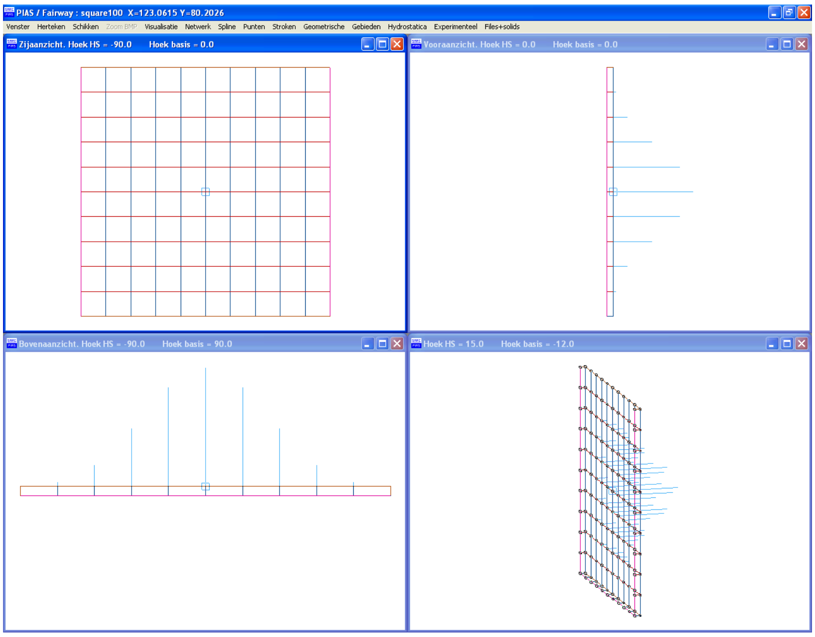

PIAS / Fairway : square100 X=123.0615 Y=80.2026
Venster Herteken Schikken Zoom BMP Visualisatie Netwerk Spline Punten Stroken Geometrische Gebieden Hydrostatica Experimenteel Files+solids
Zijaanzicht. Hoek HS = -90.0 Hoek basis = 0.0
Vooraanzicht. Hoek HS = 0.0 Hoek basis = 0.0
Bovenaanzicht. Hoek HS = -90.0 Hoek basis = 90.0
Hoek HS = 15.0 Hoek basis = -12.0

Figure 6.4: Preview of the potential shift of data points for the test case. The selection centre is indicated by a light blue square. Bars of the same colour extend from the data points, indicating their potential shift. They vary in real time according to the movements of the mouse.

These deviations must be expected in practice, because they are inherent to the modelling methodology. Apart from round-off errors, the following reasons can be given. Firstly, data points result from intersection with curves and surfaces that were *fitted* to discrete values. In between these, they represent a guess of finite quality, for which the deviation in curvature may be taken as an indicative measure. Secondly, the fitting/fairing algorithm supports non-rational curves only, which cannot represent circles perfectly*. Extending the algo-

*Only rational curves can describe conic sections. A full circle for example can be modelled with a quadratic NURBS curve with seven control points and specific weights.

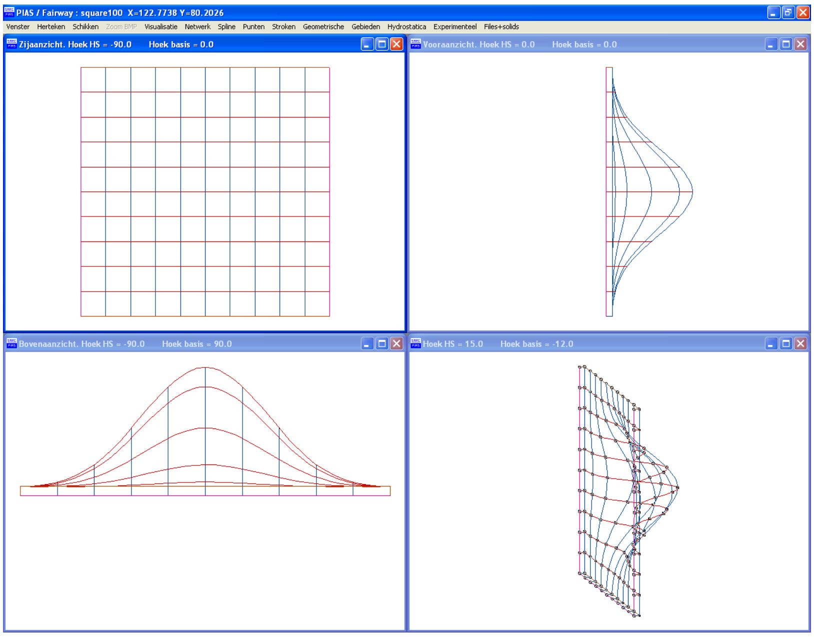

PIAS / Fairway : square100 X=122.7738 Y=80.2026
Venster Herteken Schikken Zoom BMP Visualisatie Netwerk Spline Punten Stroken Geometrische Gebieden Hydrostatica Experimenteel Files+solids
Zijaanzicht. Hoek HS = -90.0 Hoek basis = 0.0
Vooraanzicht. Hoek HS = 0.0 Hoek basis = 0.0
Bovenaanzicht. Hoek HS = -90.0 Hoek basis = 90.0
Hoek HS = 15.0 Hoek basis = -12.0

Figure 6.5: The shape of the H-rep from the test case after the operation is complete. When observed closely, the curves in the lower left window can be seen to have dropped slightly below their initial hight near their ends, which may be regarded as an imperfection. Obviously, such is the fairest curve through the given data points, as determined by the fitting/fairing algorithm; if the shape is not satisfactory, it means that the number of data points is insufficient (indeed, there could not be fewer in this example). One additional data point near either end of the curves would probably be enough to prevent this imperfection. Alternatively, we could have modelled the planar surface of this example a little wider, or have chosen the extent of the variation a little smaller.

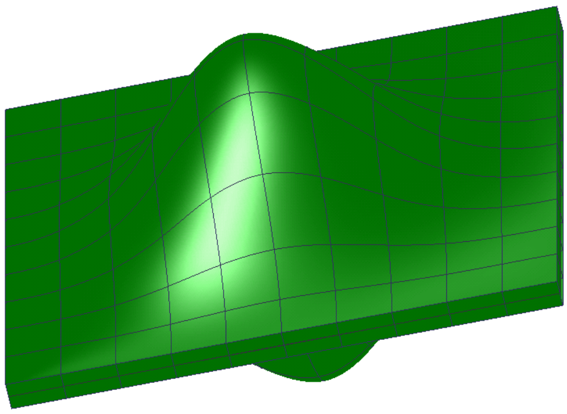


Figure 6.6: Rendering of the shape variation from the test case, including its mirror image.

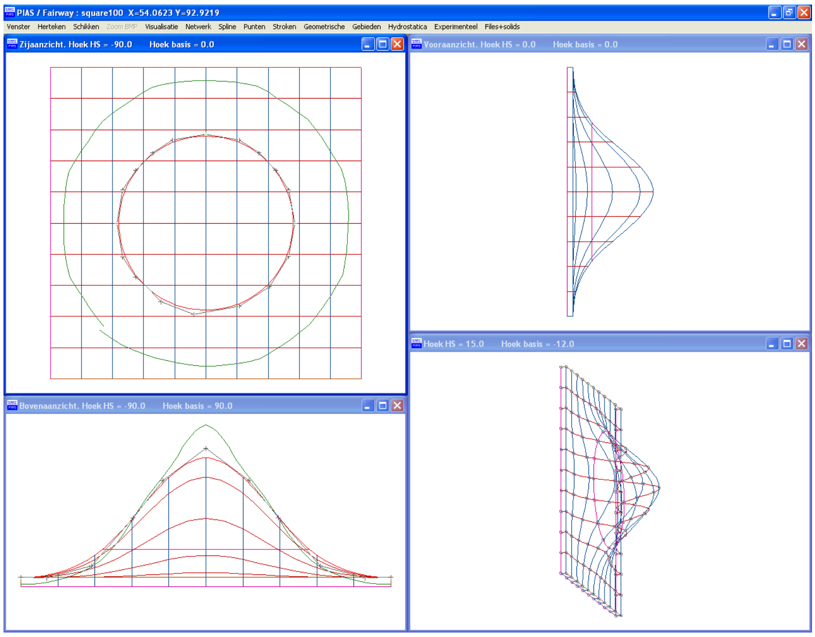

PIAS / Fairway : square100 X=54.0623 Y=92.9219
Venster Herteken Schikken Zoom BMP Visualisatie Netwerk Spline Punten Stroken Geometrische Gebieden Hydrostatica Experimenteel Files+solids
Zijaanzicht. Hoek HS = -90.0 Hoek basis = 0.0
Vooraanzicht. Hoek HS = 0.0 Hoek basis = 0.0
Hoek HS = 15.0 Hoek basis = -12.0
Bovenaanzicht. Hoek HS = -90.0 Hoek basis = 90.0

Figure 6.7: The result from the test case, intersected with a vertical longitudinal plane. The curvature of the resulting intersection curve is plotted (in green) in the upper left window, together with its control points (grey). The lower left window shows the curvature and the control points of a curve across the centre of the selection.

rithm to work in homogeneous coordinates might be possible, but the complexity would increase by one degree and efficiency would decrease accordingly. Even with such an extension it would be highly unlikely that the algorithm produces pure circles and circular arcs when they are theoretically applicable, and it may perform poorer in general cases.

The discontinuity in curvature however, may have been prevented by building more intelligence into the fitting/fairing algorithm, such that it can detect closed loops. For this it would be advantageous to extend the set of supported curve types with open curves, also known as periodic curves. Both issues are beyond the scope of this research.

6.4.3 Test Evaluation

The method presented thus far is primitive, and a number of shortcomings can be identified. Consider these, roughly in decreasing order of significance:

1. If the selection field crosses the plane of symmetry, there are no constraints that prevent the hull from splitting apart.
2. Only bump-like deformations can be pulled from a surface, and they all look alike. Especially if the selection field is not allowed to cross the plane of symmetry, the extent of the bumps are limited. Deformations of this kind are of little practical value to the designer.
3. The variation is in uniform direction.
4. All data points within the selection field are shifted, disregarding whether the selected regions of the network that they are embedded in are connected or not.

In the following sections we will try to improve on each of them.

6.4.4 Constraints

For designs that are required to be symmetric, such as ship hulls, it is advantageous to model only one half of the design and mirror it in the plane of symmetry. That is the approach taken in the implementation of Koelman, in which these experiments take place. The data points that are positioned on the plane of symmetry must not be allowed to shift away from that plane, otherwise the model will rupture. Clearly, that would fail the primary design objective for a ship hull, which is buoyancy. On the other hand, we want these data points to be free to move *in* the plane of symmetry. This special treatment must not introduce large differences in the shift magnitude and direction of data points on the centre plane and data points nearby. The transition between constrained and unconstrained data points must be continuous and smooth.

This particular case can easily be accomplished implicitly by mirroring the selection field as well as the model. Where the extent crosses the plane of symmetry, there will be two intersecting selection fields. These produce two shift vectors for each data point, of which the vector product can be taken. For data points in the symmetry plane, the components away from the symmetry plane are equal in length and opposite, and thus cancel each other out. This is probably the most efficient and straight forward way to prevent the model from rupturing over its centre plane.

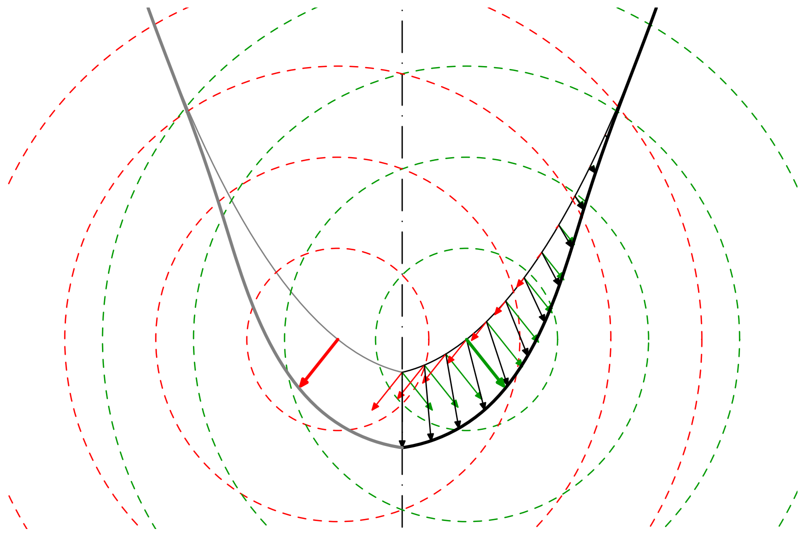


Figure 6.8: Symmetry plane constraint implemented by mirroring the selection field as well as the model. Green vectors originate from the selection field indicated by the green circles (extending up to the outer arc) and a typical shift vector indicated by the fat green arrow. The red vectors originate from the mirrored selection field indicated by red circles and a typical shift vector indicated by the fat red arrow, which is the image of the green typical shift vector. The shift vectors for the constrained variation, indicated by black arrows, are produced by the vector sum of the green and the red vectors. This figure was computer-generated with a decay function defined by equation (6.3) on page 99.

Constraints can also be defined more explicitly in a general way by treating the x, y and z coordinates[*] individually, and letting the magnitude of the shift for y coordinates decay faster, the closer points are to the plane of symmetry. For this a *de*selection field can be defined, emanating from the plane of symmetry, which scales down the effect of the selection field for y coordinates near the centre plane.

A de-selection field, enumerated by k, must act opposite to a selection field and thus its decay function $g_k(\delta)$, i.e. $g_k(d_{i,k}/r_k)$, must start out horizontally at 0 for $\delta = 0$, increase with increasing δ and level off at 1 when $\delta = 1$, e.g., $g(\delta) \equiv 1 - f(\delta)$:

$$g(\delta) \equiv \begin{cases} -6\delta^3 + 6\delta^2 & \text{if} \quad 0 \le \delta < 0.5 \\ 2\delta^3 - 6\delta^2 + 6\delta - 1 & \text{if} \quad 0.5 \le \delta < 1 \\ 1 & \text{if} \quad 1 \le \delta. \end{cases} \tag{6.4}$$

It may be that other features need protection as well. If, for example, the silhouette of the deck line needs to be preserved throughout a shape variation, one can define a de-selection field emanating from that curve, acting on the z coordinate of shift vectors.

Generalising this, we can allow multiple selection and de-selection fields to be defined for one shape variation setup, each of them emanating from a geometric element that may be a point in space, a curve or a surface[†]. Each field has an individual parameter r, which defines the extent of the field. Both r and d are defined as closest distances to the geometric element on which the field is based. Each selection field j and de-selection field k may act on any subset of coordinates $\{x, y, z\}$, by defining their selection and de-selection functions as diagonal matrices,

$$\mathbf{f}_j\left(\frac{d_{i,j}}{r_j}\right) \equiv \begin{bmatrix} f_{j,x}\left(\frac{d_{i,j}}{r_j}\right) & 0 & 0 \\ 0 & f_{j,y}\left(\frac{d_{i,j}}{r_j}\right) & 0 \\ 0 & 0 & f_{j,z}\left(\frac{d_{i,j}}{r_j}\right) \end{bmatrix} \quad \text{and}$$

$$\mathbf{g}_k\left(\frac{d_{i,k}}{r_k}\right) \equiv \begin{bmatrix} g_{k,x}\left(\frac{d_{i,k}}{r_k}\right) & 0 & 0 \\ 0 & g_{k,y}\left(\frac{d_{i,k}}{r_k}\right) & 0 \\ 0 & 0 & g_{k,z}\left(\frac{d_{i,k}}{r_k}\right) \end{bmatrix}$$

[*] x is the coordinate of length, usually measured from the rudder stock (shaft) or the point of intersection between the stem and the construction waterline. y is the coordinate of width, measured from the centre plane, and z is the coordinate of height, measured from the base plane.

[†] These curves and surfaces need not be straight and planar, although the complexity of the shortest distance problem increases when they are not.

respectively, and setting components to 0 that correspond to coordinate components that should be unaffected by the selection. Finally, each selection field j may have an individual typical shift vector $\mathbf{S}_j$.

Now, the shift vector $\mathbf{s}_i$ of data point i can be defined as the vector sum of all typical shift vectors $\mathbf{S}_j$, which have been multiplied with their selection function $\mathbf{f}_j$ and all de-selection functions $\mathbf{g}_k$:

$$\mathbf{s}_i \equiv \sum_j \left(\prod_k \left(\mathbf{g}_k\left(\frac{d_{i,k}}{r_k}\right)\right) \mathbf{f}_j\left(\frac{d_{i,j}}{r_j}\right) \mathbf{S}_j \right) \tag{6.5}$$

6.4.5 Selection Fields

The test case worked with a spherical selection field, which allowed us to draw bumps and humps in surfaces over dense networks. The resulting deformations may be useful in the entertainment industry, but in design this ability is not of much value. In order to enhance the shape of an existing design, one must be able to define the shape and strength of the selection field in higher detail.

One option is to replace the selection sphere with an ellipsoid, to make it better fit the elongated character of ship hulls. This would increase the number of variables that the user is required to define with two more radii and an orientation besides positioning. But an ellipsoid might still be too primitive in many situations.

A more flexible solution is hinted to in the previous section, namely that the selection field may be based on a curve or a surface. A curve has probably more meaning than a surface in this regard. It may be an existing curve from the network, or a special purpose space curve in its vicinity, or one that is projected onto the surface. If the NURBS algorithm is implemented in a generic way[*], one can easily vary the extent r of the field along the curve (and optionally an extra scaling of the typical shift vector $\mathbf{S}$) by letting it blend control values for these variables in addition to the control points of the curve. This greatly enhances the freedom with which a selection field can be defined.

We are of course also free to choose a different decay function with a different shape. As mentioned before, the decay function from the test case, (6.3) on page 99, was inspired by B-spline basis functions. We recall that the shape of basis functions is controlled by knot distances in the so-called knot vector. By analysis of the recursion formula of

[*]See generic programming (GP) in the Glossary, and Section A.1.2.

Cox-deBoor* for basis functions, it appears that in the cubic case the fall-off of the decay function can be controlled by one knot value, $\kappa \in (0, 1]$. For this the decay function is redefined as follows

$$f\left(\frac{d_{i,j}}{r_j}\right) \equiv f(\delta) \equiv \begin{cases} \dfrac{\kappa^2 + \kappa\delta^2(\delta - 3) + \delta^3}{\kappa^2} & \text{if } 0 \leq \delta < \kappa \\ \dfrac{(\delta - 1)^3}{\kappa - 1} & \text{if } \kappa \leq \delta < 1 \\ 0 & \text{if } 1 \leq \delta \end{cases} \tag{6.7}$$

which is plotted for different values of κ in Figure 6.9 on the next page. Note that for $\kappa = 0$ the function does not depart horizontally at $d = \delta = 0$. This is not illegal, but does not preserve smoothness very well, which is why it was excluded from the range above. Of course, the shape parameter κ may be varied over the curve or surface on which the selection field is based, as described on the previous paragraph for the extent r. This will enhance the possibilities for the designer to control how the shape of the model will be varied.

6.4.6 Variations in Non-Uniform Directions

So far we have shifted all data points in the same direction, only varying the magnitude of the shift. This works well, but only for certain types of variations. If the objective is to inflate a region of relatively high curvature, this mode of operation may prove ineffective, as is illustrated for the two-dimensional case in Figure 6.10(a) on page 110. An obvious alternative may be to shift every data point in the direction of the local surface normal (Figure 6.10(b)). However, that approach will prove to be problematic as soon as the selection

*The p-degree B-spline basis function N, corresponding to the ith control point as a function of parameter t, is defined by the following recursion

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } u_i \leq t < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$
$$N_{i,p}(t) = \frac{t - u_i}{u_{i+p} - u_i} N_{i,p-1}(t) + \frac{u_{i+p+1} - t}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(t) \tag{6.6}$$

where u_i are the non-decreasing knot values. There are $n + p + 1$ knot values, contained in the knot vector, and $u_0 \leq t \leq u_{n+p}$.

field covers a knuckle line, at which the surface normal is undefined (Figure 6.10(c))*.

In search of a working alternative, we may let us inspire by Section 6.4.5, which suggested that the extent of the selection field, the magnitude of the shift and even the shape of the decay function can be varied along the curve that defines a selection field. By adding the shift direction to that set of variables, we may be able to realise a situation in which the designer explicitly defines the new shape of a feature curve. Thus the result of the variation will be highly predictable, and it will be possible to specify the way in which the surrounding surface adapts to the changed feature curve, with a rich set of variables.

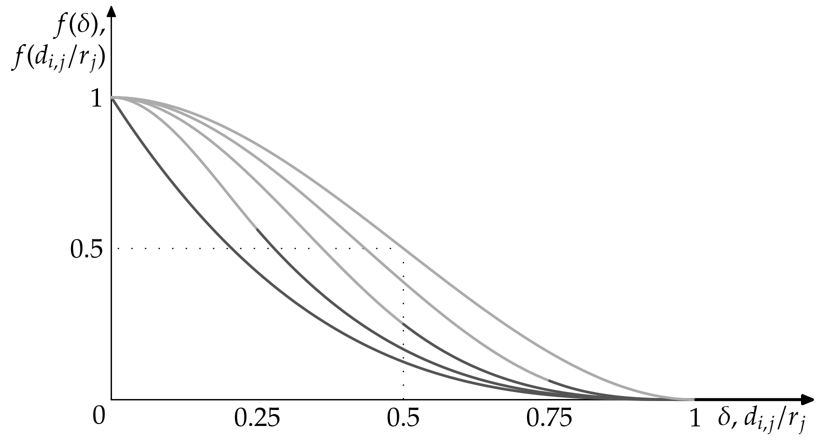
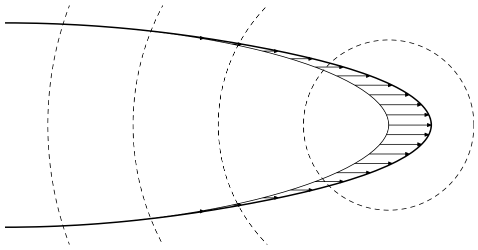


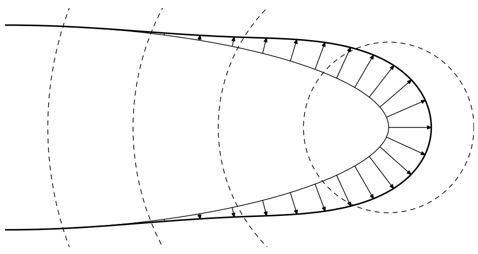
Figure 6.9: Variable cubic decay functions as defined by (6.7) on the preceding page, using light grey when $0 \leq \delta < \kappa$, dark grey when $\kappa \leq \delta < 1$ and black when $1 \leq \delta$. The middle plot is produced with $\kappa = 0.5$ and identical to the plot in Figure 6.2 on page 99. The shape can be given more body with increasing κ: the two rightmost plots were produced with $\kappa = 0.75$ and $\kappa = 1$ respectively, the latter of which is point-symmetric about $(0.5, 0.5)$. With decreasing κ, the curve becomes steeper ($\kappa = 0.25$) until a cusp is formed at $\kappa = 0$.

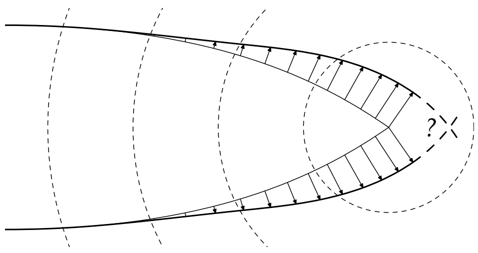
Let us denote the curve that defines the selection field by $\mathbf{c}(t)$, and its explicitly defined shape after the variation operation by $\hat{\mathbf{c}}(\hat{t})$. If $\hat{\mathbf{c}}$ is efficiently described in terms of new positions of the control points of $\mathbf{c}$ (Figure 6.11 on page 111) then these new positions can be attached as meta data to the existing control points, in the same way as was

*The new position of the knuckle point is not related to the vector sum of the shift vectors that exist immediately besides it.

suggested for the other variables. Then we can also adopt $\hat{t} \equiv t$ and define the typical shift vector along the curve $\mathbf{c}$ as $\mathbf{S}(t) \equiv \hat{\mathbf{c}}(t) - \mathbf{c}(t)$. This has the advantage that a generic spline algorithm can compute $\mathbf{S}$ simultaneously with all other variables, with a single evaluation of the spline basis functions*.


(a) Variation with uniform direction.


(b) Variation along the local surface normal.



(c) At knuckles, the surface normal is undefined.

Figure 6.10: The effect of shifting data points in a uniform direction (a) versus shifting data points in the normal direction (b); here in the 2D case. The latter will have problems with cusps and knuckle lines (c), as the normal vector at the cusp is undefined. These figures were computer-generated with a decay function defined by equation (6.3) on page 99, and a selection field that is indicated by the dashed circles, extending up to the outer arc.

*When implementing this method, one could save a few floating point multiplications by storing the difference of the new and old control points, instead of their new positions. With these, the spline algorithm would produce the typical shift vector directly. But the presented form is more graphic, and soon we will discuss extensions, following the same line of thought.

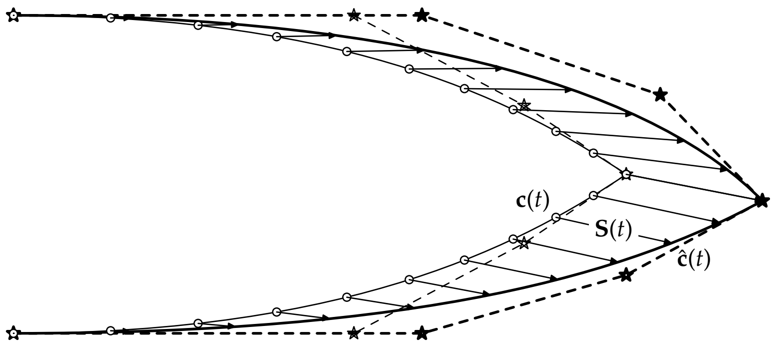


Figure 6.11: Variation by explicit definition of new control point positions. The designer has better control over the shape of the resulting variation, and knuckles and cusps cause no problems. The typical shift vector is now defined as a function of the curve parameter — namely the difference between the new and the old curve — and is variable in both length and direction.

The effect is demonstrated in Figure 6.12 on the following page. An arbitrary curve was projected onto a plane. A selection field was based on it, and the curve pulled in its entirety in normal direction. The surface follows suit as expected.

6.4.7 Re-Design of Curves

At the expense of simplicity and a little efficiency, we may choose to give the designer complete freedom in the definition of the shape of the variation, by detaching the curves $\mathbf{c}$ and $\hat{\mathbf{c}}$ from each other, and giving the latter an independent definition. But, as we will see, the thus far proposed definition of the typical shift vector $\mathbf{S}$ does not handle the increase of freedom very well.

6.4.7.1 Curve Sections as the Basis of Selections

If the original curve is defined with more than a few control points, part of the old and new curve may coincide, i.e., have identical geometry. But the parameterisation of the curves will likely be different, causing the typical shift vector to be non-zero even in parts of the curve that were not changed. This phenomenon is illustrated in Figure 6.13 on the next page.

To prevent this, we will only consider parameter values that produce a change in the geometry of the curve. In other words, say that the curve parameters t and $\hat{t}$ range from $[t_{\text{begin}}, t_{\text{end}}]$ and $[\hat{t}_{\text{begin}}, \hat{t}_{\text{end}}]$

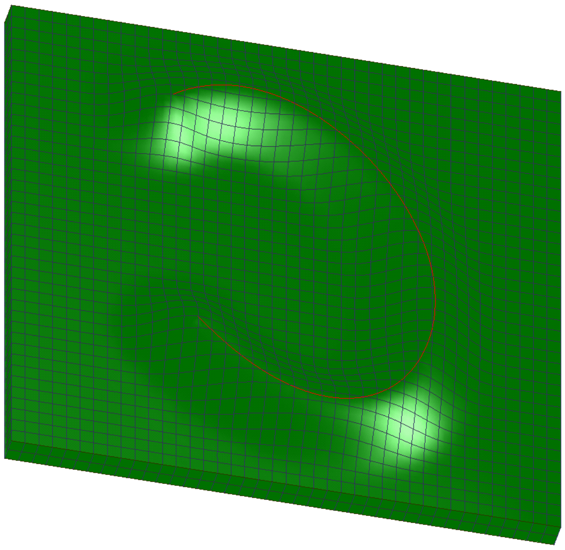


Figure 6.12: Curve based variation.

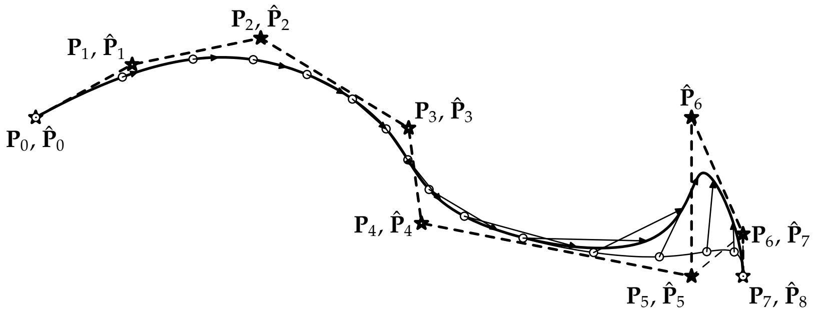

$\mathbf{P}_0, \hat{\mathbf{P}}_0$
$\mathbf{P}_1, \hat{\mathbf{P}}_1$
$\mathbf{P}_2, \hat{\mathbf{P}}_2$
$\mathbf{P}_3, \hat{\mathbf{P}}_3$
$\mathbf{P}_4, \hat{\mathbf{P}}_4$
$\hat{\mathbf{P}}_6$
$\mathbf{P}_6, \hat{\mathbf{P}}_7$
$\mathbf{P}_5, \hat{\mathbf{P}}_5$
$\mathbf{P}_7, \hat{\mathbf{P}}_8$

Figure 6.13: Illustration of the effect that different curve parameterisations have on the typical shift vector, when it is computed on the complete curve. The new curve (thick plot) differs only from the old curve (thin plot) by one extra control point. Thus they largely coincide. But they have different parameterisations, which causes the typical shift vector (indicated by arrows) to be non-zero even where the curve did not change. Both curves are ordinary cubic B-splines.

respectively, and that the curves $\mathbf{c}$ and $\hat{\mathbf{c}}$ differ from each other for $t \in [t_a, t_b]$ and $\hat{t} \in [\hat{t}_c, \hat{t}_d]$. For other parameter values they coincide (although not necessarily for equal parameter values):

$$\begin{aligned}\{\mathbf{c}(t) \mid t_{\text{begin}} \leq t < t_a\} &= \{\hat{\mathbf{c}}(\hat{t}) \mid \hat{t}_{\text{begin}} \leq \hat{t} < \hat{t}_c\}\\ \{\mathbf{c}(t) \mid t_a \leq t < t_b\} &\neq \{\hat{\mathbf{c}}(\hat{t}) \mid \hat{t}_c \leq \hat{t} < \hat{t}_d\}\\ \{\mathbf{c}(t) \mid t_b \leq t \leq t_{\text{end}}\} &= \{\hat{\mathbf{c}}(\hat{t}) \mid \hat{t}_d \leq \hat{t} \leq \hat{t}_{\text{end}}\}.\end{aligned}$$

The selection field will then be defined exclusively on $\{\mathbf{c}(t) \mid t_a \leq t \leq t_b\}$.

Consider the example of Figure 6.13 on the facing page. Both the old and the new curve are cubic B-splines with an ordinary semi-uniform knot vector. These knot vectors are dissimilar due to the extra control point in $\hat{\mathbf{c}}$, reflecting the dissimilar parameterisation: for normalised parameters, $\mathbf{c}$ has knot vector $\mathbf{U} = [\,0\;0\;0\;0\;\frac{1}{5}\;\frac{2}{5}\;\frac{3}{5}\;\frac{4}{5}\;1\;1\;1\;1\,]$ and $\hat{\mathbf{c}}$ has knot vector $\hat{\mathbf{U}} = [\,0\;0\;0\;0\;\frac{1}{6}\;\frac{2}{6}\;\frac{3}{6}\;\frac{4}{6}\;\frac{5}{6}\;1\;1\;1\;1\,]$. In order to determine the parameter values for which $\hat{\mathbf{c}}$ does not coincide with $\mathbf{c}$, we need to know what parameter values are controlled by each individual control point, and consider the control points that were moved or added. This translates into finding the support of the corresponding basis functions, i.e, where they are non-zero*. This is best illustrated by drawing out the knot vectors on the parameter axis (Figure 6.14 on the following page) and marking the support of the basis functions N_i that correspond to the control points $\mathbf{P}_i$.

From Figure 6.13 we see that $\mathbf{P}_i = \hat{\mathbf{P}}_i$ for $i \equiv 0, \ldots, 5$, and the difference starts with control point number 6. From Figure 6.14 on the next page we see that $\mathbf{P}_6$ starts to control curve $\mathbf{c}$ at $t = \frac{3}{5}$ (the lower bound of the support of N_6), and likewise, $\hat{\mathbf{P}}_6$ causes a change in $\hat{c}$ beyond $\hat{t} = \frac{3}{6} = \frac{1}{2}$. Thus we have

$$\begin{aligned}\left\{\mathbf{c}(t) \,\middle|\, 0 \leq t < \tfrac{3}{5}\right\} &= \left\{\hat{\mathbf{c}}(\hat{t}) \,\middle|\, 0 \leq \hat{t} < \tfrac{1}{2}\right\}\\ \left\{\mathbf{c}(t) \,\middle|\, \tfrac{3}{5} \leq t \leq 1\right\} &\neq \left\{\hat{\mathbf{c}}(\hat{t}) \,\middle|\, \tfrac{1}{2} \leq \hat{t} \leq 1\right\}\end{aligned}$$

and consequently the selection field should be defined on $\left\{\mathbf{c}(t) \,\middle|\, \frac{3}{5} \leq t \leq 1\right\}$. For the computation of $\mathbf{S}$, t and $\hat{t}$ must proportionally vary over $\left[\frac{3}{5}, 1\right]$ and $\left[\frac{1}{2}, 1\right]$ respectively. Figure 6.15 on the following page is drawn accordingly.

*As a direct consequence of the Cox-deBoor recursion formula, (6.6) on page 108, basis functions of order k (degree $k-1$) are non-zero over k contiguous knot spacings. More precisely, $N_i^k(t) > 0$ for $u_i < t < u_{i+k}$ and $N_i^k(t) = 0$ elsewhere. A cubic curve is denoted by $k = 4$.

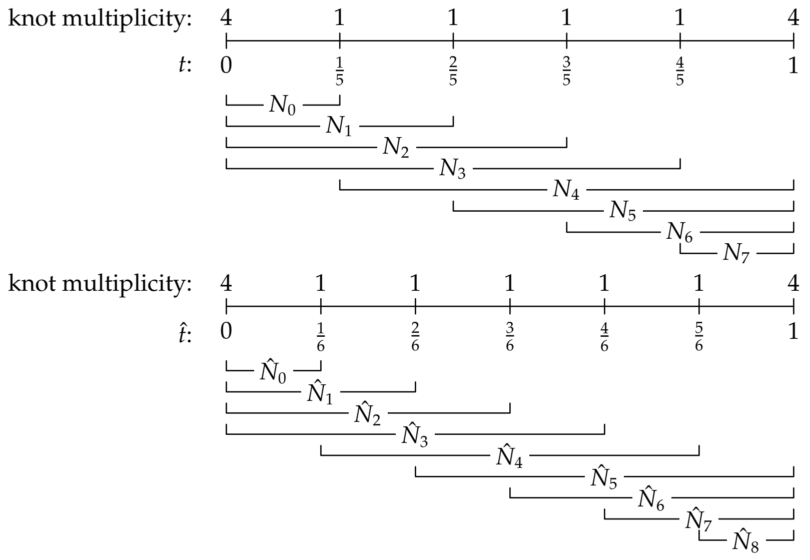

knot multiplicity: 4 1 1 1 1 4
t: 0 $\frac{1}{5}$ $\frac{2}{5}$ $\frac{3}{5}$ $\frac{4}{5}$ 1
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7
knot multiplicity: 4 1 1 1 1 1 4
$\hat{t}$: 0 $\frac{1}{6}$ $\frac{2}{6}$ $\frac{3}{6}$ $\frac{4}{6}$ $\frac{5}{6}$ 1
$\hat{N}_0$
$\hat{N}_1$
$\hat{N}_2$
$\hat{N}_3$
$\hat{N}_4$
$\hat{N}_5$
$\hat{N}_6$
$\hat{N}_7$
$\hat{N}_8$

Figure 6.14: Knot values and the support of basis functions drawn out for the curves **c** (upper) and $\hat{\mathbf{c}}$ (lower) from Figure 6.13 on page 112.

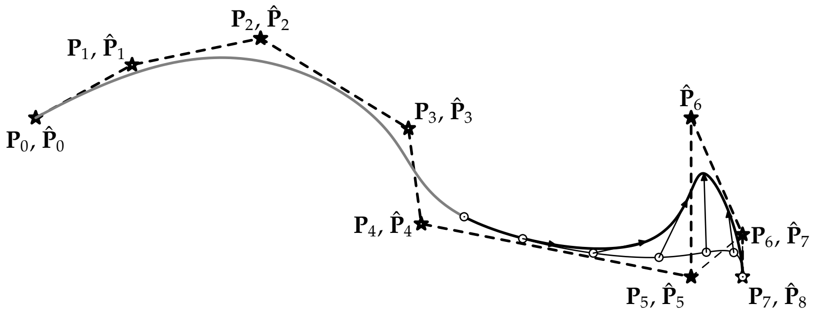

$\mathbf{P}_0, \hat{\mathbf{P}}_0$
$\mathbf{P}_1, \hat{\mathbf{P}}_1$
$\mathbf{P}_2, \hat{\mathbf{P}}_2$
$\mathbf{P}_3, \hat{\mathbf{P}}_3$
$\mathbf{P}_4, \hat{\mathbf{P}}_4$
$\mathbf{P}_5, \hat{\mathbf{P}}_5$
$\hat{\mathbf{P}}_6$
$\mathbf{P}_6, \hat{\mathbf{P}}_7$
$\mathbf{P}_7, \hat{\mathbf{P}}_8$

Figure 6.15: The typical shift vector for an explicit variation, defined exclusively on the subsection of the curve that actually changes geometry. Shifts in other parts of the curve are thus prevented. The curves displayed here are identical to the ones in Figure 6.13 on page 112.

6.4.7.2 Arc Length Parameterisation

There is another aspect that can have unwanted effects on the typical shift vector as it is defined so far. Choosing to vary t and $\hat{t}$ *proportionally*, works well only if the parameterisations of $\mathbf{c}$ and $\hat{\mathbf{c}}$ are similar. If there are large differences in their parameterisations, the typical shift vector will skew back and forth along the curves, which can cause unwanted effects in resulting shape. Dissimilar parameterisations are caused by un-proportional variations of control point spacings and un-proportional variations of knot spacings in the knot vectors. Figure 6.16(a) on the next page gives an example of this: the curves have equal knot vectors, but the control points of $\mathbf{c}$ are positioned close to the curve ends, while $\hat{\mathbf{c}}$ has two control points near the middle of the curve. For a better definition of the typical shift vector, we need to make their parameterisations similar.

The spacings of control points can be made compatible by inserting extra control points in both control polygons. This can be done without changing the geometry of the curve by a process called *knot insertion* [Piegl and Tiller, 1997, section 5.2]*. But, as its name indicates, this also inserts a knot into the knot vector (making it non-uniform) which compensates the effect of adding the control point. In fact, knot insertion does not change the parameterisation of a curve.

A working solution would be to evaluate the curves not at proportional parameter values, but at proportional arc lengths, as in Figure 6.16(b) on the following page. Arc length (the physical length of a curve or curve section) is a curve property independent of the parameterisation. For a formal definition we need a mapping m that maps t to $\hat{t}$

$$m\colon t \mapsto \hat{t}$$

such that for a certain parameter value t_i, $m(t_i)$ produces $\hat{t}_j$ (shamelessly reusing subscripts i and j from a different context) such that the arc lengths of the curve sections on either side of these parameter values are proportional:

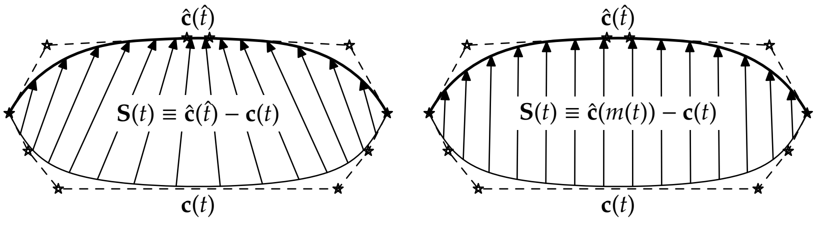
$$\frac{\int_{t_a}^{t_i} |\dot{\mathbf{c}}(t)| \mathrm{d}t}{\int_{t_i}^{t_b} |\dot{\mathbf{c}}(t)| \mathrm{d}t} = \frac{\int_{\hat{t}_c}^{\hat{t}_j} |\hat{\mathbf{c}}(\hat{t})| \mathrm{d}\hat{t}}{\int_{\hat{t}_j}^{\hat{t}_d} |\hat{\mathbf{c}}(\hat{t})| \mathrm{d}\hat{t}} \tag{6.8}$$

*Inserting several knots is possible by repeated knot insertion, or all at once by a process called knot refinement [Piegl and Tiller, 1997, section 5.3] which is more efficient.

in which the arc length of a curve is defined as the integral of the length of the first derivative of that curve with respect to the curve parameter [Boehm, 2002][*].

Now we are able to define the typical shift vector **S** as the difference between relating positions on the two curves according to arc length, expressed as a function of the curve parameter t:

$$\mathbf{S}(t) \equiv \hat{\mathbf{c}}(m(t)) - \mathbf{c}(t). \tag{6.9}$$



$\hat{\mathbf{c}}(\hat{t})$
$\mathbf{S}(t) \equiv \hat{\mathbf{c}}(\hat{t}) - \mathbf{c}(t)$
$\mathbf{c}(t)$
$\hat{\mathbf{c}}(\hat{t})$
$\mathbf{S}(t) \equiv \hat{\mathbf{c}}(m(t)) - \mathbf{c}(t)$
$\mathbf{c}(t)$

(a) Ordinary parameterisation. (b) Arc length parameterisation.

Figure 6.16: Ordinary parameterisation versus arc length parameterisation. The typical shift vector **S** defined on two cubic B-spline curves with equal knot vectors but dissimilar control point spacings.

Direct evaluation of $m(t)$ is not efficient though, for two reasons. Firstly, the integrals in (6.8) on the page before must be computed numerically, for which the complete curve sections must be evaluated. And secondly, for each evaluation of $m(t)$ an optimisation problem must be solved, as we are searching for the value $\hat{t}_j$ that produces the correct ratio in the right hand side of (6.8). For our application, the accuracy of the arc length computation is not as important as the smoothness of $m(t)$. For an interactive system, one should first consider whether arc length parameterisation is at all worthwhile, by comparing internal knot spacings and control point distances and looking for large discrepancies. If so, the function m may be approximated by evaluating $m(t)$ at distinct values of t_i, and fitting a polynomial, say

[*]Referring to physics makes the definition of arc length easy to comprehend. Let the position of a moving particle be defined as a function of time, e.g., $\mathbf{c}(t)$. The first derivative with respect to t of that function, which is the vector $\dot{\mathbf{c}}(t)$, indicates the direction of movement of the particle, and its velocity is given by the length of that vector, $|\dot{\mathbf{c}}(t)|$. The distance that the particle travels, i.e., the arc length of its path $\mathbf{c}$, is of course equal to the velocity integrated over time, $\int |\dot{\mathbf{c}}(t)| \mathrm{d}t$.

$\hat{m}(t)$, through the resulting function values. In this case, the integrals on the left hand side of (6.8) need only be computed once if they are split up into smaller sections of $\Delta t \equiv t_{i+1} - t_i$, and using combinations of these in the ratio. The computation of the integrals on the right hand side of (6.8) can also be made more efficient by reusing the answer from the previous run. Application of a Newton-Cotes formula such as Simpson's rule may further help increasing the efficiency of solving the integrals.

There is just one more issue that needs to be addressed. In Koelman's implementation, **c** may actually be a poly-curve, i.e., it may be assembled of several individual curve sections that are connected by specific master/slave relations. In such cases, we must also define $\hat{\mathbf{c}}$ to be a poly-curve, with a topology that is identical to the topology of **c**. Thus, knuckle points and other discontinuities that exist in **c**, also exist in $\hat{\mathbf{c}}$. Then, if the corresponding curve sections need arc length parameterisation, individual maps $\hat{m}$ need to be computed for each such section in the poly-curve.

6.4.7.3 The Combination, an Example

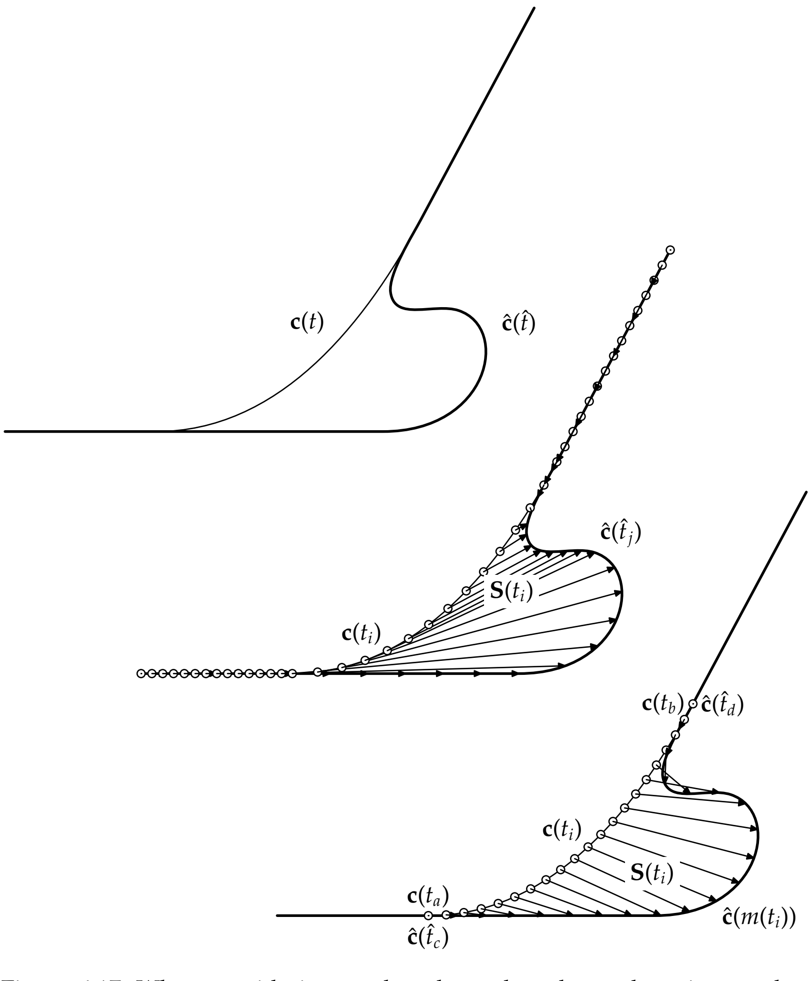
Figure 6.17 on the following page shows the effect of the measures described above. This example concerns the re-design of the stem curve of a conventional bow that is being transformed into a bulbous bow.

6.4.8 Unconnected Selections

The fourth shortcoming listed on page 104 mentions the unfortunate fact that all data points that fall within the selection field are shifted, regardless whether they form a connected region or not. As an example, consider the two sides of a wing or propeller blade. A selection field that is meant to vary only one side, will likely also select a disconnected "island" of data points on the other side of the wing or blade.

The robust solution would be to measure distances between points on the model over its surface instead of through space, (which is the subject of Section 6.5), but measuring in straight lines is so much more efficient. Because Euclidean distance dependent variations seem to work in most cases, let us briefly consider what can be done to counter the described situation — which luckily appears rather rarely.

The pragmatic approach would be to let the user identify these pockets of unwanted selections, and provide tools by which they can



$\mathbf{c}(t)$
$\hat{\mathbf{c}}(\hat{t})$
$\hat{\mathbf{c}}(\hat{t}_j)$
$\mathbf{S}(t_i)$
$\mathbf{c}(t_i)$
$\mathbf{c}(t_b)$
$\hat{\mathbf{c}}(\hat{t}_d)$
$\mathbf{c}(t_i)$
$\mathbf{S}(t_i)$
$\mathbf{c}(t_a)$
$\hat{\mathbf{c}}(\hat{t}_c)$
$\hat{\mathbf{c}}(m(t_i))$

Figure 6.17: When considering arc lengths and unchanged sections makes sense. Upper left: original stem curve $\mathbf{c}(t)$ and re-designed stem curve $\hat{\mathbf{c}}(\hat{t})$. Middle: straight-forward definition of typical shift vector $\mathbf{S}(t) \equiv \hat{\mathbf{c}}(\hat{t}) - \mathbf{c}(t)$. Data points are shifted even where geometry is not changed, and their spacing becomes unbalanced. Lower right: considering arc lengths over sections of changed geometry, $\mathbf{S}(t) \equiv \hat{\mathbf{c}}(m(t)) - \mathbf{c}(t)$, gives a much cleaner result.

simply be de-selected. One can think of a (de)selection sphere that de-selects every data point within it, or a plane that de-selects everything that falls in front or behind it.

An automatic approach can be realized by putting the topologic information from the network to good use, combined with elements from graph theory to de-select data points in isolated sets. Let us define the *root* of a selection as the data point at which a selection field is centred, or the data point that is closest to the geometric element on which a selection field is based. Then we redefine a data point to be selected if and only if

1. the data point is positioned within the selection field, *and*
    a) the data point is the selection root, *or*
    b) the data point is part of the boundary of a network cell* that borders to other selected data points.

In other words, to be selected, data points must not only be covered by the selection field, they must also be connected through network cells that have one or more selected data points in their boundary. This prunes away isolated flocks of data points that are separated from the main pack (containing the selection root) by more than one network cell.

Because of the computational cost of this analysis, it is not suited for interactive use. It would probably fit best as an optional step in between the preview stage and the actual application of the surface variation. Figure 6.18 on the next page proposes an algorithm in pseudo code that implements the above rules.

6.4.9 Finishing Up

Once the fitting/fairing algorithm has re-interpolated the curves over the shifted data points, the H-rep has become a consistent and smooth modification from the original, by which we have succeeded in our objective. However, if the original primarily consisted of planar curves, such as is customary in the design of ship hulls, these may no longer be planar after the modification.

Planar curves can be restored by intersecting the modified model with the planes in which the curves were originally defined, and adding the intersection curves to the model. These new curves take over the definition of the modified shape, by which the old curves become redundant and may be removed.

*A network cell is also known as a *face* in B-rep terminology.

```
Let $\mathbb{A}$ be an empty set, capable of containing node references.
Mark all nodes and faces in the B-rep as unconsidered.
Mark the root node as considered and add it to $\mathbb{A}$.
While $\mathbb{A}$ is not empty {
    Compute $\mathbf{s}_i$ for a node $i \in \mathbb{A}$.
    If $|\mathbf{s}_i| > 0$, then {
        For all faces $j$ that are adjacent to node $i$ and
                                        that are still unconsidered {
            Mark $j$ as considered.
            For all nodes $k$ that are adjacent to $j$ and
                                        that are still unconsidered {
                Mark $k$ as considered and add it to $\mathbb{A}$.
            }
        }
        Shift node $i$.
    }
    Remove $i$ from $\mathbb{A}$.
}
```

Figure 6.18: Pseudo-code of an algorithm that only shifts contiguous sets of data points.

6.5 Surface-Distance Dependant Variation

When the first experiment on Euclidean distance dependent variations was designed, see Section 6.4.2, it was meant as a preparation for the more difficult matter of surface-distance dependent variations. Surface-distance dependent variations, in which the shortest distance between points on the surface is computed *over the surface* instead of in a straight line through space, were expected to be more robust and more intuitive. This idea came from an early brain-storm session in which it was imagined to heat up an area on the hull surface, being made of a material with temperature-dependent flexibility, after which the area could be deformed.

Much unexpected, Euclidean distance dependent variations have in the mean time evolved to an effective and efficient tool for geometric design — as will be evaluated in the section hereafter. It does not score bad on the intuitive scale either, if that would be ones concern. Regarding the heat-and-deform analogy mentioned above, it is easy to think of the shape as a solid and not a surface that is being heated up.

Apart from the fact that time constraints in this study do not allow implementing and testing of a surface-distance dependent variation method, there is reason to expect that such an effort would *not* be fruitful; mainly due to high computational cost and requirements on precision to warrant surface fairness.

6.5.1 Geodesics

The shortest distance between two points on a curved surface is the length of a geodesic. A geodesic is a locally length-minimising curve. Equivalently, it is a path that a particle which is not accelerating would follow [Weisstein and Rowland, 2004]. In general, there may be many geodesics between two points, not all of which attain the shortest possible length.

For example, take two non-antipodal points on a sphere. These points are on a great circle. One arc of the great circle is the shortest path between the points, the other arc is not. Both arcs are geodesics.

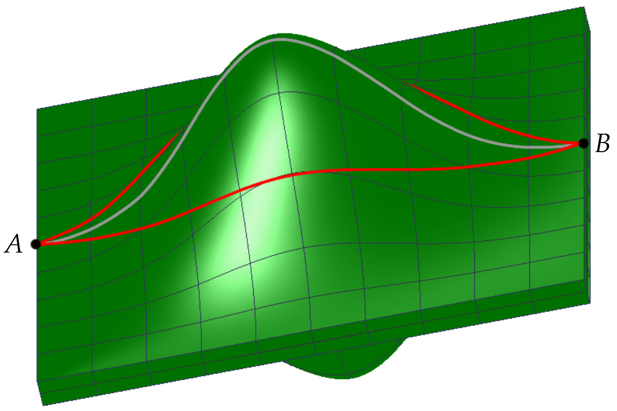


Figure 6.19: Three (approximate) geodesics between two points on the surface.

Another example is given in Figure 6.19 above, showing that different geodesics can exist quite close together. The points under consideration are marked by black dots. The fat gray curve connecting them is a geodesic, but its length is not the shortest distance between the curves. Two shorter geodesics exist, approximated by the fat red curves. In this case the elevation is symmetric and the red geodesics have the same length. This is a simple case, variants of which may very well occur in a ship hull design. So, for a robust implementation, for the computation of any one distance we need to know *all* geodesics between the two measuring points, and take the shortest one.

We will not go all the way, but just to give an impression of the complexity we would be dealing with, let us see what can be said about geodesics analytically, after Weisstein and Rowland [2004]. To keep things simple, we are considering geodesics on a single quadrilateral patch, for which the coordinates are defined parametrically by $x = x(u,v)$, $y = y(u,v)$ and $z = z(u,v)$. A geodesic can be found by minimising the arc length

$$L \equiv \int ds = \int \sqrt{dx^2 + dy^2 + dz^2}. \tag{6.10}$$

For this we turn to calculus of variations, a branch of mathematics which seeks to find the path, curve, surface, etc., for which a given function has a stationary value (which, in physical problems, is usually a minimum or maximum). Mathematically, this involves finding stationary values of integrals of the form

$$I \equiv \int f(a,b,b')da, \tag{6.11}$$

where $b' \equiv \frac{db}{da}$. I has an extremum only if the Euler-Lagrange differential equation is satisfied, i.e., if

$$\frac{\partial f}{\partial b} - \frac{d}{da}\left(\frac{\partial f}{\partial b'}\right) = 0. \tag{6.12}$$

Our aim is to recognise (6.11) above in (6.10) and then formulate the corresponding Euler-Lagrange differential equation. Solving that differential equation would give us the geodesics.

In (6.10) above, note that

$$dx = \frac{\partial x}{\partial u}du + \frac{\partial x}{\partial v}dv, \quad \text{so}$$

$$dx^2 = \left(\frac{\partial x}{\partial u}\right)^2 du^2 + 2\frac{\partial x}{\partial u}\frac{\partial x}{\partial v} + \left(\frac{\partial x}{\partial v}\right)^2 dv^2$$

and similarly for dy^2 and dz^2. Substitution gives

$$L \equiv \int \Bigg\{ \left[\left(\tfrac{\partial x}{\partial u}\right)^2 + \left(\tfrac{\partial y}{\partial u}\right)^2 + \left(\tfrac{\partial z}{\partial u}\right)^2 \right] du^2 + 2\left[\tfrac{\partial x}{\partial u}\tfrac{\partial x}{\partial v} + \tfrac{\partial y}{\partial u}\tfrac{\partial y}{\partial v} + \tfrac{\partial z}{\partial u}\tfrac{\partial z}{\partial v} \right] du\,dv + \left[\left(\tfrac{\partial x}{\partial v}\right)^2 + \left(\tfrac{\partial y}{\partial v}\right)^2 + \left(\tfrac{\partial z}{\partial v}\right)^2 \right] dv^2 \Bigg\}^{\frac{1}{2}}. \quad (6.13)$$

To simplify the notation, define

$$P \equiv \left(\frac{\partial x}{\partial u}\right)^2 + \left(\frac{\partial y}{\partial u}\right)^2 + \left(\frac{\partial z}{\partial u}\right)^2$$

$$Q \equiv \frac{\partial x}{\partial u}\frac{\partial x}{\partial v} + \frac{\partial y}{\partial u}\frac{\partial y}{\partial v} + \frac{\partial z}{\partial u}\frac{\partial z}{\partial v}$$

$$R \equiv \left(\frac{\partial x}{\partial v}\right)^2 + \left(\frac{\partial y}{\partial v}\right)^2 + \left(\frac{\partial z}{\partial v}\right)^2,$$

and (6.13) above becomes

$$L \equiv \int \sqrt{P du^2 + 2Q du dv + R dv^2}.$$

Bringing du^2 outside the root and writing v' for $\frac{dv}{du}$ gives

$$L \equiv \int \sqrt{P + 2Qv' + Rv'^2}\, du.$$

Now L can be recognised as being an integral of the form (6.11) on the facing page, with $da \equiv du$, $b' \equiv \frac{db}{da} \equiv \frac{dv}{du} \equiv v'$. More to the point, $a \equiv u$ and $b \equiv v$, and

$$f \equiv \sqrt{P + 2Qv' + Rv'^2}.$$

Taking the derivatives,

$$\begin{aligned}
\frac{\partial f}{\partial v} &= \tfrac{1}{2}(P + 2Qv' + Rv'^2)^{-\frac{1}{2}}\left(\frac{\partial P}{\partial v} + 2\frac{\partial Q}{\partial v}v' + \frac{\partial R}{\partial v}v'^2\right)\\
&= \frac{\frac{\partial P}{\partial v} + 2v'\frac{\partial Q}{\partial v} + v'^2\frac{\partial R}{\partial v}}{2\sqrt{P + 2Qv' + Rv'^2}} \quad \text{and}\\
\frac{\partial f}{\partial v'} &= \tfrac{1}{2}(P + 2Qv' + Rv'^2)^{-\frac{1}{2}}(2Q + 2Rv')\\
&= \frac{Q + Rv'}{\sqrt{P + 2Qv' + Rv'^2}},
\end{aligned}$$

so the Euler-Lagrange differential equation (6.12) on page 122 then gives

$$\frac{\frac{\partial P}{\partial v} + 2v'\frac{\partial Q}{\partial v} + v'^2\frac{\partial R}{\partial v}}{2\sqrt{P + 2Qv' + Rv'^2}} - \frac{d}{du}\left(\frac{Q + Rv'}{\sqrt{P + 2Qv' + Rv'^2}}\right) = 0. \tag{6.14}$$

If you can solve this differential equation, you have the geodesics of one quadrilateral patch. Remains to find a similar differential equation for n-sided patches where $n \neq 4$, find a framework to concatenate the shortest geodesics over adjacent patches and find the appropriate patch boundary crossings, and to find an efficient method for solving the differential equations — which will likely be a numerical method. But, we will not go there in this thesis.

6.5.2 Concerns for Surface Fairness

Consider Figure 6.19 on page 121 once more, but imagine this time that one of the two points in that figure, say point B, is duplicated into B_1 and B_2. Then offset the two copies slightly from the original position of B, in opposite directions — B_1 to the left and B_2 to the right when seen from A. Now the shortest geodesic between A and B_1 will be very different from the shortest geodesic between A and B_2, even if B_1 and B_2 only differ by the smallest fraction. One will look like the red geodesic that passes behind the elevation, the other like the one in front in Figure 6.19. This example is symmetric in AB. In the general case however, the two geodesics may be of very different shape and length.

Because distances may be measured over very different paths even for measurements that are almost the same, truncation errors caused by the numerical computation of the geodesics can cause an error in the computed distances $|AB_1|$ and $|AB_2|$ that is large in relation to the distance $|B_1B_2|$. This error propagates via the shift vectors to the new position of B_1 and B_2, labelled $\hat{B}_1$ and $\hat{B}_2$. Because $\hat{B}_1$ and $\hat{B}_2$ are still closely spaced with respect to the parameter values of the curve that is going to be fitted through them, the error may be amplified in the curve to a visible magnitude. This phenomenon is illustrated in Figure 6.20 below.

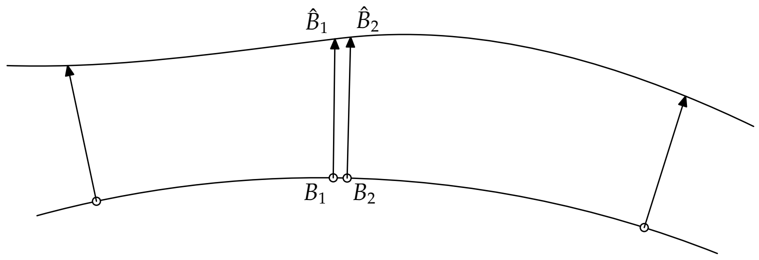


Figure 6.20: How a small error in the shift vector can show up as a large error in fairness of the resulting shape.

6.5.3 Computational Complexity

So far we have only considered the distance between two points on the surface. For a curve-based variation, however, shortest distance between a point and a curve. Therefore we need not only do a global search for the shortest geodesic, but we need to do that while we are searching for the point on the curve to measure from that will give the shortest distance — which is in itself a global search with respect to the curve. This pushes the computational complexity to a higher order, and the constants are solid. The time needed for the computations to complete would likely be far longer than the time users are willing to wait.

We can conclude that the efforts of implementing a surface-distance dependent shape variation are too high and the prospects too dim to be considered any longer in this thesis, especially since we have a working and much simpler method, the Euclidean distance dependent variation.

6.6 Dirichlet Parameterisation

As discussed on page 83, Marsan *et al.* [2001] accomplish interesting results by pursuing the heat transfer analogy further, by means of what they call Dirichlet parameterisation. However, they find it necessary to work on a 2D mapping of the surface, which may be problematic for some variations. Also, their method solves the Laplace equation numerically by means of FEM, which involves a discretisation. This may introduce small discontinuities and a numerical error, which gives rise for concerns for surface fairness as discussed in Section 6.5.2.

6.7 Chapter Summary and Look Ahead

In the above chapter a solution was developed that answers research questions 1 and 2 (see Section 1.2.1). The principle of shifting points by means of B-spline decay functions was shown to be applicable to H-rep models, and extendible to the degree that an H-rep surface can be made to adapt, over a variable area in a variable way, to changes in its defining curves. The concept of de-selection functions was introduced to constrain particular point sets from shifting in one or more main directions.

The presented method is purely geometric. Distances are measured in Euclidean space, which gives good interactive response. Other parameterisations were considered, but discarded because of lower predicted efficiency and/or effectiveness. The contribution is an improvement to the state of the art in the paradigm of surface representation by interpolation of arbitrarily intersecting curves.

In the next chapter, we will evaluate the method with respect to its practical value and the degree of its innovation. Recommendations for an industrial-strength implementation will also be made.

CHAPTER 7

Evaluation

In this chapter we will reflect on the method that was developed in the previous chapter. We will start with an evaluation of the value and limitations of the method with respect to practical design situations. Then, we will consider the relevance and novelty of the contribution. Recommendations follow, for an industrial-strength implementation of the proposed method*. Conclusively, we will step back for a wider view, and compare the improved state of the art in the interpolation paradigm with the approximation paradigm once again, answering the third and last research question that was posed in Section 1.2.1.

7.1 Practical Value

Theses on new design methodology and new computer methods often include a user base evaluation, in which a group of external individuals of varying professionalism use and test the method. Their actions are monitored and their feedback is documented, and the result is meant to be a measure of the value of the contribution. For such a study to be meaningful, the computer programme has to be of fairly high quality. Humans are easily distracted and likely irritated by small issues that are irrelevant to the method, like a programme crash, an awkward user interface, long waiting times due to a suboptimal implementation, lacking conveniences such as "undo", incomplete interactive control and lacking support for specific situations. The

*Aspects of programming methodology are discussed in Appendix A.

currently available implementation of the method serves only proof of concept, and has all the above deficiencies. A user base evaluation at this stage would mainly have produced a lot of noise, and it might be impossible to draw any meaningful conclusion.

A better option is to do the evaluation ourselves, and try to be as complete and objective as possible. In the following, four specific cases will be presented in which the shape of an example H-rep hull model will be varied. In some situations this involved making adjustments to the computer code, to accommodate configurations that otherwise would have been possible through an advanced user interface. In each case, we will reflect on the result by considering how an application of FFD would have performed, and how the same variation can be produced by traditional H-rep modelling. This should give a clear impression of the practical value of the proposed method.

7.1.1 Narrowing the Fore Ship

Figure 7.1 on the facing page illustrates the effect of a point based variation, which has a spherical field of influence. The original shape is visible as a red wire-frame overlay. The field was based on a data point on the surface at half height, about $\frac{1}{8}$ of the ships length from the front. The extent of the field was large, reaching almost up to the stern, leaving about $\frac{1}{4}$ of the ships length unaffected. The direction of the typical shift vector was transverse and inward. There was a constraining de-selection field set up, emanating from the centre plane with an extent reaching up to $\frac{2}{5}$ of the ships breadth, to prevent the hull from deforming into its own mirror image.

Because the direction of the typical shift vector was in the plane of both water lines and frames, only the buttock lines were drawn out of their defining planes. For this they were inserted anew.

7.1.1.1 Comments

There are some things that can be said about the quality of the variation. There is one shape deficiency just below the aft end of the knuckle line, where three curves intersect in (almost) the same point. This is an artifact inherent to the current H-rep implementation as is discussed further on page 146.

Another artifact can be seen in front of the very foremost frame. An unwanted inflection in the surface is visible, which is not a cause of the surface variation but of a local scarcity of defining curves. This

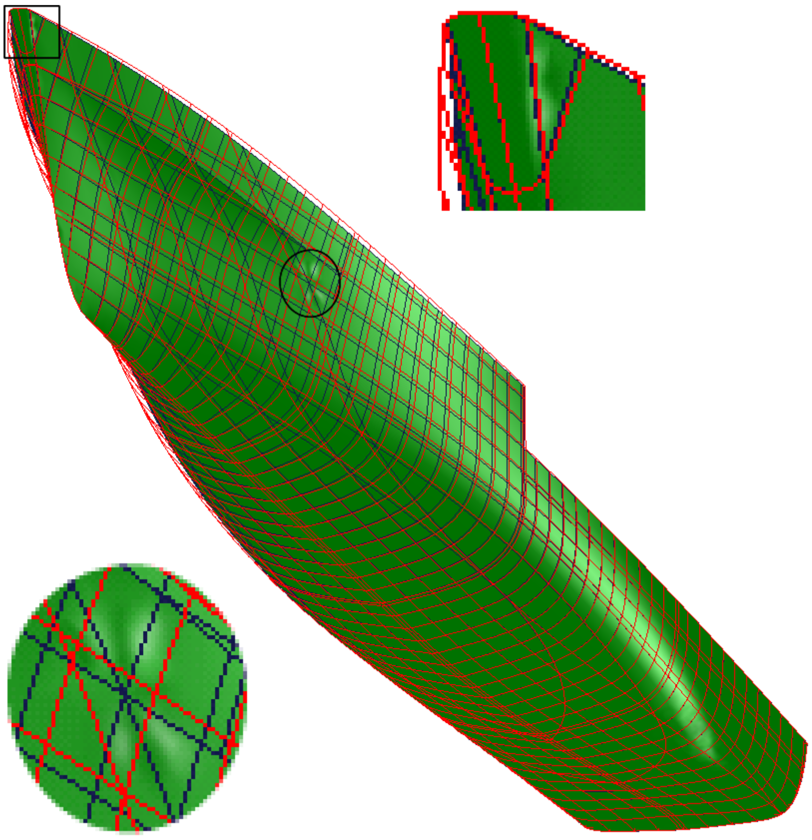


Figure 7.1: Point based variation. The original shape is displayed as a red wire-frame overlay. The varied hull, globally manipulated to become slightly narrower in the front, is displayed in green with a blue wire-frame. Details of shape deficiencies are enlarged.

can be prevented by a more relaxed mode of surface interpolation or by adding an extra water line in that region.

The shown variation can be criticised for the shape near the stem curve. It seems that the old geometry shines through too much, causing an extra inflection under the knuckle line towards the stem curve. Here the de-selection function has preserved not only the position of points near the centre plane but also (as a consequence) the angle at which the port and starboard shell plating meet at the stem. Because the bow has become narrower, the angle should have become smaller as well, if the variation would have been correct. Probably the problem could have been reduced by decreasing the extent of the de-selection field. An even better approach would have been to not base the de-selection field on the centre plane but on the stem curve and base line itself, with the shape of the decay function varying along it by means of the κ variable. By giving it a cusp ($\kappa = 0$) below the water line down to the base line, a proper shape variation can be expected. Unfortunately, time constraints have not allowed implementation of curve based de-selection fields, so this statement cannot be verified.

7.1.1.2 Alternatives

This is the kind of shape variation that could be very well accomplished with free-form deformation (FFD) as well, especially when direct manipulation of the FFD lattice [Hsu *et al.*, 1992] is used. But FFD would still need the definition of a lattice, and it could not have beaten the efficiency of point based shape variation with a decay function. The demonstrated operation was very fast and interactive; there were no waiting times for the designer. The whole variation was a question of 1–2–3:

1. Point and click to mark the centre of the selection field.
2. Move the mouse to dynamically set the extent and the magnitude of the variation, while watching the preview of the shift.
3. click to finalise the variation, which happens instantly.

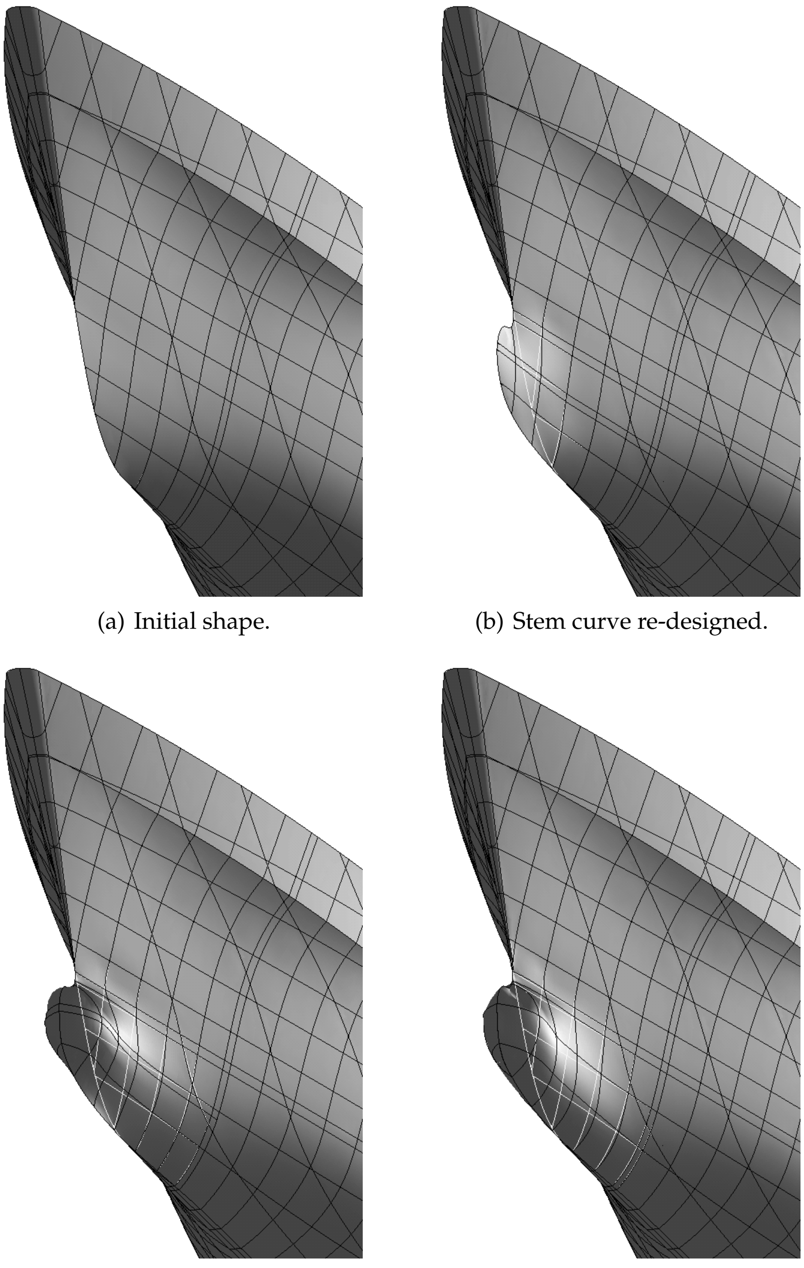
A shape variation of this kind without any means of global manipulation, using only prior existing functionality in *Fairway*, is very hard. One would start with removing as many curves as possible, to reduce the number of curve intersections. It is important to remove only curves that are redundant for the shape definition, otherwise the shape looses detail in this process. There is no functionality in *Fairway* to help determine whether a curve is redundant or not[*]. Then one would start changing curves one at a time, probably deleting control points to reduce the degrees of freedom and to better be able to attain curve fairness. At this point, network consistency will probably be lost. In the editing of each curve, one must attempt to apply a similar change as applied to other curves. Then starts an iterative process of restoring network consistency, consisting of snapping data points back onto curves, setting weights on data points for the fairing algorithm, adjusting curves to each other, checking intersection tolerances and evaluating curve fairness. One may have to visit each curve several times. At the end of this work, curves that were removed may be re-inserted.

7.1.2 Construction of a Bulbous Bow

Point based variations are useful for subtle variations over very large areas, of the kind that was demonstrated above. They are not suited for many other variations, if they are to be constructive. For those, we have to turn to curve based variations.

Curve based variations allow us to completely re-design any curve in the H-rep curve network, without needing to worry about network consistency. This is illustrated in Figure 7.2 on pages 132 and 133, where a bulbous bow is constructed in three successive curve based variations. Figure 7.2(a) shows the initial shape. For the first variation, a selection field is based on the stem curve, which is formed into the contour of a bulb below the construction water line. The typical shift vector is defined as the difference between the old and the new stem curve, just as in Figure 6.17 on page 118, so that the hull shape is stretched to match the new stem curve (Figure 7.2(b)). To give the bulb body, we want to pull at half height of the bulb. Because the nearest water line is just above half height, a special purpose curve is inserted for the occasion, on which the second variation is based (Figure 7.2(c)). The shape near the bottom of the bulb still can use some work, because some frames in this region have an extra inflection. This is worked out by dragging the lowest water line a bit wider (Figure 7.2(d)). The shape of the curves are looking good at this point, only the surface has a dent at the top of the bulb. This is because of large local changes in curvature without enough curves to guide the surface. By inserting a few extra curves and manipulating them (no shape variations needed) the dent can be polished away (Figure 7.3(e)).

[*]The feeling is comparable with to renovate an apartment and you want to remove most of the walls, but you do not know which walls are load bearing. . .

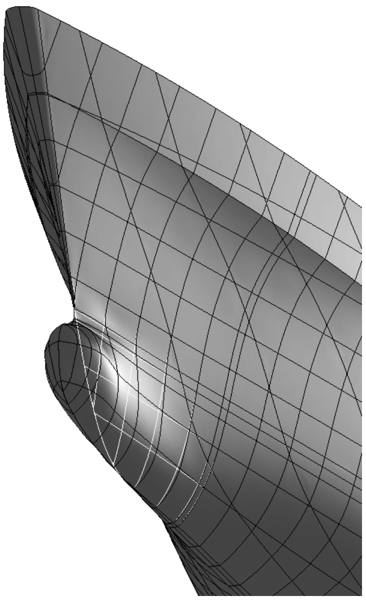

(a) Initial shape.

(b) Stem curve re-designed.

(c) Middle water line drawn out.

(d) Lower water line adjusted.

Figure 7.2: The construction of a bulbous bow by means of curve based shape variations. The original shape is blended in as a white wire-frame.



(e) Extra curves remedy data scarcity.

Figure 7.2: (Continued.)

This example demonstrates that with the proposed method, the number of curves and curve intersections in an H-rep model are really no hindrance anymore to make radical design changes. As a designer, you are free to change your mind any time you like.

7.1.2.1 Alternatives

This kind of re-design would have been very hard with FFD. Any bulb produced by means of FFD would probably have a coincidental shape, there is by no means the precise control over the shape of the individual curves that we have used here.

With traditional methods, the situation is much more manageable than the global variation from the example before. One would probably remove the inner buttock line and remove the parts of the curves that are now white in Figure 7.3(e) above. For this, one would split frames and water lines at a suitable place. In hindsight it is easily determined where to split curves, but without a given bulb shape it may involve more guessing. Now the stem curve can be re-designed as above. Then one would extend each curve, one by one, to its original length and simultaneously manipulating it to its proper shape, probably starting with the lower two water lines. In this process, the shape of the resulting bulb will not be immediately apparent, and intermediate OpenGL renderings may be necessary to guide curve manipulations.

7.1.3 Change of the Aft Keel

Shape variations based on curves are also suitable for changes in larger parts of the design, as Figure 7.3 below illustrates. This variation was based on the rear half of the keel line. The objective of such a change could be to shift the length centre of buoyancy forward, or to improve the inflow to the propeller.

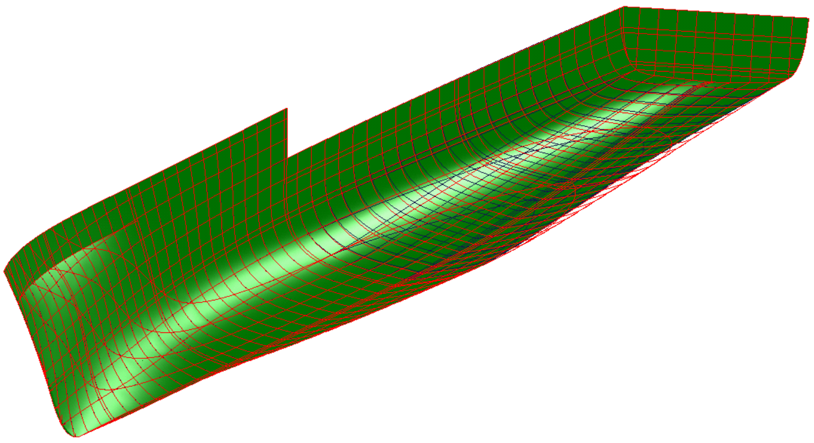


Figure 7.3: The aft body, adjusted by a single variation based on a manipulation of the keel line. The original shape is shown as a red wire-frame, the varied shape shaded in green and a dark blue wire-frame.

In this case, we wanted to preserve as much of the character of the shape in transverse direction, for which the selection field was made transversely uniform. This was accomplished by considering the 2D distance between points, taking only length and height coordinates into account. This is analogous to projecting the points on the centre plane before calculating their distance.

7.1.3.1 Alternatives

This is a variation that could have been accomplished with FFD. By projecting points as described above, the deformation could essentially be performed in 2D, reducing the number of degrees of freedom. Curve based directly manipulated FFD [Gain, 2000] could have been used to re-shape the keel line, although not at the level of precision of our approach. Gain [2000] claims that curve-based directly manipulated FFD is one or two orders of magnitude more efficient than methods that are based on the shortest distance between points and a curve, depending on the method used (functional composition and degree reduction, or curve sampling and ordinary direct manipulation [Hsu *et al.*, 1992]). Still, FFD does need the definition of a lattice, and the level of detail in the variation is governed by the density of the control points in the lattice.

To apply this change by traditional H-rep modelling would be very costly. As usual, one would remove large numbers of curves, adjust the keel line and adapt remaining curves one by one to restore the network consistency. A correct shape and surface fairness in this region would largely have to be built up from the ground.

7.1.4 Increase of Curvature in the Bilge

To drag the bilge of our example model outward so its curvature increases, requires a bit preparation. The straight-forward approach of basing a selection field on a frame at half the ships length, then designing a new bilge shape, and give an extent of the selection field so large that it covers about the full length of the hull, will not produce the result that we are after. Because the frame is defined in a vertical transverse frame, the extent of the selection field will propagate in a horizontal direction. However, the bilge (which for the occasion can be defined as the area where frames have the highest curvature) travels upward in the aft ship. This is best seen in Figure 7.3 on the facing page. So the selection field will not cover the bilge correctly.

Better is to project a new curve on the hull, that tracks the bilge along the length of the ship, and to base a shape variation on that curve. This is the approach taken in the variation displayed in Figure 7.4 on the next page. The curve on which the variation was based can be seen in red, just below the second water-line in the fore-ship (it does not extent to the very front) while it moves up above the third water-line in the aft ship. To vary the shape, this curve was pulled outward slightly, approximately in the direction of the viewer.

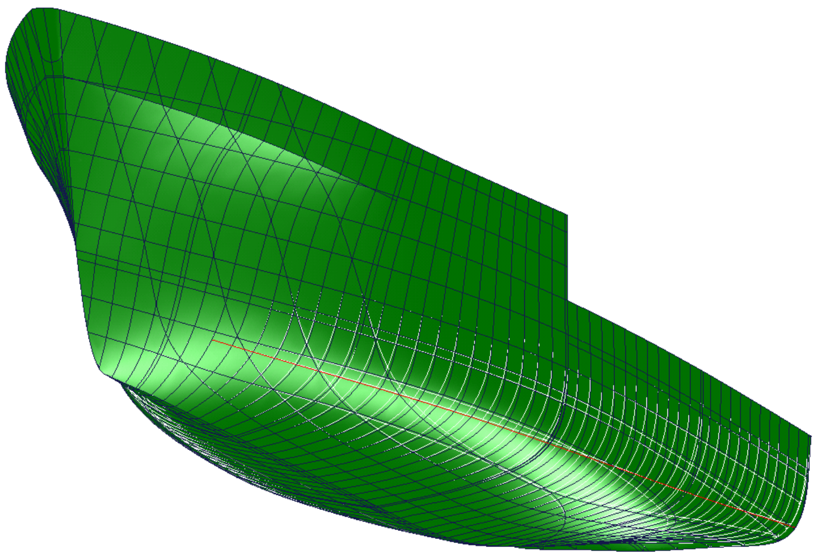


Figure 7.4: The curvature in the bilge has been increased by a variation based on a curve that was projected on the hull for the occasion. The original shape does shimmer through as a white wire-frame.

7.1.4.1 Alternatives

This is a variation not easily accomplished with FFD. To address the issue of the changing height of the bilge, one would probably have to design an initial shape of the lattice that tracts the bilge nicely [Coquillart, 1990]. The consequence is that the mapping between Cartesian coordinates and coordinates in the parametric space of the deforming volume becomes non-linear, which involves a costly search for the parameter values that correspond to the position of the data points.

In defining the deformation, it will not be obvious whether the resulting shape is fair along the length of the ship. During the curve-based shape variation of above, it was easy to check the fairness of the changed curve by plotting its curvature during editing. If the curve on which the variation is based is fair both before and after the editing, the typical shift vector will vary smoothly, resulting automatically in a fair shape variation.

Curve based directly manipulated FFD [Gain, 2000] does allow to check the curvature of the destination curve in a similar way, but whether the concept of direct manipulation is readily transferable to lattices of arbitrary shape is not explicitly stated in the consulted literature. In addition, because curve based directly manipulated FFD is approximative, it may leave flaws in the fairness of the resulting shape.

When left to the means of original H-rep modelling, this is the kind of design change that one very much would try to avoid. Because about half of the hull area is affected, a change like this would involve very much re-designing, checking and re-fairing. Likely, a large part of the shape definition would vanish in the process, and therefore it is very well possible that the end result would differ in more than one respect from the original shape.

At the same time, a change of this kind is interesting with respect to internal space, displacement, static and dynamic stability, resistance, behaviour in waves etc. For optimisation of any of these qualities it is very valuable to be able to explore such variations on a design.

7.2 Limitations

The proposed method of shape variation by means of decay functions is not a panacea. There are limitations and consequences of its application.

7.2.1 Loss of Constraints

Fairway allows the specification of several kinds of shape constraints.

1. Specification of **curve type**, as discussed on page 63.
2. Prescribed **end conditions** of tangency and curvature.
3. **Master/slave relations** between curve sections, as discussed on page 65.
4. "**Developability**" of the surface.

The presented method shifts all data points in the selection field according to the decay function, disregarding the constraints that were defined on the curves to which they belong. The shape of decay functions is in general not compatible with these constraints. In addition, the presented method pulls some or all curves out of their plane of definition, by which all affected curves are converted into unconstrained space curves. The planarity of the curves may be restored as discussed in Section 6.4.9, but the constraints are lost and possibly violated with respect to their original value.

Application of FFD would have the same consequence, as the conflict between these local shape constraints and global smooth shape manipulation is fundamental. To the author's knowledge, parametric design of hull shapes [Abt *et al.*, 2001] is the only method for variation that preserves these features. However, this allows only variation of parameters that have been built into the design, such as bilge radius and specific angular and length measures, producing a particular family of shapes.

Regarding master/slave relationships, when there are few curves in the model, it may be possible to restore the constraints without consequences for surface fairness. For example, if the knuckle line in the bow of our example hull is defined as an offset to the deck line, the curve can be disregarded during the variation and regenerated afterwards by the rules of the master/slave relationship — unless the curve is intersected by other curves, with which it would no longer be compatible. A recommendation that keeps the number of curves low is given later in Section 7.5.1.

The developability of a surface deserves special attention. It is an important shape feature for low-cost production of shell plating. A developable surface has curvature only in one direction, so steel plates can simply be bent, without applying strain along the edges. Conic surfaces are the only developable surfaces [SARC, 2004]. Here the radius of the cone need not be constant, and the top of the cone may move along a space curve. The top may even be infinitely far away, by which the cone turns into a cylindroid.

Fairway supports construction of developable surfaces. The most general way of doing this is by specifying two defining curves, which must be knuckle lines, in between which the developable surface spans. A property of this surface is that from any point on one of the defining curves, there is always one *linear* path over the surface leading up to the other defining curve. These paths are called rulings.

Shape variations assume that every data point in the model takes part in the definition of the shape. For the case of developable surfaces, only data points belonging to the defining curves take part in the definition of the shape. So if there are other curves present in this surface, such as frames or water-lines, a shape variation will result in an over-determination of the shape, by which it will no longer be developable. To correct this, internal curves in developable surfaces must be disregarded, and the developable surface must be regenerated (giving new rulings) after the shape variation. Again, the general recommendation that will be given in Section 7.5.1, includes this solution.

7.2.2 Introduction of Discontinuity

Assuming the objective is to change the curvature underneath the knuckle line in the bow of our example ship hull, one might be tempted to base a selection field on a frame, somewhere half way along the knuckle line. This strategy will give rather disappointing results, as illustrated in Figure 7.5 below (the change is slightly over-done to emphasise the effects).

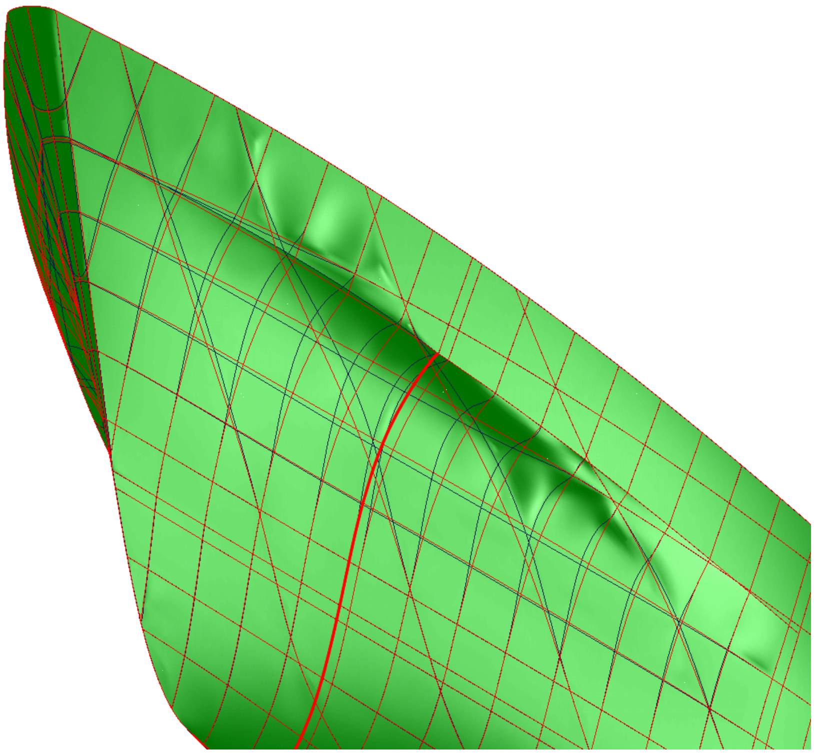


Figure 7.5: Distortion caused by a change in tangent at the end point of the curve on which the variation is based (bold red curve) while the end position did not change. The original shape is printed in a red wire-frame overlay.

The reason is a first order discontinuity in the vector field of shift vectors. This arises when the end point of the curve, on which the selection field is based, is not changed, while the tangent at that point is. So even if the field of influence is G^2 (by means of a G^2 decay function), the vector field, which is the product between the decay function and the typical shift vector, is first order discontinuous if the typical shift vector varies discontinuously in its first derivative. In this case, the length of the typical shift vector goes to zero at an angle.

For a G^2 vector field, the typical shift vector should decrease to zero in a smooth way, in the same way as the decay function, or the typical shift vector should not be zero at all at the end point. In the latter case, the vector field extends beyond the curve, and the decay function takes care of a smooth decay of the vector field. This mechanism can be seen in action in Figure 6.12 on page 112.

The first order discontinuity in the vector field appears in a plane containing the end point, to which the tangent of the curve, on which the variation is based, is normal. We would get the desired variation if this plane contained the knuckle line. However, that is not the case. Upon close inspection of Figure 7.5, we can see that aft of the curve on which the variation is based, frames start to change shape at some distance from the knuckle line. In front of the base curve, there is variation above the knuckle line. The bulging in the surface is caused by the fairing algorithm, that attempts to fit smooth curves through sequences of data points that are essentially not smooth.

To eliminate the bulging, one could connect the discontinuities in the point sequences by a second knuckle line, if that is the design intention. In the cases that it is not, users can be protected from making this naive mistake by an intelligent user interface that does not allow to change the tangent at curve ends unless the ends themselves are moved as well.

Another situation that is capable of introducing discontinuities is when the selection field self-intersects. Curve based selections are in theory capable of self-intersection. What happens then is that two adjacent data points can have their corresponding closest point on the curve at completely different places. Likely, the typical shift vector will be different at these places, causing the discontinuity. Self-intersection happens when the extent of the selection field is locally greater than the radius of curvature of the curve on which the selection is based, or when otherwise two sections of the same curve come too close. Fortunately, in our application, both these situations are very uncommon.

7.2.2.1 Alternatives

Application of FFD would suffer from the same limitation, unless the lattice can be positioned in such a way that it borders exactly on the knuckle line. With traditional H-rep modelling one would proceed in the usual way, removing many curves, manipulate a few, and repair the damage. Again, the recommendation made in Section 7.5.1 will probably solve this particular case.

An alternative application of the method presented in this thesis, with more advanced constraints, may produce a better result. For this we would use a variation based on a point below the knuckle line. We would base a de-selection field on the knuckle line itself, with a decay function that has a cusp at the top ($\kappa = 0$). Hopefully, this would result in a desirable shape below the knuckle line. To prevent the hull above the knuckle line from changing, we would in addition need to constrain all data points that are positioned above the knuckle line. Unfortunately, time did not allow implementing this particular functionality, so this configuration could not be validated.

7.3 Relevance

The alert reader may have noticed that the examples that are used to test and illustrate the proposed method, work on models with a high density of curves. Most of these curves are not actually important for the shape definition, they have only been interpolated for one or more of the following reasons:

1. shape visualisation,
2. to obtain a sufficient accuracy in the computation of hydrostatic data*, or
3. for production purposes, representing the profiles of structural members or contours of shell plating.

In fact, the design of the shape has been carried out using far fewer curves. So, if the design needs to be changed, why not temporarily remove the curves that have not been modified since their creation and go back to this coarser network to apply the change? The chance that intersections with other curves are lost is much smaller then. And if intersections do turn into crossings, their number may be small so that they can be restored by hand without too much trouble.

Of course, this is a correct observation. The examples work on the full collection of curves, as a demonstration that it is capable of preserving intersections also in dense networks. Indeed, if we would only consider curves that have been manipulated since their creation, the responsiveness of curve based shape variations could be greatly increased, because the number of data points that we have to compute

**Fairway* bases the computation of hydrostatic data on the frames in the model, i.e., on vertical transverse intersection curves [SARC, 2004].

shortest distances of would in general be several orders of magnitude smaller. So the above strategy is recommended, an the proposed method can be used to effectively maintain the curve intersections that remain.

At the time of this writing, *Fairway* does have rudimentary support for the removal and re-insertion of curves at several pre-selected locations simultaneously [SARC, 2004]. But the selection has to be defined by the user explicitly with an alpha-numerical user interface, and the insertion or removal is performed without consideration of whether curves already exist or not, and whether they have been manipulated or not. Thus, this feature is capable of throwing away valuable work unintentionally. As a protection, it is possible to "lock" individual curves, but that has to be taken care of explicitly by the user. A better approach is recommended in Section 7.5.1.

7.4 Novelty

As indicated in the introduction (page 1), one can try to be innovative, but the chance that somebody else has had the same idea is very real, and it happens all the time. In this section we will have a look at what elements of the proposed method can be found in the literature, and what elements seem to be new. We will also look at why some established approaches were not adopted.

7.4.1 Decay Functions

The concept of decay functions is very old [Parent, 1977] and the use of B-spline basis functions for this purpose has been proposed at least a decade ago [Borrel and Rappoport, 1994], albeit in the context of FFD. The latter involves an upwards scaling of the function, so it has a top at 1.0, and a re-interpretation of the knot vector. An explicit expression of this function in power basis, as given in (6.7) on page 108, has not been encountered in the consulted literature, and neither has the value of being able to vary the shape of the decay function this way been emphasised much.

The traditional application of decay functions is in sculpting of polygonal models. The application to shape variation of models that interpolate a network of curves is novel.

7.4.2 Curve-Based Variations

Consideration of the shortest distance of points to a curve, and subsequent manipulation of the curve, has been applied in axial deformations [Lazarus *et al.*, 1994; Singh and Fiume, 1998]. However, these references use no special mapping between parameter values of the original curve and the edited curve, i.e., the two parameters run completely in parallel. In our application, arc-length parameterisation was found to be essential when re-designing curves in the H-rep. On the other hand, the twisting and radial scaling that these references provide was considered to be of no practical value to our application.

As an alternative to varying the extent of the variation as a scalar value along the curve, an implementation may consider the use of domain curves [Singh and Fiume, 1998]. Domain curves give direct control over the extent and also allow a different extent on either side of the curve. However, as the "side" of a 3D curve is not well defined, this requires the use of a heuristic, and it is unclear how well the heuristic that Singh and Fiume [1998] give performs on highly sculpted surfaces.

7.4.3 Interaction between Vector Fields

When more than one point-based variation is applied simultaneously, and the base of one is covered by the field of influence of another, Borrel and Rappoport [1994] solve a small constraint system to make the variation obey the typical shift vectors exactly. In this thesis, solving a constraint system is considered as unnecessary. Apart from the fact that constraint solving can lead to space tearing (see page 82), typical shift vectors can easily be changed interactively and the effect shown in real-time to produce the desired shape variation. However, a production-grade implementation of the method may include this functionality as an option.

When using curve-based variations, it is more elaborate to change the typical shift vectors of more than one selection field interactively. The constraint problem is not that straight-forward anymore either. An option is to average the influence of the fields, according to Singh and Fiume [1998]. However, in our application, the practical value of a variation due to the editing of several (intersecting or otherwise closely positioned) curves simultaneously is questionable, and has not been implemented.

Interaction between vector fields in general, which in this thesis is taken as the vector sum, has not actually been evaluated other than in

Figure 6.8 on page 105.

7.4.4 De-Selection Fields

The problem of constraining the model to the plane of symmetry, when only half of it is being modelled, has not been encountered in the surveyed literature. The application of de-selection fields for this and other purposes seems to be a new invention.

7.5 Recommendations

The following is a collection of recommendations for a production-grade implementation of the proposed method, as well as for a general implementation of the H-rep concept as compared to the implementation presented by Koelman [1999]. The increased complexity that these recommendations impose on computer programming, is easier to deal with when object-oriented programming (oop) is adopted. See Section A.1.1 for a discussion.

7.5.1 Data Management

As an improvement to the management of curves as described on page 142, I would recommend a more robust and more efficient approach, like the following. Upon creation of a new curve, the curve is not immediately added to the H-rep data structure, but starts its life in a collection of automatically updated curves. In other words, these curves do not participate in the definition of the surface. The shape of each curve in this collection is automatically re-computed, only when at least one of the surface patches changes, on which the curve in question is based.

When the user is about to select a curve to make changes to the surface, curves that are already in the H-rep are preferred, in order to not increase the amount of shape defining data unnecessarily. This can be accomplished by promoting shape defining curves by special highlighting. The user is still free to choose a different curve instead, one that is not yet in the H-rep, by which that curve is automatically moved from the collection to the H-rep, so that it starts taking part in the shape definition of the surface.

An advanced application of this principle could even consider only the parts of the curve that are manipulated. If, e.g., a waterline is selected from the collection of unmodified curves, which is subsequently manipulated for only a third of its length, it could be added to the H-rep data structure for just that length, only minimally extended to make it start and end at intersecting curves.

7.5.1.1 Advantages

Apart from the positive effect on the freedom to make changes in the design and on the responsiveness of the proposed curve-based variation operations, there are several other advantages of the above approach. To begin with, there are the already indicated positive effects on the matter of master/slave relationships and developable surfaces. In addition, this recommended way of data management will improve curvature continuity, reduce the number of artifacts in the surface and result in a smaller file size.

Curvature Continuity Since the transfinite surface patches remain larger, larger areas of the surface are G^2. After all, with every new subdivision by an added curve in the H-rep, geometric continuity across that curve is reduced to G^1. However, the "almost curvature continuity" (ϵG^2) across patch boundaries, induced by the crossing curves, will decrease as the patch boundaries become longer, and the constraining intersections fewer and separated further apart. When good ϵG^2 is needed for, e.g., manufacturing of a physical model, this effect can be easily countered by adding all curves in the collection to the H-rep. By still keeping the bookkeeping about whether curves have been manipulated or not, this action can also be reversed.

Another measure is to construct G^2 Gordon surfaces over quadrilateral patch compositions, for which an algorithm is presented by Koelman [1999]. Originally this was supported in *Fairway*, but support has been removed because of little gain* and computational expenses [Koelman, 2002]. However, as the recommended approach would lead to fewer and bigger patches, the expenses will be much lower than previously, and the gain will be higher as it will convert most of the patch connections from (decreased) ϵG^2 into true G^2 connections.

*The gain has been evaluated primarily with respect to the interpolation of new curves. The difference between curvature continuity and discontinuity across patch boundaries is in general only apparent close to the boundaries. Since curves are generated to interpolate a finite number of points on the surface, this only affects a small number of these points in a marginal way. Of course, the curves themselves are G^2 per definition, and the differences are likely to be reduced courtesy the fairing algorithm. So indeed, the gain of G^2 patch transitions can be neglected as far as curves are concerned. However, when the focus is on the surface, e.g., in rendering photo-realistic images or production of a physical model, curvature continuity becomes more important.

Fewer Artifacts When more than two curves in the H-rep intersect in (almost) the same point, shape artifacts can occur. These artifacts appear as ripples or undulations in the surface and are caused by un-proportionally spaced tangent data that the tangent ribbons interpolate smoothly, together with relatively large changes in these data [SARC, 2004]. The same phenomenon exists for positional data, as illustrated in Figure 6.20 on page 125.

Ripples can also appear in relatively flat regions of the surface, as seen in Figure 7.1 on page 129, which suggests that this larger change is caused by curves not passing exactly through the data points — which is a compromise for curve fairness. This imprecision may be microscopic in absolute sense, but if three curves intersect almost in the same point, a microscopic triangle is created, so the tangent data (at the corners of the triangle) is also spaced microscopically tight. So, relative to the spacing of tangent data, this imprecision becomes large enough to produce visible ripples. Currently, *Fairway* can be instructed to generate tangent ribbons by linear interpolation, which reduces the rippling at the cost of a little smoothness.

The recommended approach will reduce the number of these artifacts by leaving most of the curves out of the H-rep data structure. Remaining artifacts can be handled by the approach proposed in Section 7.5.2 below.

Smaller File Size File size is probably seldom an issue, as the current format is already economic compared to explicit surface methods. Nevertheless, the above recommended approach will reduce the file size even more, especially for production-ready designs, since for curves that were never manipulated only their position needs to be saved.

7.5.2 Elimination of Artifacts

What follows is a proposal to eliminate artifacts, as seen in Figure 7.1 on page 129, by disregarding limited curve sections. Consider Figure 7.6 on the facing page, which shows a detail of a curve network. A surface artifact can result from the dense spacing of tangent information at the corners of the tiny triangle in the middle of the detail (Figure 7.6, left), together with an imprecision as explained above.

The density of tangent data can be reduced to an ordinary level, eliminating the artifact, by locally disregarding one or more curves. Thereby, tiny surface patches essentially vanish. This can be accomplished by splitting excess curves at nearby intersections on both sides of the problem area, and removing the offending curve section from the H-rep data structure (Figure 7.6, right). On a higher level, the curve can (and should) still be defined as one continuous curve.

Assuming that non-contributing curves have already been taken out of the H-rep data structure, in accordance with the recommendation made in Section 7.5.1, it follows that disregarding any curve section will affect the shape of the surface to some degree. However, since the curves pass (almost) through the same point, the amount by which they individually contribute to the definition of the local shape is limited. It is reasonable to expect that the surface will only be affected marginally by removing a section from the data structure. Nearby curves will cover for its absence. And after all, we were not happy with the shape in the first place.

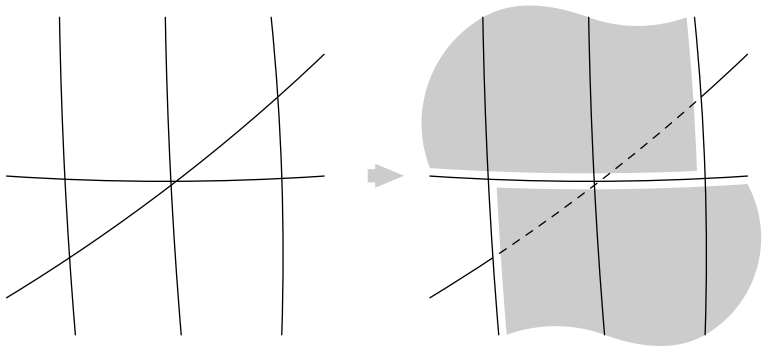


Figure 7.6: When more than two curves intersect in almost the same point (left), artifacts can be eliminated by disregarding limited curve sections (right). The shaded areas indicate the possible extension of quadrilateral mesh cell complexes over which Gordon surfaces can be constructed. Alternatively, a Gordon surface may be constructed over the full height of the figure, in-between the three vertical curves, covering the problem area.

There are of course several criteria that can be used to select the curve or curves that are to be disregarded this way. Selecting the shortest possible section will probably minimise the effect on the surface shape. Other possibilities are to make a selection such that the remaining curves are as orthogonal as possible, or such that the quadrilateral area that can be covered by Gordon surfaces is maximal (shaded in Figure 7.6).

7.5.3 Aspects of Efficiency

Shape variations with selections based on points and planes are fast. However, curve-based selections can be time consuming, because computing the global closest distance of a point to a curve is expensive. In the current proof-of-concept implementation, all data points in the entire model are processed. Depending on the length and complexity of the curve, the time needed for distance computations in the examples of Figure 7.2 on page 132 took up to several minutes per variation on a 1.5 GHz PC. Once the distances are computed, shift vectors can be previewed interactively while the extent r and shape parameter κ are varied.

This performance can be improved. Firstly, it may be possible to omit several iterations in the closest point finding algorithm per data point, if the loop termination is not based on the accuracy of the closest point, but on the accuracy of the resulting typical shift vector. Secondly, the recommendation made in Section 7.5.1 will cut down greatly on the amount of data points to be considered. And finally, the shape variations that are discussed here, rarely involve every single data point in the model; in case it does, an ordinary affine transformation probably performs better. So computing the distance for all data points in the model is a waste of time. It would be much better to consider data points on demand, based on the extent of the selection. For this a graph search is required, similar in nature to the algorithm presented in Figure 6.18 on page 120.

7.5.4 Open Interface

A non-standard surface representation hinders a large scale acceptance in industry due to the limited possibilities for the exchange of surface models. Without a standard, the realisation of well supported data exchange requires cooperation and good-will of a large number of actors on the CAD market. To stimulate support for this foreign surface representation one could try to make it very easy and virtually cost-less to create an interface for the evaluation of H-rep models.

For this, a public data file format is not sufficient, as evaluation of the model will require the implementation of a considerable number of algorithms, an effort that cannot be expected from many. A better chance for acceptance is to release the algorithms as well. There is no secret in this as the algorithms are documented in the literature, so releasing in object code alone (which is machine readable only) serves no particular purpose. In stead, the algorithms can be distributed as a library of source code (of which humans can make sense as well). This makes integration into third party software easier, allows a higher system efficiency because data conversion layers can be omitted and creates good-will. It also relieves us from the need to support object files for a high number of different computer architectures and operating systems. In addition, releasing source code requires much less support, as third parties that are having trouble making the code work can help themselves.

There is a misconception in the corporate world that releasing source code is the same as giving away assets, from which money can be made, to your competitors. By publishing code, one does not give away ownership, and copyright still applies. What others are allowed to do with the code depends on special exceptions to copyright as stated in a license. Contrary to the customary software licenses that accompany commercial software, which explicitly limit the freedom of the user, our license would state freedoms that copyright would otherwise prohibit, and therefore we need not worry about a user's acceptance of it. The Lesser General Public License (LGPL), as defined by the Free Software Foundation, is a popular license that can serve the proposed purpose [Free Software Foundation, 2000]. This license has the advantage that any optimisations, improvements, bug-fixes and extensions that third parties apply to the code, flow back to the owner of the code (and everybody else) by obligation. This principle makes the software largely self-sustaining.

Currently, to make this policy work, it would require exemptions of certain patent claims [Koelman, 2004]. To prevent abuse of intellectual property, one could selectively license the patent for evaluation of H-rep models but not for modification of the data structure. As a side effect, a publicly available implementation of the algorithms increases the marketability of complete patent licenses for a fee.

With this policy, what we may see in the future is support of the H-rep concept in applications that "use" hull forms for further design and analysis, such as the following.

- Hydrodynamical analysis by means of CFD, e.g., of sea loads and motion in waves.
- Structural design.
- Structural analysis.
- Piping etc.
- General arrangements plan.
- Machine room design and evaluation.
- Other aspects of detailed design.

- Photo-realistic rendering of presentations.

These third party programs could use ordinary CSG, which involves trimming by curves to disregard portions of the surface, and do not need to modify the H-rep data structure*. With respect to ship hulls, this allows for example the design of a bow thruster tunnel, the opening bow of a roll-on-roll-off passenger ferry, appendages such as stabilisers, the rudder, propeller shaft struts, thrusters etc, as well as smaller openings for intakes and outlets.

7.6 Two Paradigms Revisited

Consider once again two paradigms for the definition of surfaces with arbitrary topology. On the one hand we have approximation of arbitrarily connected points, and on the other hand interpolation of arbitrarily intersecting curves.

With the possibility to change only one data element at a time (control point or curve, depending on the paradigm), the definition of the shape becomes more rigid with the amount of defining data. This is a logical consequence, but it is much more severe for the interpolating paradigm. For the approximation paradigm, the reduction in freedom to do larger-area shape manipulation is linearly dependent on the number of data elements: with every added control point there is one more point that has to be positioned in harmony with the rest. For the interpolating paradigm, this freedom reduces at a much higher rate, relative to the number of defining curves in the model. In the worst case, if there are n defining curves, then one extra curve can add n new constraints in the form of curve intersections.

With the proposed method of curve-based variations, these constraints have essentially been eliminated for the interpolation paradigm, as curves can be re-designed no matter how dense the curve network is.

Thus, the interpolation paradigm has been brought on a competitive level with the approximation paradigm, by which the paradigms are better comparable. To be fair in this comparison, one should allow point-based variations on the (arbitrary) control polyhedron of surface splines or subdivision surfaces, which allow for larger-area shape variations. In the following, we will compare the two paradigms from an engineering point of view with respect to surface quality, practical

*Since recent times, *Fairway* supports boolean operations on H-rep models. However, its approach is different from CSG, as it modifies the data structure, and no history is preserved, i.e., there is no CSG-tree.

usability and complexity of the underlying method, with the proposed method and recommendations taken into account.

7.6.1 Surface Quality

With respect to surface quality, we are interested in curvature continuity and in the absence of artifacts. Catmull-Clark subdivision surfaces (Section 3.1.1) are G^2 everywhere, except at extraordinary points, where unbounded curvature can lead to visible shape deficiencies. The conversion of Catmull-Clark surfaces to surface splines (Section 3.2.2) improves on this, using standard bi-cubic NURBS patches to form a surface that is G^2 where the control polyhedron is regular. At extraordinary points, the surface is G^1, with bounded curvature and no artifacts. At the cost of constraint solving and higher order polynomials, surface splines can produce completely G^2 (or even G^k) surfaces.

On the other hand, H-rep surfaces are G^2 only over regular quadrilateral regions of the curve network (when Gordon surfaces are applied), and G^1 across curves elsewhere. However, at intersection points, the surface is G^2 because of the continuity in the curves. Artifacts can still appear when tangent ribbons wobble to interpolate tangent data. Usually, these can be corrected by adding curves to (or removing curves from) the data structure, but it nevertheless requires careful inspection.

In the current state of the art, we must conclude that the approximation paradigm is capable of achieving higher surface quality than the interpolating paradigm. However, this is primarily due to surface splines, which are not yet widely applied in industry.

7.6.2 Practical Usability

With respect to practical usability, we are interested in surface manipulation and in the ability to describe a particular shape. At the time of this study, there was no access to an implementation of the approximation paradigm for the design of surfaces of arbitrary topology, and therefore a formal comparison of the usability of the two paradigms was not possible. Nevertheless, we can discuss the differences at a conceptual level.

One has to be careful in making statements about the differences in surface manipulation between the two paradigms. At first sight, the difference is fundamental, the prime reason to differentiate between two paradigms at all: approximating surfaces are controlled by control points and interpolating surfaces are controlled by curves. Hence the former are manipulated by control point manipulation and the latter by curve manipulation. However, curves themselves are manipulated by control point manipulation — or data point manipulation and subsequent application of the fitting/fairing algorithm. Therefore one might argue that surfaces are manipulated through points no matter the paradigm, and that the curves in the interpolation paradigm serve only as a level of abstraction.

But for usability, abstraction is important. A curve is much easier given a particular shape than a surface. This is especially the case when the curve is planar, so a complete dimension is abstracted away. Planar curves also make conception of curvature along the curve straightforward, as curvature indicators can be plotted perpendicular to the curve, in the plane of definition.

The ability to work with curves is even more important for ship hull form definition. Hull forms are traditionally defined by lines plans, consisting of planar curves of perpendicular intersections. Even today, a shape is much better communicated between experts in the form of a lines plan than in the form of a computer rendering. To work in this traditional format allows traditional rules of thumb for good shape to be applied. Also, existing lines plans can be accurately digitised into H-rep models, which is a valuable feature for two reasons. Firstly, it is common to base the design of a new vessel on a comparable existing design. With the ability to digitise and import lines plans, the lines plan becomes a platform-independent format for shape definition, which means that there is a much larger set of hull shapes to choose from than the limited set of hull shapes that has been designed in-house on the platform that is currently in use. Secondly, when vessels are enlarged or converted to serve a different purpose, "sponsons" may have to be welded to the hull in order to obtain sufficient stability. For the sponsons to fit the hull like a mould, the shape of the existing hull must be available in digital form when the sponsons are designed. This can easily be accomplished in the interpolation paradigm as the shape can be extracted from the lines plan or even from structural plans. The latter ensures that the stiffeners in the sponson will match the stiffeners in the hull precisely. This advantage counts for other appendages as well, such as a bulbous bow.

Accomplishing surface fairness in the interpolation paradigm is accomplished through the fairing of curves. Although the fairness of a curve is easily interpreted and improved, fairing a curve in isolation does not result in a fair surface per se. Firstly, there are implications for the curves that intersect the curve that is being faired. The fairness of these curves may actually degrade as a consequence. And secondly, even when the curves are fair and the network consistent, fairness of the interpolating surface can only be guaranteed along the curves. In between, shape deficiencies can occur, e.g., by un-proportionally spaced tangent data such as in Figure 7.2(d) on page 132.

In the approximation paradigm, control points need not be lined up as they are in the interpolating paradigm, where they define curves. Since they can be positioned and ordered according to shape requirements exclusively, we can posit that surfaces following the approximation paradigm in general can do with fewer control points than surfaces that follow the interpolation paradigm. This means that fewer control points need to be positioned in harmony to obtain a fair surface. According to this postulate, achieving surface fairness is easier in the approximation paradigm than it is in the interpolation paradigm; provided that an effective method for surface curvature interrogation is available, such as an interactive Zebra plot and colour renderings of the curvature.

Although approximating curves, such as B-spline curves, are regarded by some as un-intuitive to work with, others firmly believe that manipulation of the control polygon is the best way to obtain aesthetically pleasing curves. This belief is extended to surfaces as well, as in the following quote.

> "Not surprisingly, averaging [(the approximation paradigm)] results in more pleasing surfaces than curve interpolating algorithms [...], and one may wonder what other than the tradition of Coons construction attributes the curve mesh such preference when constructing a surface." [Peters, 1995a]

We may conclude our discussion that the interpolation paradigm is better suited if an exact surface shape is required in a limited set of well-defined planes, such as often is the case in engineering applications. On the other hand, when aesthetics are more important than tolerances, the approximation paradigm provides a better modelling interface.

7.6.3 Method Complexity

The abstraction into curves, however practical in one respect, is also a simplification. This simplification has consequences so severe that it is possible to take a PhD on the subject. In the hierarchy of geometric elements, curves are positioned on a lower level than surfaces. Curves can easily be extracted from a surface, but the construction of a surface from curves is fraught with ambiguity, inconsistency and limited accuracy*. The H-rep concept has a high level of pragmatism, brought forth by an engineering approach to geometry. The contribution in this thesis is no exception. A state-of-the-art implementation of the interpolation paradigm is a complex system with an extensive array of features. Among these are construction of curves of various types, surfaces of different definitions for different numbers of sides and continuity, a non-trivial data structure, fitting and fairing of curves, graph traversing algorithms and a number of hacks and work-arounds to counter limitations.

A much cleaner approach is taken in the approximation paradigm. Subdivision surfaces are governed by a small set of simple rules, and the theory of surface splines is firmly based in mathematics. Approximating surfaces are precise, consistent and unambiguous— that is, when considering the advances discussed in Chapter 3, not the current state in the industry.

7.6.4 Conclusion

In short, the difference between the approximation paradigm and the interpolation paradigm in the current state can be summarised as follows.

Solutions for the modelling of shapes with arbitrary topology that follow the *approximation paradigm* seek elegance in their approach to the problem. They are mathematically sound and firmly based on theoretical foundations. They are well suited to model aesthetically pleasing shapes without strict geometrical requirements. Subdivision surfaces are widely applied in the entertainment industry.

Solutions that follow the *interpolation paradigm* for the modelling of shapes with arbitrary topology consist of an assembly of parts with various functionality, that have been engineered to work together as a complex system. The resulting machine is not pretty, but when operated correctly, it works for the tasks that it was built for. The

*In the implementation of Koelman [1999], a network of intersecting curves is accepted as consistent if curves intersect within a limited accuracy. By zooming in on the intersection far enough, curve intersections can in general be identified as being actually curve crossings. The accepted inaccuracy is user-definable and can be set below production tolerances. The inaccuracy is a trade for easier curve fairing, but also a source of surface artifacts as discussed before. Alternatively, the implementation of Michelsen [1995] features exact intersections by means of interpolating spline curves, which are however more difficult to fair.

shape can be controlled exactly in selected planes, a feature that is desirable when designing plate constructions. The approach fits well with the traditional method of ship hull form design, and currently enjoys modest use in the maritime industry.

In one sentence, one could say that the elegance of approximation lies more in its theory that in its practice, and that the elegance of interpolation lies more in its practice than in its theory.

7.7 Chapter Summary

Point-based and curve-based variations have shown to be a versatile tool to bring about a wide range of practical design variations in free-form shapes of arbitrary topology. Curve-based variations integrate seamlessly with the curve-based surface manipulation of the interpolation paradigm. This relieves the designer from the burden of manually maintaining intersections of curves, which has been a source of distraction in the creative process of design and a major obstacle in the way of shape optimisation and reuse of prior designs. The proposed method allows both radical design changes and the redesign of specific form features, as well as more subtle variations that may cover large parts of the model. We have made suggestions for an effective implementation of the method and how the industry can be stimulated to a wider acceptance of the interpolation paradigm.

CHAPTER 8

Summary and Future Work

In the first chapters of this thesis we have taken a long look at the standard practice of how surfaces are represented in computers, and how the shortcomings of this practice limit the designer in his freedom to actually design 3D free-form shapes on the computer. We have also identified two paradigms of surface representation. One paradigm, which we have termed the approximation paradigm, defines surfaces by averaging or blending discrete control points, by which the surface smoothly approximates a polyhedron of which the control points are the corners. Standard surface representations follow this paradigm. The other paradigm, which we termed the interpolation paradigm, defines surfaces by transfinitely interpolating between boundary curves, by which the surface passes through the curves.

We are not the first to have complaints about the versatility of standard surface patches, and a great many people have looked at the problem. The problem, is two-fold. Firstly, standard patches, which are quadrilateral, can only be mapped straight-forwardly to a small set of geometric objects. The problematic shapes are said to have an arbitrary topology, with which we mean that the mapping would be non-regular or unstructured. The other problem is fundamental and exists in both paradigms, but is amplified by the structure of standard surface representations. This problem pertains to the phenomenon that the "rigidity" of a design increases with the amount of shape defining data. In other words, the shape defining data (points or curves, depending on the paradigm) is generally organised in a certain balance, a harmony, and not only the absolute value (shape, position) of the data is important but its relative value in relation to the rest of the data as well. The "rigidity" results from the need to maintain this harmony, which is jeopardised by manipulation of single data elements.

In chapters 3 and 4 we looked at a considerable number of references that address the first problem within the approximation paradigm and the interpolation paradigm respectively. We observed that the rigidity problem is much more limiting in the interpolation paradigm than in the approximation paradigm. In Chapter 5 we reviewed methods that change not just a single data element but a complete subset of the defining data in a harmonious way, which can help address the rigidity problem.

In Chapter 6 we considered how solutions from the preceding chapters could be combined to form a computer method for geometric design that is free from the problems of arbitrary topology and rigidity. As the interpolation paradigm has particular value for engineering design, and because it suffered un-proportionally from the rigidity problem (meaning that solving it would be most valuable), we selected the H-rep concept to address the topology issue, being the technology that is heading the developments in the interpolation paradigm.

Fortunately, the most important contribution of the H-rep concept to the interpolation paradigm is the integration of a curve fitting and fairing algorithm, with the side effect that the surface is not only defined by curves, but by data points on the surface as well, from which the curves can be derived. This means that the methods for polygonal sculpting and spatial deformation from Chapter 5 can be applied. Among these, the methods based on decay functions and axial deformation have been the most valuable source of inspiration in the development of a solution to the rigidity problems of the H-rep concept.

The research process moved between hypothesis construction on one side and testing and reflection on the other side in the reciprocal fashion of reflection-in-action [Schön, 1983], each move motivating the next. We started with simple point-based variations, continued with constraints and de-selection functions, then variable decay functions, curve-based variations, the disregarding of unchanged curve parts, arc-length parameterisation, the culling of unconnected selections and finally the restoration of planar curves. We also found that the Euclidean distance between geometric elements is a suitable input to the decay functions. Computation of the Euclidean distance of data points does not require evaluation of the surface, which is an advantage to the more advanced parameterisations that were briefly considered. The latter were discarded as being too complicated, expensive, and because of an identified risk of deficiencies in the quality of the resulting shape.

During the evaluation in Chapter 7, the product of our efforts showed to be powerful enough to accomplish real world design changes, and we can conclude that the rigidity problem of the interpolation paradigm has been significantly reduced, if not defeated. However, our approach does have implications for some of the features in the H-rep implementation of Koelman [1999], and it may produce unexpected results when used naively. This was discussed in Section 7.2. Nevertheless, due to its improved condition, the interpolation paradigm is able to better compete with the approximation paradigm, which has motivated a comparison between the two in Section 7.6. We concluded that the interpolation paradigm does not produce the highest surface quality, and that implementations must deal with high system complexity, and that nevertheless interpolation is the preferred paradigm for many engineering design tasks due to its curve-based modelling interface.

The contribution of this thesis can be summarised as an improvement in the freedom of designers, by allowing them to change their mind at any time. For the shipbuilding industry in particular, this translates into the value that hull forms can be optimised further, and where new designs are based on prior designs (a common practice in shipbuilding), results can be achieved faster.

8.1 Future Work

There is room for future work. Besides an industry strength implementation of the presented method and recommendations, it may be possible to extend the functionality of constraints. But the next big step towards the optimal method for CAGD might happen in the other paradigm. A quick look at both these statements will conclude this thesis.

8.1.1 Constraints

Up to now, only constraints were discussed that act on the position that points had before the variation operation, i.e., the departure position. An example is to constrain all points that are on the plane of symmetry to that plane, while they are free to move in that plane. This particular constraint does not prevent points with a departure position in the vicinity of that plane from passing through it; although that would be very unlikely for reasonable typical shift vectors. But in order to rule this out, we should also constrain on the destination position — which could be a future extension.

Constraining single points on the destination position is not difficult, but achieving a smooth transition in the shift of neighbouring points is all the more. If we had that functionality, we could also use it to guarantee that the design does not violate certain maximum main dimensions, such as Panamax or a maximum draught. But the problem of constraining points from moving through the centre plane is different from constraining points to move beyond the maximum breadth. Whereas constraining shape variations from violating the maximum breadth or draught would be meaningful, because it would result in a flat region which is common in merchandise ships (FoS and FoB), it would not be meaningful to have a flat region of the shell in the plane of symmetry, the centre plane. This illustrates the complexity of the problem.

But, before investing resources into the development of this feature, one should consider the following. An automatically defined transition between constrained and unconstrained points may be visibly artificial. As the transition between flat regions and curved regions of the hull are prominent shape features both aesthetically and hydrodynamically, most designers would rather want to design them than to have them automatically generated.

8.1.2 Approximation Once Again

What the interpolation paradigm suffers from most at this point is arguably its limited surface quality. The approximation paradigm provides much better surface quality; in theory, approximation supports up to arbitrarily continuous (G^k) surfaces. For the ultimate free-form modelling approach it may therefore be beneficial to investigate whether a curve-based manipulation technique can be imposed on G^2 surface splines [Peters, 2002a], or at least on patched or exactly evaluated Catmull-Clark surfaces [Peters, 2000; Stam, 1998]. To have as many curve handles on the shape as there are typically curves in the lines plan of a ship hull is utopian I fear, and unnecessary as well. However, a small but sufficient number of feature curves to control the surface may be doable. One possibility is to align a chain of control points in the control polygon with the curve, and either solve a constraint system, or the B-spline interpolating subdivision surfaces of Nasri [1997] may serve as source of inspiration.

For global shape variation there is an array of possibilities. Besides using decay functions to shift nearby control points and other spatial deformations to deform the control polyhedron, there is the possibility of multi-resolution modelling which has been shown to work for G^1 surface splines [Gonzalez-Ochoa and Peters, 1999]. For deformations based on physics, a static equilibrium analysis of a system where the connections in the control polyhedron resemble structural beams [Léon and Trompette, 1995] will provide a most interesting experiment.

Appendices

APPENDIX A

Computer Programming

This appendix contains a discussion on the practical side of implementing an H-rep modelling system that incorporates the method that was proposed in Chapter 6, including the recommendations made in Section 7.5. The discussion is largely motivated by my experiences of working with the source code that was generously provided by SARC, and using the programming language that it is written in. This text is not to be seen as critisism of SARC or its working methods; the design decisions for *Fairway* were sensible ones at the time they were made, and whether or not it is wise to change language and programming paradigm is not at all clear.

A.1 Modern Software Engineering

Software is inherently complex [Booch, 1991, chapter 1]. Over time, different programming paradigms have been developed, together with programming languages in support of these paradigms, to manage ever higher levels of complexity. One of these paradigms is called "structured programming", which allows to build complex systems using algorithms as their fundamental building blocks [Booch, 1991]. The commercial implementation of the H-rep concept that we are using as a reference, follows this paradigm of structured programming, using the language of (Extended) Pascal.

Although Pascal has some particular advantages (e.g., symplicity, strong typing, compilation speed), there is a limit to the complexity that structured programming supports. Since the focus is on algorithms and not on data, the access to data and protection of its integrity can become a problem in larger projects. It requires a very high level of awareness of the programmer about the state and nature of data throughout the program. This has implications for quality, maintainability, extendibility, the cooperation between software engineers in a team and the ability of newcomers to work with the code.

The following discusses newer programming paradigms, and consequences of adopting them.

A.1.1 The Object Model

A newer programming paradigm, referred to as object-oriented programming (oop) or, in more general terms, the object model, introduces several new elements that build upon earlier models. The object model allows the abstraction of data into objects and classes of objects, and generally enables us to handle systems of higher complexity. Booch [1991] gives the following definition of oop:

> *Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.* [Booch, 1991, p. 36]

A clarification of the difference between objects and classes may be in place. In general, a class forms the definition of the structure of some data and how that data can be manipulated. Most classes can be instantiated into "tangible" objects which can be stored, moved, passed around, done things to or otherwise used. Objects from the same class share the same functionality, and differ only in the value of their data.

To give a complete and detailed description of the mechanisms, by which the object model enables us to manage a higher level of complexity, is beyond the scope of this text. There are numerous excellent sources in the literature [e.g., Booch, 1991; Coad and Yourdon, 1991] and on the Internet*. In short, the essence of objects is that they are manipulated through an interface, whilst their implementation (including data) is shielded off from the rest of the system. This has several advantages, such as

*`www.pegasus.rutgers.edu/~elford/class/oo.html`, `www.objectfaq.com`, or `ootips.org`.

1. Classes of objects can be designed, implemented and tested in isolation. They are also maintained and extended in isolation.
2. Objects can maintain their data in a consistent state.
3. Objects are used through a clean interface, and knowledge about its implementation is not needed.

The process of partitioning the problem space this way is called "abstraction" or "indirection", and the hiding of implementation and protection of data is called "encapsulation". Other principles of the object model are "inheritance" and "poly-morphism". Inheritance means that classes can inherit working functionality and properties from other classes. This also improves "code re-use", meaning that things need to be implemented only once. Poly-morphism is the concept that operators and procedures can handle data of various types, leading to consistent interfaces and expressions.

Apart from its ability to break down complexity, the object model can lead to a better preservation of investments.

> "[T]he use of the object model produces systems that are built upon stable intermediate forms, and thus are more resilient to change. This also means that such systems can be allowed to evolve over time, rather than be abandoned or completely redesigned in response to the first major change in requirements." [Booch, 1991, p. 71]

A.1.2 Generic Programming

There is news on the algorithm front as well, with the concept of GP. Generic programming makes it possible to define procedures (e.g., swapping the values in two variables), concepts (e.g., equality) and container classes (e.g., dynamic arrays) that work on data of yet unknown type. Using this functionality, it is possible to write generic algorithms (e.g., for sorting data). Given a template of a generic concept, a compiler that supports GP is able to autonomously implement a specialisation of the concept for any given data type.

Generic programming is probably most frequently used through the application of concepts from third party libraries, such as the C++ standard template library (STL). Interestingly, it may be the container templates for which the STL is best known, and container templates help with the administration of collections of data. Fortunately, the concepts of GP and OOP do coexist very well, and their combination provides an even better means to master system complexity.

A.1.3 Application

Booch [1991, section 2.3] gives 32 fields of application that typically have the complexity to benefit from OOP, and CAD is among them. It is my opinion that an implementation of the H-rep concept would be most successful by means of the combination of OOP and GP, and adoption of the proposed method for shape variation and the above recommendations would have been easiest in such an implementation.

The compiler that is currently in use to compile *Fairway* supports elements of OOP, so a smooth transition from structured programming to object-oriented programming is in theory possible. However, this support is due to a non-standard language extension, which would mean that the future of *Fairway* would be coupled to the future of that particular compiler. In addition, support of OOP in that compiler is still immature, as evidenced by compiler crashes, and GP is not supported. Appendix A.2 on page 170 further illustrates that the current language and compiler are not ideal for programming according to these modern paradigms.

A.1.4 Cost

This leaves the option of re-implementing the system in a different language. Currently, the C++ programming language [Stroustrup, 1997] is the best language for high performance OOP due to its maturity, run-time performance and the availability of third party libraries. However, a naive application of the language can be disastrous, and thorough training is essential. The general expressiveness of the language comes at a performance cost, and when applied to algebraic expressions, this can lead to poor numerical performance — unless highly advanced expression templates are used [Veldhuizen, 1998]. The books of Meyers [1998, 1996, 2001] are recommended sources for serious C++ programmers, featuring a total of 135 rules for effective use of the language and the STL. This implies that C++ can easily be used ineffectively. Wilson [2004] also identifies imperfections of the language, enough to write a 400+ page book about them.

In a few years time, the D programming language [Bright, 2002] may be a better alternative. At the time of this writing, version 1.0 of the D language specification is due shortly, featuring interesting and valuable improvements with respect to C++ and other languages. Amongst other things, it brings compilation time back to the level of Pascal. However, it will take time before third party libraries are abundant.

While learning a new language and implementing a CAD system anew requires high investments already, it will not be enough. As Booch puts it,

> [...] if we try to use languages such as C++ [. . .] as if they were only traditional, algorithmically oriented languages, we not only miss the power available to us, but we usually end up worse off than if we had used an older language such as C or Pascal. Give a power drill to a carpenter who knows nothing about electricity, and he would use it as a hammer. He will end up bending quite a few nails and smashing several fingers, for a power drill makes a lousy hammer. [Booch, 1991, p. 33]

The power that is spoken of above is not the power of the OOP language as such, but of the object model as a whole, covering the full spectre of analysis, design, implementation, testing, and maintenance. Of these, object-oriented design (OOD) may be the most important aspect regarding the success of an application of the object model. Design following the object model is defined as follows.

> *Object-oriented design is a method of [software] design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design.* [Booch, 1991, p. 37]

In other words, it is in design where the complexity of the problem is decomposed into an effective assembly of manageable subproblems. Booch [1991, figure 7-3] recommends to assign about three times as many resources to design as to coding. Analogously, thorough training in OOD may be a couple of times more important than training in OOP, at least in the long run. An "object-oriented programming language is not "just another programming language" that can be learned in a three-day course or by reading a book. It takes time to develop the proper mind-set for object-oriented design, and this new way of thinking must be embraced by both developers and their managers alike" [Booch, 1991, p. 218].

There are a hand-full of other renowned sources on OOD besides the one referred to in this section. All of them propose a graphical notation to describe the structure of a design, but the notations differ.

More recently, a uniform notation has been standardised as the unified modelling language (UML) [Fowler, 2003, see also `www.uml.org`].

Currently, application of the H-rep concept in industry is rather modest. Although a conversion to the object model comes at a high cost, I expect that the object model has to be adopted at some time, if the H-rep concept is to have a prosperous future in the industry. Parties that are interested in contributing in this, should be aware of patent issues [Koelman, 2004].

A.2 Container Class

This container, implementing an array whose bounds expand on demand, was specially implemented to investigate and illustrate the shortcomings of Pascal, the programming language in which *Fairway* is written. Standard Extended Pascal does not support containers other than static arrays.

A static array would have sufficed for the experiments conducted in this work; we then would have allocated an amount of memory that exceeds the requirements for most situations, but too little for some. But, personally I consider the waste of resources and the limitation on capacity unacceptable for an implementation for commercial use — which is the objective if the method shows to be successful.

With the use of non-standard support for OOP of the compiler in use, a self-scaling array was successfully implemented after the model of the C++ STL; although not nearly as efficient. Unlike C++, there is no support for generic programming (GP) in Pascal, so this container can only hold data of the type for which it was designed. For programming tasks like these, I would recommend a language that supports OOP and GP, like C++ or D; but of course container classes are already standard inventory in the libraries of these languages. We note that the container could also have been implemented as a linked list, but it would have had a higher overhead and sorting would have been slower.

The following example shows a possible use, wherein the container is filled with three items using `.append`, then sorted. With `.get(i)` the i^{th} element is retrieved.

```
BEGIN
  {The container:}
  nodes : DynamicRandomAccessContainer;
```

```
   {A handle:}
   current_node : rich_node_ptr;

   {Some things to put in:}
   a, b, c : vertex_ptr; {Some vertex pointers.}
   normal : vector_type; {Some vector.}
   discard : shortreal; {A bit bucket.}

   nodes := DynamicRandomAccessContainer.Create;

   current_node^.node := a;
   current_node^.d := 1.42;
   current_node^.S := normal;
   {In Pascal, you can not call a function as if it were a procedure, discard its return
   value.}
   discard := nodes.append( current_node );

   current_node^.node := b;
   current_node^.d := 0.717;
   current_node^.S := normal;
   discard := nodes.append( current_node );

   current_node^.node := c;
   current_node^.d := 3.14;
   current_node^.S := normal;
   discard := nodes.append( current_node );

   nodes.sort;

   FOR i := 0 to nodes.size-1 DO BEGIN
      {Do something with nodes.get(i).}
   END
END
```

A.2.1 Class Interface

```
MODULE dynarray interface;

{Copyright 2002 Bastiaan Veelo
This file may not be used for purposes that are not related to the PhD research of
Bastiaan Veelo without his prior written consent.
Bastiaan.N.Veelo@ntnu.no}

{dynarray is an OO array with dynamic bounds.
Although Object Oriented Programming (OOP) has just reached Pascal (outside the
standard that is), it does not allow Generic Programming. A generic implementation
of this container would have been favourable, then it could have been used to contain
any type of data without changing the implementation.
At this moment, the implementation is tailored to contain records consisting of a
vertex_ptr, a shortreal (distance value, used as the key by the sorting algorithm)
and additional data used by the Deformer class (see bnv_deform.pas). When in the
future different types of data need to be contained, the data record can be turned into
```

a schema and the class implementation extended to handle every differentiation of the schema. Any closer to the Standard Template Library you will not get with Pascal; if its not enough, convert to C++.}

```
EXPORT
  dynarray =
  (
    DynamicRandomAccessContainer,
    rich_node_type,
    rich_node_ptr
  ;

IMPORT
  fairway;
  vector;

TYPE
  rich_node_type =
    RECORD
      {vertex belonging to some halfedge:}
      node : vertex_ptr VALUE nil;
      {distance to some other vertex; sort key:}
      d : shortreal VALUE 3.4E+38;
      {which is approximately the maximum value.}
      {Typical shift vector S for this halfedge, and latest scaling factor with which it
      was drawn, i.e. S[dim_gewicht]:}
      S,
      new_pos : vector_type;
    END;
  rich_node_ptr = ^rich_node_type;

  container(capacity:integer) = ARRAY[0..capacity] OF rich_node_ptr;
  container_ptr = ^container;

CONST
  {Minimum size of the container. Setting this value too low makes it grow more often,
  which is expensive.}
  initial_capacity = 64;

TYPE
  Internal_DRAC =
    CLASS (Root)
      s : integer VALUE 0; {size}
      c : container_ptr;
      CONSTRUCTOR Create; override;
      DESTRUCTOR Destroy; override;
      {Indexes start at 0.}
      FUNCTION get( index : integer ) : rich_node_ptr;
      {Returns the index of the appended item:}
      FUNCTION append( content : rich_node_ptr ) : integer;
      {Returns the number of items in the container:}
      FUNCTION size : integer;
      {Returns the current capacity:}
      FUNCTION capacity : integer;
      {This discards the contents of the container, and thus invalidates any references
      to it:}
      PROCEDURE reset;
      PROCEDURE sort;
      {===Private methods:===}
      FUNCTION partition(low, high : integer) : integer;
      PROCEDURE quicksort(low, high : integer);
      {Current implementation copies every element when growing:}
      PROCEDURE grow;
    END;

  {VIEW OF is a hack to hide the private members of Internal_DRAC.}
  DynamicRandomAccessContainer =
    VIEW OF Internal_DRAC (Root)
      Create,Destroy,get,append,size,capacity,reset,sort
    END;

END. {MODULE dynarray}
```

A.2.2 Class Implementation

```
MODULE dynarray implementation;

{Copyright 2002 Bastiaan Veelo
This file may not be used for purposes that are not related to the PhD research of Basti-
aan Veelo without his prior written consent.
Bastiaan.N.Veelo@ntnu.no}

{dynarray is an OO array with dynamic bounds.}

IMPORT
  ywin;
  d3_grap;

CONSTRUCTOR Internal_DRAC.Create; override;
VAR x : Internal_DRAC;
BEGIN
  INHERITED Create;
  new(c,initial_capacity);
END;

DESTRUCTOR Internal_DRAC.Destroy; override;
BEGIN
  dispose(c);
  INHERITED Destroy;
END;

FUNCTION Internal_DRAC.get( index : integer ) : rich_node_ptr;
BEGIN
```

```
    IF( index >= s ) THEN
    BEGIN
      Y_message('Internal_DRAC overflow',
        d3_real_naar_string(index,1,0) +
        '>=' +
        d3_real_naar_string(s,1,0));
      get := nil;
    END
    ELSE
    BEGIN
      get := c^[index];
    END;
  END; { Internal_DRAC.get }

  FUNCTION Internal_DRAC.append( content : rich_node_ptr ) : integer;
  BEGIN
    IF( s > c^.capacity ) THEN grow;
    c^[s] := content;
    append := s;
    s := s + 1;
  END; { Internal_DRAC.append }

  FUNCTION Internal_DRAC.size : integer;
  BEGIN
    size := s;
  END; { Internal_DRAC.size }

  FUNCTION Internal_DRAC.capacity : integer;
  BEGIN
    capacity := c^.capacity;
  END; { Internal_DRAC.capacity }

  PROCEDURE Internal_DRAC.reset;
  var i : integer;
  BEGIN
    FOR i := 0 TO s-1 DO dispose(c^[i]);
    s := 0;
  END; { Internal_DRAC.reset }

  PROCEDURE Internal_DRAC.sort;
  BEGIN
    {Sort the items in this container in ascending order. We use in-place quicksort (ref.
    http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/qsort1a.html)} quicksort(
    0, s-1 );
  END; { Internal_DRAC.sort }

  {According to the manual this one could have been nested in procedure sort, but that
  crashes the compiler...}
  FUNCTION Internal_DRAC.partition(low, high : integer) : integer;
  var p_pos, left, right : integer;
    p_val : shortreal;
    tmp : rich_node_ptr;
```

```
BEGIN
   {set the pivot position to the median item of three samples:}
   p_pos := -1; {SECURITY CHECK}
   {when high = low+1, then ((low+high) div 2) = low. So therefore we use <= when
   comparing middle with low.}
   IF (c^[low]^.d < c^[high]^.d) AND

      (c^[low]^.d <= c^[(low+high) div 2]^.d) THEN
   BEGIN
      IF (c^[(low+high) div 2]^.d < c^[high]^.d) THEN
      BEGIN
         p_pos := (low+high) div 2;
      END
      ELSE
      BEGIN
         p_pos := high;
      END;
   END
   ELSE IF (c^[high]^.d < c^[low]^.d) AND
      (c^[high]^.d <= c^[(low+high) div 2]^.d) THEN
   BEGIN
      IF (c^[(low+high) div 2]^.d < c^[low]^.d) THEN
      BEGIN
         p_pos := (low+high) div 2;
      END
      ELSE
      BEGIN
         p_pos := low;
      END;
   END
   ELSE IF (c^[(low+high) div 2]^.d <= c^[low]^.d) END
      (c^[(low+high) div 2]^.d < c^[high]^.d) THEN
   BEGIN
      IF (c^[low]^.d < c^[high]^.d) THEN
      BEGIN
         p_pos := low;
      END
      ELSE
      BEGIN
         p_pos := high;
      END;
   END;
   IF p_pos = -1 THEN p_pos := (low+high) div 2;
   p_val := c^[p_pos]^.d;
   left := low;
   right := high;
   WHILE left < right DO
   BEGIN
      { Move left while its key < pivot value }
      WHILE (left < size) AND_THEN (c^[left]^.d <= p_val) DO
      BEGIN
         left := left + 1;
      END;
      { Move right while its key > pivot value }
      WHILE c^[right]^.d > p_val DO
      BEGIN
         right := right - 1;
         IF right < 0 THEN Y_message('Internal_DRAC','underrun!!!');
      END;
      IF left < right THEN BEGIN
         {swap the items under left and right. One of them may be the pivot, so maintain
         p_pos.}
         tmp := c^[left];
         c^[left] := c^[right];
         c^[right] := tmp;
         IF p_pos = left THEN p_pos := right
         ELSE IF p_pos = right THEN p_pos := left;
      END;
   END;
   { right is final position for the pivot }
   tmp := c^[p_pos];
   c^[p_pos] := c^[right];
   c^[right] := tmp;
   partition := right; {and of course p_pos:=right, but we don't care any more}
END;

{According to the manual this one could have been nested in procedure sort, but that
crashes the compiler...}
PROCEDURE Internal_DRAC.quicksort(low, high : integer);
var pivot : integer;
BEGIN
   IF high > low THEN { Termination condition! }
   BEGIN
      pivot := self.partition( low, high );
      quicksort( low, pivot-1 );
      quicksort( pivot+1, high );
   END;
END;

PROCEDURE Internal_DRAC.grow;
VAR c_old : container_ptr;
   i : integer;
BEGIN
   c_old := c;
   new(c, c_old^.capacity*2);
   FOR i := 0 TO c_old^.capacity do c^[i] := c_old^[i];
   dispose(c_old);
END; { Internal_DRAC.grow }

END.
```

APPENDIX B

Article in "Ship Technology Research"

The article on the following pages has been published in the international scientific journal of Schiffstechnik/Ship Technology Research [Veelo, 2004a].

Shape Modification of Ship Hulls in H-rep

Bastiaan N. Veelo, Norwegian University of Science and Technology[1]

1 Introduction

The transfinite interpolation of an irregular network of curves is an effective modelling methodology for the design of non-trivial free-form shapes. Recent literature refers to this methodology with the term hybrid representation or H-rep, to indicate that it is based on the merger of wire-frame modelling and solid modelling. In practice, the term H-rep also implies the integration of a curve fitting/fairing algorithm, which is essential for the discussions in this article. *Koelman (1999)* gives a detailed description of the conception of the H-rep concept, and of an implementation called 'Fairway', which is a module in the 'PIAS' package and tailored to (but not limited to) the geometric design of ship hulls. Introductions to the H-rep concept and its value for the maritime industry include *Koelman et al. (2001)*, *Koelman (2003a,b)*, *Veelo (2004)*.

In essence, Koelman's implementation restores the traditional way of lines plan draughting in a computer method. To the user, the system presents itself as a curve modeller. The surface generation, which consists of filling the mesh cells of the curve network with transfinite surface patches, is completely hidden for the user. The patches may have an arbitrary number of sides and are tangent-plane continuous across shared boundaries. Advantages of this modelling concept are the absence of topological restrictions on surface features, the independence of curves and their details, and tight control over the exact shape of the composite surface.

Typically, when starting a design from scratch using this system, the designer starts with an initial model defined by a centre line contour, a deck line and a mid-ship ordinate. These curves are computer-generated, based on user-defined main dimensions. The shape of the surface patches is completely derived from the shape of the curves, so the only means to control the shape of the model is through manipulation of curves. Thus, the first step in the design process is to modify the existing curves to their correct shape, by traditional control point manipulation. When the surface is visualised at this stage, the designer will be probably not satisfied with the composite shape of surface patches. The patches are still large and the defining curves are too far apart to describe every detail that is envisioned for the design. The solution is to add more curves to the network, effectively subdividing patches into smaller ones. New curves can be generated automatically by intersecting the model with a user-defined plane, or by projecting a separate curve onto the surface. After addition of a new curve, the shape of newly subdivided patches can be modified by manipulating the curve. With this process, the shape of a sculpted model evolves from a coarse definition to a detailed definition, until the designer is satisfied with the result.

There is a downside to this modelling methodology. As the design progresses and more curves are added to the model, more of its shape is rigidly defined. The more curves present, the smaller the surface patches, and the more local shape manipulations become. More curves also mean more curve intersections. As a result, during curve manipulation, there is a higher risk that a curve is pulled away from these intersections, rendering the network invalid as a surface description. Such inconsistencies appear as gross surface defects. Currently, no mechanism is implemented that prevents this, or that resolves the incompatibilities. Although it is possible to restore the intersections manually, by adjusting all affected curves to the changes, this is a lengthy, iterative task.

Consequently, making design changes that affect larger surface areas of the model, is discouraged at late design stages. One could say that designing sculpted shapes this way, in practice is a one-dimensional process because the design has a preference to evolve only in one direction. This paper proposes a simple and efficient method for shape manipulation of a dense curve network that is not strictly local and does not destroy the consistency and the geometric continuity of the network. This

[1]NTNU, Dept. Eng. Design and Materials, N-7491 Trondheim, Norway, Bastiaan.N.Veelo@ntnu.no

makes it possible to migrate directly from one shape variation to the other, by which the design process becomes, in our way of speaking, multi-dimensional.

2 Background

In the computer-aided design (CAD) of free-form or sculpted shapes, 'Non-Uniform Rational B-Spline' (NURBS) surfaces enjoy great popularity. NURBS surface patches derive their geometry from control points that they approximate. Whether their popularity is justified, is debatable. The challenge of designing sculpted shapes boils down to two main problems: geometric continuity and control.

A single NURBS surface patch without degenerate sides or corners fits only well to deformations of the square, the cylinder and the torus – of which only the torus can describe the boundary of a solid. All other geometries, thus including most subjects of design, are commonly denoted as having arbitrary topology. For a composition of (non-degenerate) patches to describe shapes of arbitrary topology, the patches must be allowed to be organised in an arbitrary manner, where the number of patches that mutually connect with one of their corners is not always four. Maintaining geometric continuity, i.e. a smooth composite surface, is especially difficult at these irregular points.

The problem of control is implied by the fact that NURBS surface patches approximate a regular grid of control points. The problem becomes eminent, e.g. when more control points are needed locally in order to define some local detail in a surface patch. Since extra control points can only be added in complete rows or columns, they also appear in regions where they are not wanted, because they make achieving surface fairness more difficult.

Transfinite surface patches, which derive their geometry from curves that they interpolate, do not suffer from a control problem, as the patch does not care how its bounding curves are defined. For adjacent transfinite patches to connect with tangent plane continuity, tangent information is required along the curves, which is represented by so-called tangent ribbons. In order to model arbitrary topology, the curve network must also be arbitrary, i.e. without regularity requirements. The only requirement on curves is that they intersect and not cross each other (within some tolerance) and that they start and end at other curves. *Jensen et al. (1991)* were the first to develop a technique for the generation of tangent ribbons on such networks by using a boundary representation (B-rep), which is a data structure used primarily in solid modelling. A B-rep data structure consists of topological elements of type 'node', 'edge' and 'face'. References exist in the data structure such that for each element, its neighbouring elements can be determined. *Jensen et al.* applied their technique to automotive styling. *Van Dijk (1994)* took this to conceptual industrial design and *Michelsen (1995)* to naval architecture.

Although having developed an alternative to NURBS surface modelling without the associated problems, it remained a challenge to keep the curve network simultaneously fair and consistent as a surface representation. By integrating a curve fitting and fairing algorithm, *Koelman (1999)* was able to improve that situation, and produced the described implementation for the geometric design of ship hulls, at production quality. In addition, he removed the need for the user to worry about surface patches, by following a suggestion of *Michelsen (1995)* to use the B-rep to its full potential as a solid representation.

Fig.1 shows the hybrid nature of the H-rep. The nodes in the B-rep refer to intersection points in the wire-frame for their geometry. The edges refer to the curve sections in-between the intersection points and the faces refer to the n-sided patches that can be generated to fill the openings in the wire-frame. Tangent ribbons are also partly indicated.

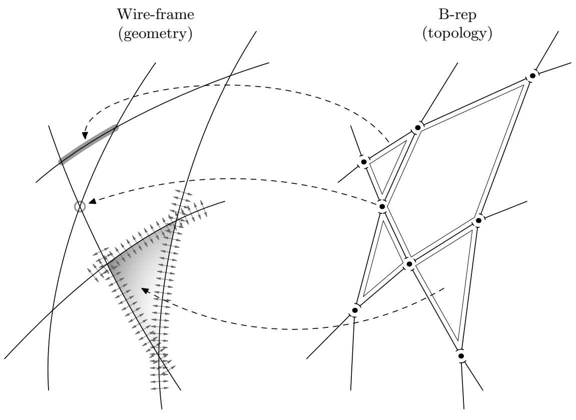


Fig.1: The hybrid representation with references between topology and geometry

3 Related Research

The problem of the global shape of a model getting fixated by the definition of details is not specific to the H-rep and its precedents. It also exists in systems that are based on approximation of control points such as NURBS surfaces, although the consequences are not as dramatic as the surface defects that can arise in an H-rep. If a surface region is to be modified for which a larger number of control points need to be shifted, this must be done in a way that preserves the coherence between the control points, so that both the global shape remains fair and the detailed surface features are not damaged. This has been addressed by the integration of physics based properties, e.g. by *Terzopoulos and Qin (1994)* and *Leon and Trompette (1995)*, and hierarchical refinement, by *Forsey and Bartels (1988)*. These references do not explicitly consider arbitrary topology however. *Gonzalez-Ochoa and Peters (1999)* proposed the hierarchical refinement principle for so-called surface splines, which approximate an arbitrary mesh of control points and thus support models with arbitrary topology. Contrary to plain NURBS surfaces, their contribution may be a viable alternative to the H-rep.

Free-form deformation (FFD), popularised by *Sederberg and Parry (1986)*, is a technique to reshape a geometric model indirectly by warping the space in which it is defined. FFD is independent of the model definition, and thus competes with the method presented here.

4 Manipulation of Sets of Data Points

We will state our problem as follows: "Given a certain region on a surface that interpolates a network of curves, manipulate all curves in that region simultaneously, in a way that does not destroy the consistency of the network and does not introduce unwanted geometric discontinuities."

This statement can be symplified after the following observation. The details behind the process of adding a new curve consist of tracing a string of data points over the surface, as a sampling of the intersection curve or the projected curve. Then the fitting/fairing algorithm is invoked to generate a curve through these points, which is added to the model. During manipulation of curves (and thus the surface) these data points can be made to move with the curve, so that they indeed remain positioned on the surface. If we assume for the moment that all data points are persistent, meaning that they remain in existence throughout the modelling process, then the complete model can be regenerated from the data points and the B-rep alone, with the help of the fitting/fairing algorithm. Thus, we can reformulate the problem as: "Given a point set belonging to a consistent H-rep, shift a selection of points to a new position, so that the distance and the direction of the shift of each individual point varies smoothly over the set."

We will now assert our assumption. Data points that are associated with intersections between curves are persistent, because they represent the geometry of node elements in the B-rep. Currently, other data points are not persistent, as they serve no purpose after the creation of a curve. Nevertheless, in a dense curve network, there will likely be enough intersections (and thus persistent data points) to record the shape of the curves. A simple heuristic can verify this, e.g. by checking whether the number and distribution of data points belonging to a curve stands in proportion to the number and distribution of control points of the curve. If the verification fails, extra data points can be inserted at low computational cost.

4.1 Shift Vectors

Let us declare $\mathbf{s}_i$ to be the 'shift vector' for a data point i, i.e. the difference between the position of that point after and before the shape modification. We will define this shift vector as the vector sum of the sample of one or more three-dimensional vector fields. A vector field is primarily defined by a 'selection field' j of varying intensity, which is concentrated around a point, a curve or a surface, which we will call the 'base' of the selection field, Fig.2. This base may be part of the model, or be dedicated to support the selection field. The intensity f_j of the field will be unity at its base, decreasing smoothly with increasing distance d to the base, and level off to zero at a distance r_j to the base, which we will call the 'extent' of the selection field. If the base is singular, r_j is a constant; but if the selection field is based on a curve (or surface), r_j may be a function of the curve parameter (or surface parameters). In a similar fashion, we will define a vector on the base, whose length and direction may be a function of the base parameters. We will call this vector the 'typical shift vector' of the selection field, denoted by $\mathbf{S}_j$. For the vectors in the field to vary smoothly, it is important that selection fields do not self-intersect.

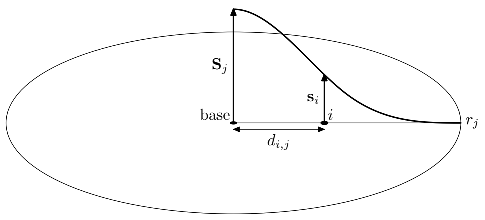


Fig.2: Schematic presentation of how the shift ($\mathbf{s}_i$) of a point i is derived from a selection field of smoothly decaying intensity f_j and a given shift ($\mathbf{S}_j$) at the selection base. In this case, the base is singular and $\mathbf{s}_j = f_j(d_{i,j}/r_j)\mathbf{S}_j$.

In addition to a selection field, one or more 'deselection fields', enumerated by k, may take part in the definition of a vector field. Deselection fields reverse the effect of the selection field. Their definition is similar to the definition of selection fields, except that they lack a typical shift vector and their intensity g_k is opposite to the intensity of selection fields: unity outside their extent, smoothly decreasing in inverse proportion to the distance d to the base inside their extent, and levelling off to zero at their base.

The vector field is then defined as the typical shift vector, evaluated on the closest point on the base, multiplied by the selection field intensity and the deselection field intensities. Especially for deselection fields it is interesting to have them act differently on the x, y and z coordinates of the vectors in the field, and thus we will redefine field intensities as diagonal matrix functions:

$$\begin{aligned}
\mathbf{f}_j\left(\frac{d_{i,j}}{r_j}\right) &\equiv \begin{bmatrix} f_{j,x}\left(\frac{d_{i,j}}{r_j}\right) & 0 & 0 \\ 0 & f_{j,y}\left(\frac{d_{i,j}}{r_j}\right) & 0 \\ 0 & 0 & f_{j,z}\left(\frac{d_{i,j}}{r_j}\right) \end{bmatrix} \\
\mathbf{g}_k\left(\frac{d_{i,k}}{r_k}\right) &\equiv \begin{bmatrix} g_{k,x}\left(\frac{d_{i,k}}{r_k}\right) & 0 & 0 \\ 0 & g_{k,y}\left(\frac{d_{i,k}}{r_k}\right) & 0 \\ 0 & 0 & g_{k,z}\left(\frac{d_{i,k}}{r_k}\right) \end{bmatrix}
\end{aligned} \tag{1}$$

If we then say $g_{k,y} \equiv 0$ for a deselection field based on the plane $y = 0$, data points in that plane will only shift in that plane and not away from it, regardless of the direction of the typical shift vector. This is advantageous if the design is symmetrical around $y = 0$ and only one half of it is modelled. Note that in this specific case, an alternative is to mirror the selection field in the symmetry plane. Deselection fields however are capable of enforcing more general constraints.

The definition of the shift vector can now be formalised as

$$\mathbf{s}_i \equiv \sum_j \left(\prod_k \Big(\mathbf{g}_k(d_{i,k}/r_k) \Big) \mathbf{f}_j(d_{i,j}/r_j) \mathbf{S}_j \right), \tag{2}$$

where $d_{i,j}$ denotes the shortest distance through space between data point i and the closest point on the base of selection field j, and $\mathbf{S}_j$ and $\mathbf{f}_j$ are evaluated at that position on the base. Analogously, $d_{i,k}$ denotes the shortest distance through space between data point i and the closest point on the base of deselection field k, and $\mathbf{g}_k$ is evaluated at that position on the base. What remains is to find suitable definitions for the selection functions f and g, and for the typical shift vector $\mathbf{S}$.

4.2 Selection Functions

Any function that behaves as described will give useful results. For more control of the shape of the resulting modification, one may want to vary the shape of the selection function over the base of the selection, as a function of the base parameters. This is possible in the following definition of a cubic piecewise polynomial, in which a parameter $\kappa \in [0, 1]$ defines how fast the function falls off.

$$f\left(\frac{d_{i,j}}{r_j}\right) \equiv f(\delta) \equiv \begin{cases} \dfrac{\kappa^2 + \kappa\delta^2(\delta - 3) + \delta^3}{\kappa^2} & \text{if } 0 \le \delta < \kappa \\ \dfrac{(\delta - 1)^3}{\kappa - 1} & \text{if } \kappa \le \delta < 1 \\ 0 & \text{if } 1 \le \delta \end{cases} \tag{3}$$

in which δ has been substituted for $d_{i,j}/r_j$ for simplicity. This function, Fig.3, is derived from the Cox-deBoor recursive definition of B-spline basis functions. The selection function of deselection fields can simply be defined as $g \equiv 1 - f$.

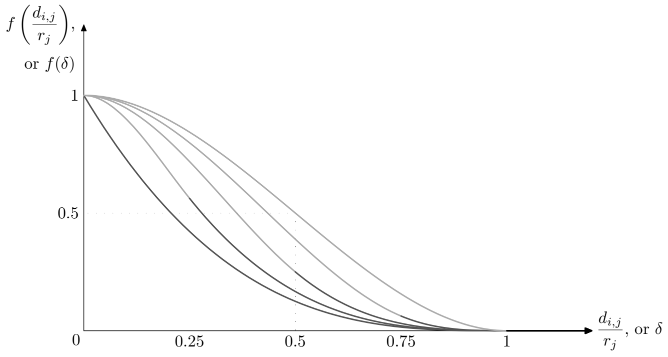


Fig.3: The selection function defined by Eq.(3): light grey for $0 \le \delta < \kappa$, dark grey for $\kappa \le \delta < 1$, black for $1 \le \delta$. Smaller values of κ make the function fall off faster. Plotted are $\kappa = 1.0$, which is point-symmetric about (0.5,0.5), $\kappa = 0.75$, $\kappa = 0.5$, $\kappa = 0.25$ and $\kappa = 0.0$. The latter has a cusp at the base, which is not beneficial for surface fairness and therefore not of much interest to us.

4.3 Typical Shift Vector

A selection field with a singular base can very well be based on a data point on the surface. It will be natural to take the surface normal at that point as the typical shift vector, scaled up or down if necessary.

For selection fields that are based on a curve, a powerful modelling tool results if the typical shift vector can be varied along the curve. Put simply, the shift vector can be defined as the difference between the base curve, say $\mathbf{c}(t)$, and an other curve, say $\hat{\mathbf{c}}(\hat{t})$. If $\mathbf{c}(t)$ is a curve on the surface prior to the shape modification, then $\hat{\mathbf{c}}(\hat{t})$ is exactly what the model will look like at this location, after the modification. Thus, designers will be able to manipulate feature curves, or even completely redesign them, while they will be able to control how the other curves (and thus the surface) in their vicinity adapts to the changes with the parameters r and κ. In addition, they will be able to protect other feature curves during the modification, by basing a deselection field on them.

To make this principle work as expected, it needs to be refined. As $\hat{\mathbf{c}}(\hat{t})$ may be completely different from $\mathbf{c}(t)$, their parameterisation may be different. In other words, when two particles are considered, one travelling down each curve at proportional increments of t and $\hat{t}$, the variation in velocity of the two particles may not be parallel. The effect on the typical shift vector will be that it changes direction more often than necessary. We remedy this by evaluating the curves with respect to arc length. In addition the designer may not want to change the complete curve, and $\hat{\mathbf{c}}(\hat{t})$ may partly coincide with $\mathbf{c}(t)$. But due to the different lengths of $\mathbf{c}(t)$ and $\hat{\mathbf{c}}(\hat{t})$, the typical shift vector may still have non-zero length in these parts, which is not intended. To counter this, we must evaluate the curves only over the curve sections that actually have different geometries.

Let the curves $\mathbf{c}(t)$ and $\hat{\mathbf{c}}(\hat{t})$ differ from each other for $t \in [t_a, t_b]$ and $\hat{t} \in [\hat{t}_c, \hat{t}_d]$, with $t_{\text{begin}} \le t_a < t_b \le t_{\text{end}}$ and $\hat{t}_{\text{begin}} \le \hat{t}_c < \hat{t}_d \le \hat{t}_{\text{end}}$. For other parameter values, the curves coincide, although not necessarily for equal parameter values. The exact value of t_a, t_b, $\hat{t}_c$ and $\hat{t}_d$ can be determined by analysis of the control points and knot vectors of the curves. For a formal definition of the shift vector, we need a mapping $m : t \mapsto \hat{t}$. For a certain parameter value t_i, $m(t_i)$ must produce a $\hat{t}_j$ so that the arc lengths of the curve sections on either side of these parameter values are proportional:

$$\frac{\int_{t_a}^{t_i} |\dot{\mathbf{c}}(t)| \mathrm{d}t}{\int_{t_i}^{t_b} |\dot{\mathbf{c}}(t)| \mathrm{d}t} = \frac{\int_{\hat{t}_c}^{\hat{t}_j} |\dot{\hat{\mathbf{c}}}(\hat{t})| \mathrm{d}\hat{t}}{\int_{\hat{t}_j}^{\hat{t}_d} |\dot{\hat{\mathbf{c}}}(\hat{t})| \mathrm{d}\hat{t}} \tag{4}$$

in which the arc length of a curve is defined as the integral of the length of the first derivative of that curve with respect to the curve parameter. Because m is inefficient to be evaluated directly, one should first assess whether arc length evaluation is at all worthwhile, by comparing internal knot spacings and control point distances of $\mathbf{c}(t)$ and $\hat{\mathbf{c}}(\hat{t})$, looking for large discrepancies. If so, the map m may be approximated by evaluating $m(t)$ at distinct values of t_i, and fitting a polynomial, say $\hat{m}(t)$, through the mapped values $\hat{t}_j$.

Now we are able to define the typical shift vector $\mathbf{S}$ as the difference between corresponding positions on the two curves according to arc length, expressed as a function of the curve parameter t:

$$\mathbf{S}(t) \equiv \hat{\mathbf{c}}(\hat{m}(t)) - \mathbf{c}(t) \tag{5}$$

Fig.4 illustrates the necessity of going through the trouble of considering only curve sections of dissimilar geometry and computing arc lengths.

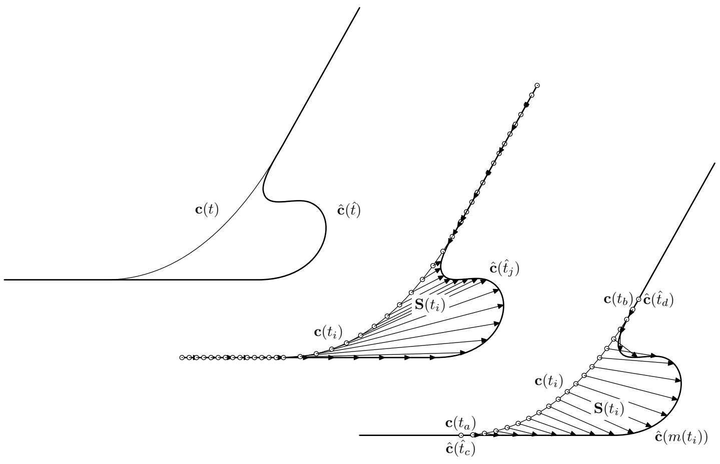

$\mathbf{c}(t)$
$\hat{\mathbf{c}}(\hat{t})$
$\hat{\mathbf{c}}(\hat{t}_j)$
$\mathbf{S}(t_i)$
$\mathbf{c}(t_i)$
$\mathbf{c}(t_b)$
$\hat{\mathbf{c}}(\hat{t}_d)$
$\mathbf{c}(t_i)$
$\mathbf{S}(t_i)$
$\mathbf{c}(t_a)$
$\hat{\mathbf{c}}(\hat{t}_c)$
$\hat{\mathbf{c}}(m(t_i))$

Fig.4: Upper left: original stem curve $\mathbf{c}(t)$ and re-designed stem curve $\hat{\mathbf{c}}(\hat{t})$. Middle: straight-forward definition of typical shift vector $\mathbf{S}(t) \equiv \hat{\mathbf{c}}(\hat{t}) - \mathbf{c}(t)$. Data points are shifted even where geometry is not changed, and their spacing becomes unbalanced. Lower right: considering arc lengths over sections of changed geometry, $\mathbf{S}(t) \equiv \hat{\mathbf{c}}(m(t)) - \mathbf{c}(t)$, gives a much cleaner result.

4.4 Unintended Selections

The proposed method for shape modification is simple, as we do not regard the surface of the model at all, and only consider data points and their shortest distance to selection bases. The advantage is speed. Shift vectors can be computed quickly enough to visualise them in real time, while the designer manipulates the modification parameters. They give a sufficient indication of how the shape will be modified once the parameters are accepted. Therefore, the presented method for shape modification is highly interactive.

However, in some situations this approach can be too simple. For instance, when modifying an area on the upper side of a thin wing. Because the data points on the lower side are close to the upper side, they may be selected unintentionally. Even though this may be prevented by careful definition of deselection fields, there is an alternative that can be automated, which involves putting the B-rep to good use.

Let us define the 'root node' of a selection field as the node that references the data point closest to the base of the selection field. If there are several nodes that qualify, any one of these will do. The algorithm in Fig.5 will only shift data points that form a contiguous selection that is rooted at the base, and prune away isolated sets of selected data points that are separated from the main selection by more than one surface patch.

```
Let $\mathcal{A}$ be an empty set, capable of containing node references
Mark all nodes and faces in the B-rep as unconsidered
Mark the root node as considered and add it to $\mathcal{A}$
While $\mathcal{A}$ is not empty {
    Compute $\mathbf{s}_i$ for a node $i \in \mathcal{A}$
    If $|\mathbf{s}_i| > 0$, then {
        For all faces $j$ that are adjacent to node $i$ and that are still unconsidered {
            Mark $j$ as considered
            For all nodes $k$ that are adjacent to $j$ and that are still unconsidered {
                Mark $k$ as considered and add it to $\mathcal{A}$
            }
        }
        shift node $i$
    }
    remove $i$ from $\mathcal{A}$
}
```

Fig.5: Pseudo-code of an algorithm that only shifts contiguous sets of data points.

5 Finishing Up

Once the fitting/fairing algorithm has re-interpolated the curves over the shifted data points, the H-rep has become a consistent and smooth modification from the original, by which we have succeeded in our objective. However, if the original primarily consisted of planar curves, such as is customary in the design of ship hulls, these may no longer be planar after the modification.

Planar curves can be restored by intersecting the modified model with the planes in which the curves were originally defined, and adding the intersection curves to the model. These new curves take over the definition of the modified shape, by which the old curves become redundant and may be removed.

6 Applications

6.1 Example of Shape Variation

Fig.6 shows how the fore ship of a frigate has been made slightly narrower. The selection base was a point on the hull, and the shift was in transverse direction. The system has no difficulties with the knuckle line that is present in this region. A critique on this particular shape variation may be that the shell near the stem contains too much of the original shape, resulting in an extra inflection. This is due to the shape of the deselection function that was chosen to constrain the shell to the plane of symmetry, as the value of κ in Eq.(1) was set to 1.0, Fig.3. A value of 0 would have been better in this regard. Optimal would have been not to base the deselection field on the plane of symmetry, but on the contour line itself. Then κ could have been varied along it, 0 where the stem is sharp and non-zero elsewhere.

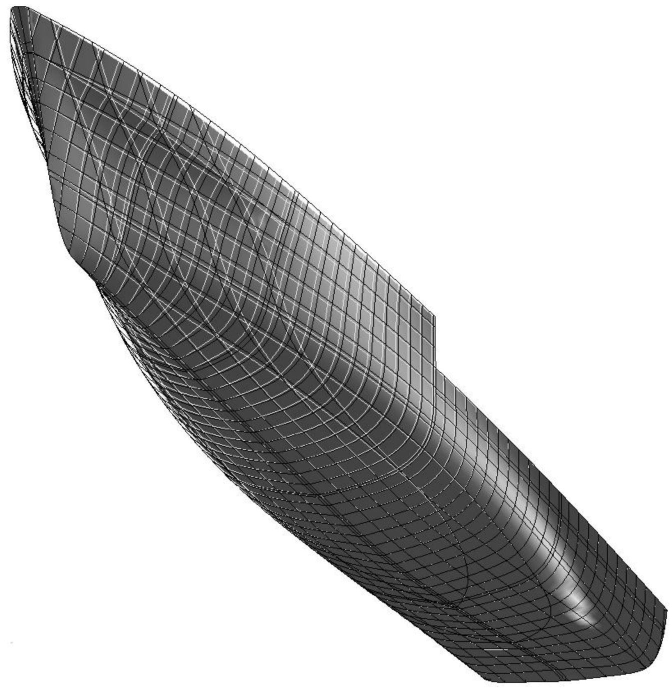


Fig.6: Modification of the hull of a frigate (shaded surface and light grey wire-frame) together with the original shape (black wire-frame overlay).

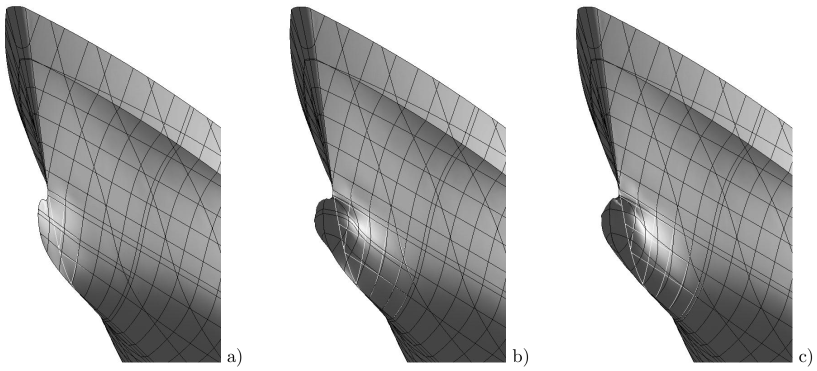


Fig.7: Three successive curve-based selections turn a plain bow (white wire-frame) into a bulbous bow.

6.2 Example of Bulb Design

Fig.7 shows shape variations resulting from a curve-based selection, illustrating that the method provides a powerful modelling tool. It shows all three steps in the process of designing a bulb from scratch. Starting off with the same model as in Fig.6, the first step was to re-design the stem curve, Fig.7a. An auxiliary waterline was added ending in the fore-most position of the bulb, just below the second ordinary waterline counted from below. On this line the second selection was based, giving the bulb more body, Fig.7b. Finally, the lowest waterline was dragged slightly outward, to improve the shape of the frames in the lower region, Fig.7c. The dent at the top of the bulb is due to scarce geometric data. This was corrected with two extra frames and one extra waterline, Fig.8.

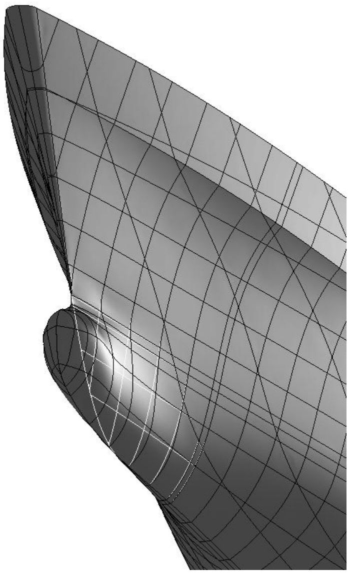


Fig.8: Extra defining data added to correct dent in upper part of bulb.

6.3 Performance

Shape variations with selections based on points and planes are fast. However, curve-based selections can be time consuming, because computing the global closest distance of a point to a curve is expensive. In the current implementation, which is not for production and serves proof of concept only, all data points in the entire model are processed. Depending on the length and complexity of the curve, the time needed for distance computations in the examples of Fig.7 took up to several minutes on a 1.5 GHz PC. Once the distances are computed, shift vectors can be previewed interactively while r and κ are varied.

This performance can be improved. Firstly, it may be possible to omit several iterations in the closest point finding algorithm per data point, if the loop termination is not based on the accuracy of the closest point, but on the accuracy of the resulting shift vector. Secondly, it may be possible to disregard data points on curves that were never manipulated since they were added to the model. These curves do not actually contribute to the definition of the shape, and serve only for surface visualisation, surface quality interrogation and/or manufacturing (e.g. frame contours and the butts and seams of shell plating). In late stages of the design, there may be many of these curves. Unless the resulting shape has so much detail that the extra curves actually contribute to the definition of the new geometry, they can safely be deleted before the shape variation and restored afterwards. A heuristic based on the typical shift vector can determine this possibility. Finally, the shape variations that are discussed here rarely involve every single data point in the model; in case it does, an ordinary affine transformation probably performs better. So computing the distance for all data points is a waste of time. It would be much better to consider data points on demand, based on the extent of the selection. For this a graph search is required, similar in nature to the algorithm discussed in Ch.4.4.

With the above in mind, it is advisable for an implementation of the H-rep concept that when new curves are interpolated, their addition to the B-rep data structure be deferred until they are selected by the designer for explicit manipulation, which is the point at which they start taking part in actual shape definition. Curves that are not in the data structure should then be regenerated automatically whenever any of their underlying patches change.

7 Conclusion

A simple method for the modification of a network of intersecting curves was presented, which preserves the consistency of the network, the fairness of the surface and local surface features. Curves may be redesigned explicitly, regardless of the detail in a design, with intuitive control over how the surface in their vicinity adapts to the changes. This can be regarded as an advantage over the competing method of free-form deformation (FFD).

References

FORSEY, D.R.; BARTELS, R.H. (1988), *Hierarchical B-spline refinement*, 15th Annual Conf. Computer Graphics and Interactive Techniques, ACM Press, pp.205-212

GONZALEZ-OCHOA, C.; PETERS, J. (1999), *Localized-hierarchy surface splines (LeSS)*, Symp. Interactive 3D graphics, Atlanta, pp.7-15

JENSEN, T.W.; PETERSEN, C.S.; WATKINS, M.A. (1991), *Practical curves and surfaces for a geometric modeler*, Computer Aided Geometric Design 8/5, pp.357-369

KOELMAN, H.J. (1999), *Computer support for design, engineering and prototyping of the shape of ship hulls*, TU Delft, ISBN 90-901-2888-3

KOELMAN, H.J. (2003a), *Exploring the H-rep ship hull modelling concept*, 2nd Conf. Comp. and IT Appl. in the Maritime Ind., COMPIT, Hamburg, pp.140-153, `http://www.sarc.nl/compit03.pdf`

KOELMAN, H.J. (2003b), *Application of the H-rep ship hull modelling concept*, Ship Techn. Res. 50, pp.172-181

KOELMAN, H.J.; HORVÁTH, I.; AALBERS, A. (2001), *Hybrid representation of the shape of ship hulls*, Int. Shipb. Progr. 48, pp.247-269

LEON, J.C.; TROMPETTE, P. (1995), *A new approach towards free-form surfaces control*, Computer Aided Geometric Design 12/4, pp.395-416

MICHELSEN, J. (1995), *A free-form geometric modelling approach with ship design applications*, PhD thesis, NAOE, TU Denmark, Lyngby

SEDERBERG, T.; PARRY, S. (1986), *Free-form deformation of solid geometric models*, ACM SIGGRAPH 20/4, pp.151-160

TERZOPOULOS, D.; QIN, H. (1994), *Dynamic NURBS with geometric constraints for interactive sculpting*, ACM Trans. on Graphics 13/2, pp.103-136, `http://mrl.nyu.edu/~dt/papers/tog94/tog94.pdf`

VAN DIJK, C.G.C. (1994), *Interactive modeling of transfinite surfaces with sketched design curves*, PhD thesis, TU Delft

VEELO, B.N. (2004), *Shape modification of hull models in H-rep*, 3rd Conf. Comp. and IT Appl. in the Maritime Ind., COMPIT, (Ed. Bertram), Siguënza, pp.212-222

Bibliography

C. **Abt**, S.D. **Bade**, L. **Birk** and S. **Harries** [2001]. Parametric Hull Form Design — a Step Towards One Week Ship Design. In 8th *International Symposium on Practical Design of Ships and Other Floating Structures (PRADS 2001)*. Shanghai.

Jeff B. **Allan**, Brian **Wyvill** and Ian H. **Witten** [1989]. A Method for Direct Manipulation of Polygonal Meshes. In *Proceedings of Computer Graphics International* (edited by Brian **Wyvil**), pp. 451–469. Springer-Verlag, Leeds, England.

Pierre **Bézier** [1978]. General Distortion of an Ensemble of Biparametric Surfaces. *Computer-Aided Design*, vol. 10, no. 2, pp. 116–120.

James R. **Bill** [1994]. *Computer Sculpting of Polygonal Models using Virtual Tools*. Master's thesis, University of California, Santa Cruz. Available from `http://citeseer.ist.psu.edu/bill94computer.html`.

W. **Boehm** [2002]. Differential Geometry I. In Farin [2002a], chap. 10, pp. 179–189.

Jeffrey **Bolz** and Peter **Schröder** [2002]. Rapid Evaluation of Catmull-Clark Subdivision Surfaces. In *Proceeding of the seventh international conference on 3D Web technology*, pp. 11–17. ACM Press. Also available from `http://www.multires.caltech.edu/pubs/fastsubd.pdf`.

Grady **Booch** [1991]. *Object-Oriented Design with Applications*. The Benjamin/Cummings Publishing Company, Inc.

Paul **Borrel** and Ari **Rappoport** [1994]. Simple Constrained Deformations for Geometric Modeling and Interactive Design. *ACM Transactions on Graphics*, vol. 13, no. 2, pp. 137–155.

Walter **Bright** [2002]. The D Programming Language. *Dr. Dobb's Journal*. See also `www.digitalmars.com/d/`.

Hervé **Brönnimann** [2001]. Designing and Implementing a General Purpose Halfedge Data Structure. In *Algorithm Engineering: 5th International Workshop, WAE 2001 Aarhus, Proceedings* (edited by G.S. **Brodal**, F. **Frigioni** and A. **Marchetti-Spaccamela**), vol. 2141/2001 of *Lecture Notes in Computer Science*, pp. 51–66. Springer-Verlag Heidelberg.

E. **Catmull** and J. **Clark** [1978]. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, vol. 10, no. 6, pp. 350–355.

G. **Celniker** and D. **Gossard** [1991]. Deformable curve and surface finite elements for free-form shape design. In *Computer Graphics Proceedings*, vol. 25 of *Annual Conference Series*, pp. 257–266. ACM SIGGRAPH.

G. **Celniker** and W. **Welch** [1992]. Linear constraints for deformable B-spline surfaces. In *Symposium on Interactive 3D Graphics*, ACM Conference Proceedings Series, pp. 165–170. ACM Press.

Hiroaki **Chiyokura** and Fumihiko **Kimura** [1983]. Design of Solids with Free-Form Surfaces. *Computer Graphics, SIGGRAPH'83 Conference Proceedings*, vol. 17, no. 3, pp. 288–298.

Peter **Coad** and Edward **Yourdon** [1991]. *Object-Oriented Design*. Yourdon Press Computing Series. Prentice-Hall.

Sabine **Coquillart** [1990]. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 24, pp. 187–196. ACM Press.

J. **Cotrina Navau** and N. **Pla Garcia** [2000]. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design*, vol. 17, pp. 643–671.

F. **Dachille IX**, H. **Qin** and A. **Kaufman** [2001]. A novel haptics-based interface and sculpting system for physics-based geometric design. *Computer-Aided Design*, vol. 33, no. 5, pp. 403–420. Also available from the author's homepage `http://www.cs.sunysb.edu/~dachille/pub/hapticJournal.html`.

P. **Dierckx** [1993]. *Curve and Surface Fitting with Splines*. Oxford University Press, Oxford, UK.

Casper G.C. **van Dijk** [1994]. *Interactive modeling of transfinite surfaces with sketched design curves*. Ph.D. thesis, Delft University of Technology, the Netherlands.

D. **Doo** and M. **Sabin** [1978]. Behaviour of recursive division surfaces near extroardinary points. *Computer-Aided Design*, vol. 10, no. 6, pp. 356–360.

Matthias **Eck** and Hugues **Hoppe** [1996]. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 325–334. Also available at `http://citeseer.nj.nec.com/eck96automatic.html` and `ftp://ftp.research.microsoft.com/pub/tech-reports/Winter95-96/Tr-96-01.ps` and `http://wwwbib.mathematik.tu-darmstadt.de/Math-Net/Preprints/Listen/shadow/pp1800.html`.

Gerald **Farin** [2002a]. *Curves and Surfaces for CAGD: a Practical Guide*. Fifth edn. Academic Press.

Gerald **Farin** [2002b]. A History of Curves and Surfaces in CAGD. In Farin *et al.* [2002], chap. 1, pp. 1–21.

Gerald **Farin**, Josef **Hoschek** and Myung-Soo **Kim** (editors) [2002]. *Handbook of Computer Aided Geometric Design*. First edn. Elsevier Science.

Jieqing **Feng**, Pheng-Ann **Heng** and Tien-Tsin **Wong** [1998]. Accurate B-Spline Free-Form Deformation of Polygonal Objects. *Journal of Graphics Tools*, vol. 3, no. 3, pp. 11–27. Also available from `http://www.cad.zju.edu.cn/home/jqfeng/Publications.htm`.

Sven **Fjeldaas** [1985]. *Geometry and Topology of Binary Pictures*. Dr.techn. thesis, NTH, Trondheim, Norway.

James D. **Foley**, Andries **van Dam**, Steven K. **Feiner** and John F. **Hughes** [1990]. *Computer Graphics; Principles and Practice*. Second edn. Addison-Wesley.

David R. **Forsey** and Richard H. **Bartels** [1988]. Hierarchical B-spline refinement. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pp. 205–212. ACM Press.

Barry **Fowler** and Richard **Bartels** [1993]. Constraint-Based Curve Manipulation. *IEEE Computer Graphics and Applications*, vol. 13, no. 5, pp. 43–49.

Martin **Fowler** [2003]. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley/Pearson Education.

Free Software Foundation [2000]. GNU Lesser General Public License. `http://www.fsf.org/copyleft/lesser.html`.

James Edward **Gain** [2000]. *Enhancing Spatial Deformation for Virtual Sculpting*. Ph.D. thesis, University of Cambridge, United Kingdom. Also available from `http://citeseer.nj.nec.com/337384.html`.

Sarah F. F. **Gibson** and Brian **Mirtich** [1997]. A Survey of Deformable Models in Computer Graphics. Tech. Rep. MERL-TR-97-19, A Mitsubishi Electronic Research Laboratory, 201 Broadway, Cambridge, Massachusetts. URL `http://www.merl.com/reports/TR97-19/` and `http://citeseer.nj.nec.com/gibson97survey.html`.

Carlos **Gonzalez-Ochoa** [1997]. Interactive Modeling using Surface Splines. In *1er Encuentro de Computacion, ENC 97*. Sociedad Mexicana de Ciencia de la Computacion. URL `http://citeseer.nj.nec.com/120179.html`.

Carlos **Gonzalez-Ochoa** and Jörg **Peters** [1999]. Localized-hierarchy surface splines (LeSS). In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pp. 7–15. ACM Press. `http://doi.acm.org/10.1145/300523.300524`.

John A. **Gregory** [1983]. C^1 Rectangular and Non-Rectangular Surface Patches. In *Surfaces in Computer Aided Geometric Design* (edited by R. E. **Barnhill** and W. **Boehm**), pp. 25–33. North-Holland, Amsterdam.

John A. **Gregory** and Jörg M. **Hahn** [1989]. A C^2 Polygonal Surface Patch. *Computer Aided Geometric Design*, vol. 6, p. 1989.

John A. **Gregory** and Jianwei **Zhou** [1999]. Irregular C^2 surface construction using bi-polynomial rectangular patches. *Computer Aided Geometric Design*, vol. 16, pp. 423–435.

Cindy M. **Grimm** and John F. **Hughes** [1995]. Modeling Surfaces of Arbitrary Topology using Manifolds. In *Proc. Computer Graphics and Interactive Techniques, SIGGRAPH'95*, pp. 359–368. ACM Press.

Available online `http://www.cs.wustl.edu/MediaAndMachines/publications/papers/sig95manifolds.pdf`.

Zhidong **Guan**, Jin **Ling**, Ning **Tao**, Xi **Ping** and Tang **Rongxi** [1997]. Study and Application of Physics-Based Deformable Curves and Surfaces. *Computers & Graphics*, vol. 21, no. 3, pp. 305–313.

S. **Guillet** and J.C. **Léon** [1998]. Parametrically Deformed Free-Form Surfaces as Part of a Variational Model. *Computer-Aided Design*, vol. 30, no. 8, pp. 621–630.

Jörg M. **Hahn** [1989]. Geometric Continuous Patch Complexes. *Computer Aided Geometric Design*, vol. 6, pp. 55–67.

William M. **Hsu**, John F. **Hughes** and Henry **Kaufman** [1992]. Direct manipulation of free-form deformations. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 26, pp. 177–184. ACM Press. Also available from `http://citeseer.nj.nec.com/william92direct.html`.

Junji **Ishida** [1997]. The General B-spline Interpolation Method and its Application to the Modification of Curves and Surfaces. *Computer-Aided Design*, vol. 29, no. 11, pp. 779–790.

T. W. **Jensen**, C. S. **Petersen** and M. A. **Watkins** [1991]. Practical curves and surfaces for a geometric modeler. *Computer Aided Geometric Design*, vol. 8, pp. 357–369.

Håvard D. **Jørgensen** [2004]. *Interactive Process Models*. Ph.D. thesis, Norwegian University of Science and Technology.

HoSeok **Kang** and Avi **Kak** [1996]. Deforming virtual objects interactively in accordance with an elastic model. *Computer-Aided Design*, vol. 28, no. 4, pp. 251–262.

Herbert J. **Koelman** [1999]. *Computer Support for Design, Engineering and Prototyping of the Shape of Ship Hulls*. Ph.D. thesis, Delft University of Technology, the Netherlands. Published by SARC BV, `sarc@sarc.nl`.

Herbert J. **Koelman** [2002]. Private communication.

Herbert J. **Koelman** [2003]. Exploring the H-rep Ship Hull Modelling Concept. In *Proceedings of the 2nd International Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT'03)* (edited by V. **Bertram**), pp. 140–153. Hamburg, Germany. Also available at `http://www.sarc.nl/compit03.pdf`.

Herbert Jan **Koelman** [2004]. Method and Device for Designing Surfaces with a Free Form. US Patent no. 6731280. Dutch patent number is 1010452, European patent number is EP 1 003 131 A1.

H.J. **Koelman**, I. **Horváth** and A. **Aalbers** [2001]. Hybrid representation of the shape of ship hulls. *International Shipbuilding Progress*, vol. 48, no. 3, pp. 247–269.

Bruno **Latour** [1987]. *Science in Action, How to follow scientists and engineers through society*. Harvard University Press.

Bryan **Lawson** [1997]. *How Designers Think, the design process demystified*. Third edn. Architectural Press, Oxford, UK.

Francis **Lazarus**, Sabine **Coquillart** and Pierre **Jancène** [1994]. Axial Deformations: an Intuitive Deformation Technique. *Computer-Aided Design*, vol. 26, no. 4, pp. 607–613.

J.C. **Léon** and P. **Trompette** [1995]. A new approach towards free-form surfaces control. *Computer Aided Geometric Design*, vol. 12, no. 4, pp. 395–416.

Charles **Loop** and Tony **DeRose** [1990]. Generalized B-spline Surfaces of Arbitrary Topology. *Computer Graphics, SIGGRAPH '90 Conference Proceedings*, vol. 24, no. 4, pp. 347–356.

Charles T. **Loop** [1994]. A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, vol. 11, no. 3, pp. 303–330.

Charles T. **Loop** and Tony D. **DeRose** [1989]. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics (TOG)*, vol. 8, no. 3, pp. 204–234. `http://doi.acm.org/10.1145/77055.77059`.

Charles Teorell **Loop** [1992]. *Generalized B-spline Surfaces of Arbitrary Topological Type*. Ph.D. thesis, University of Washington. `http://www.research.microsoft.com/~cloop/dissertation.pdf`.

Ron **MacCracken** and Kenneth I. **Joy** [1996]. Free-Form Deformations With Lattices of Arbitrary Topology. In *Computer Graphics Proceedings SIGGRAPH'96*, Annual Conference Series, pp. 181–188. ACM, Addison-Wesley.

Chhandomay **Mandal**, Hong **Qin** and Baba C. **Vemuri** [1998]. A Novel FEM-Based Dynamic Framework For Subdivision Surfaces. Tech. Rep. 021, University of Florida, Computer & Information Science & Engineering, Gainesville, FL, USA. URL `ftp://`

ftp.cise.ufl.edu/cis/tech-reports/tr98/tr98-021.ps.gz or http://citeseer.nj.nec.com/306635.html. See also Mandal *et al.* [2000], which is more polished.

Chhandomay **Mandal**, Hong **Qin** and Baba C. **Vemuri** [2000]. A novel FEM-based dynamic framework for subdivision surfaces. *Computer-Aided Design*, vol. 32, no. 8–9, pp. 479–497. Also published as a technical report in 1998 Mandal *et al.* [1998].

Dinesh **Manocha** and John F. **Canny** [1991]. A New Approach for Surface Intersection. *International Journal of Computational Geometry & Applications*, vol. 1, no. 4, pp. 491–516. URL ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/INTERSECT/IJCGA.pdf.

Martii **Mäntylä** [1988]. *An Introduction to Solid Modeling*. Computer Science Press, Mayland, USA.

Anne L. **Marsan**, Yifan **Chen** and Paul **Steward** [2001]. A Finite Element Approach to Direct Surface Manipulation. In *Proceedings of DETC'01, 2001 ASME Design Engineering Tecnical Conferences*. American Society of Mechanical Engineers. Part of *Design Automation Conference*, paper DETC2001/DAC-21103.

Kevin T. **McDonnell** and Hong **Qin** [2001]. FEM-based Subdivision Solids for Dynamic and Haptic Interaction. In *Proceedings of Sixth ACM Symposium on Solid Modeling and Applications (Solid Modeling 2001)*, pp. 312–313. Available from http://www.cs.sunysb.edu/~ktm/publications.html.

Kevin T. **McDonnell** and Hong **Qin** [2002]. Virtual Clay: Haptics-based Deformable Solids of Arbitrary Topology. In *Proceedings of the Second International Workshop on Articulated Motion Deformable Objects*, Lecture Notes in Computer Science, pp. 1–20. Springer-Verlag. Invited paper. Available from www.cs.sunysb.edu/~ktm/publications.html.

Kevin T. **McDonnell** and Hong **Qin** [2004]. Sculpting, Simulation and Visualization of Dynamic Free-form Solids. *ACM Transactions on Graphics*. Submitted for review. Available from www.cs.sunysb.edu/~ktm/publications.html.

Kevin T. **McDonnell**, Hong **Qin** and Robert A. **Wlodarczyk** [2001]. Virtual clay: A real-time sculpting system with haptic toolkits. In *ACM Symposium on Interactive 3D Graphics*. Also available from http://citeseer.nj.nec.com/mcdonnell01virtual.html.

Kevin Terence **McDonnell** [2003]. *DYNASOAR: DYNAmic Solid Objects of ARbitrary topology*. Ph.D. thesis, Stony Brook University. Available from `www.cs.sunysb.edu/~ktm/publications.html`.

Scott **Meyers** [1996]. *More Effective C++: 35 new ways to improve your programs and designs*. Addison-Wesley professional computing series. Addison-Wesley.

Scott **Meyers** [1998]. *Effective C++: 50 specific ways to improve your programs and designs*. Addison-Wesley professional computing series, second edn. Addison-Wesley.

Scott **Meyers** [2001]. *Effective STL: 50 specific ways to improve your use of the Standard Template Library*. Addison-Wesley professional computing series. Addison-Wesley.

Jacob **Michelsen** [1995]. *A Free-Form Geometric Modelling Approach with Ship Design Applications*. Ph.D. thesis, Techn. Univ. of Denmark, Dept. of Naval Architecture and Ocean Engineering, Lyngby.

Carl **Mitcham** [1994]. *Thinking Through Technology, The Path between Engineering and Phylosophy*. The University of Chicago Press.

J. **Montagnat**, H. **Delingette** and N. **Ayache** [2001]. A review of deformable surfaces: topology, geometry and deformation. *Image Vision Computing*, vol. 19, no. 14, pp. 1023–1040.

A. **Nasri** [1997]. Curve interpolation in recursively generated B-spline surfaces over arbitrary topology. *Computer Aided Geometric Design*, vol. 14, pp. 13–30.

H. **Nowacki**, M.I.G. **Bloor** and B. **Oleksiewicz** (editors) [1995]. *Computational Geometry for Ships*. World Scientific Publishing.

Richard E. **Parent** [1977]. A System for Sculpting 3-D Data. *SIGGRAPH Computer Graphics*, vol. 11, no. 2, pp. 138–147.

Jörg **Peters** [1994]. Constructing C^1 Surfaces of Arbitrary Topology Using Biquadratic and Bicubic Splines. In *Designing Fair Curves and Surfaces* (edited by N. **Sadipadis**), chap. 1, pp. 277–294. SIAM, Philadelphia. Preprint available from `http://www.cise.ufl.edu/research/SurfLab/pre99-papers/9294.pdf`.

Jörg **Peters** [1995a]. Biquartic C^1-surface splines over irregular meshes. *Computer-Aided Design*, vol. 27, no. 12, pp. 895–903. Also available from `http://citeseer.nj.nec.com/peters95biquartic.html`.

Jörg **Peters** [1995b]. C^1 Surface Splines. *SIAM Journal of Numerical Analysis*, vol. 32, no. 2, pp. 645–666. Preprint available from `http://www.cise.ufl.edu/research/SurfLab/pre99-papers/93ffss.pdf`.

Jörg **Peters** [1996]. Curvature Continuous Spline Surfaces over Irregular Meshes. *Computer Aided Geometric Design*, vol. 13, no. 2, pp. 101–131. `http://citeseer.nj.nec.com/peters94curvature.html`.

Jörg **Peters** [2000]. Patching Catmull-Clark Meshes. In *Computer Graphics*, Proceedings SIGGRAPH 2000, pp. 255–258. ACM Press. Also available from `http://www.cise.ufl.edu/research/SurfLab/papers/00pccm.pdf`, see also `http://www.cise.ufl.edu/research/SurfLab/pccm_demo/index.html`.

Jörg **Peters** [2002a]. C^2 free-form surfaces of degree $(3, 5)$. *Computer Aided Geometric Design*, vol. 19, no. 2, pp. 113–126.

Jörg **Peters** [2002b]. Geometric Continuity. In Farin *et al.* [2002], chap. 8, pp. 193–227. Also available as `http://www.cise.ufl.edu/research/SurfLab/papers/01handbook.pdf`.

Jörg **Peters** [2003]. Smoothness, fairness and the need for better multi-sided patches. In *Topics in Algebraic Geometry and Geometric Modeling* (edited by Ron **Goldman** and Rimvydas **Krasauskas**), vol. 334 of *Contemporary Mathematics*. American Mathematical Society. Also available from `http://www.cise.ufl.edu/research/SurfLab/papers/02vilnius.pdf`.

Les **Piegl** and Wayne **Tiller** [1997]. *The NURBS Book*. Monographs in Visual Communication, second edn. Springer Verlag.

William H. **Press**, Saul A. **Teukolsky**, William T. **Vetterling** and Briam P. **Flannery** [1992]. *Numerical Recipes in C, The Art of Scientific Computing*. Second edn. Cambridge University Press.

Hong **Qin** and Demetri **Terzopoulos** [1996]. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 1, pp. 85–96. URL `http://citeseer.nj.nec.com/482831.html` and `http://mrl.nyu.edu/~dt/papers/vcg96/vcg96.pdf`.

Hong **Qin** and Demetri **Terzopoulos** [1997]. Triangular NURBS and their Dynamic Generalizations. *Computer-Aided Geometric Design*, vol. 14, no. 4, pp. 325–347. Also available from the author's homepage `http://www.mrl.nyu.edu/~dt/papers/cagd96/cagd96.pdf`.

David F. **Rogers** and J. Alan **Adams** [1990]. *Mathematical Elements for Computer Graphics*. Second edn. McGraw-Hill.

Malcolm **Sabin** [2002]. Subdivision Surfaces. In Farin *et al.* [2002], chap. 12, pp. 309–325.

SARC [2004]. *PIAS-Fairway Manual*. SARC BV, Eikenlaan 3, 1406 PK, Bussum.

Ramon F. **Sarranga** [1987]. G^1 Interpolation of Generally Unrestricted Cubic Bézier Curves. *Computer Aided Geometric Design*, vol. 4, no. 1–2, pp. 23–39. Errata: Sarranga [1989].

Ramon F. **Sarranga** [1989]. Errata: G^1 Interpolation of Generally Unrestricted Cubic Bézier Curves. *Computer Aided Geometric Design*, vol. 6, no. 2, pp. 167–171.

Donald A. **Schön** [1983]. *The Reflective Practitioner; How Professionals Think in Action*. Maurice Temple Smith Ltd., London. Citations are from the paperback edition first published in 1991 by Ashgate Publishing Ltd, Aldershot, UK.

T. **Sederberg** and S. **Parry** [1986]. Free-form deformation of solid geometric models. *ACM SIGGRAPH '86 Proceedings*, vol. 20, no. 4, pp. 151–160.

Thomas W. **Sederberg**, Jianmin **Zheng**, David **Sewell** and Malcolm **Sabin** [1998]. Non-Uniform Recursive Subdivision Surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 387–394. ACM Press.

Karan **Singh** and Eugene **Fiume** [1998]. Wires: A Geometric Deformation Technique. In *Computer Graphics (SIGGRAPH 98)*, Annual Conference Series, pp. 405–414. ACM.

Jos **Stam** [1998]. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 395–404. ACM Press. Also available at `http://www.dgp.toronto.edu/people/stam/reality/Research/pdf/sig98.pdf`.

A.J. **Stoddart**, A. **Hilton** and J. **Illingworth** [1994]. Slime: a new deformable surface. In *British Machine Vision Conference*, pp. 285–294. BMVA Press, York, England. Also available at `http://citeseer.nj.nec.com/stoddart94slime.html`.

Bjarne **Stroustrup** [1997]. *The C++ Programming Language*. Addison-Wesley/Pearson Education.

Demetri **Terzopoulos** and Hong **Qin** [1994]. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, vol. 13, no. 2, pp. 103–136. URL `http://citeseer.nj.nec.com/terzopoulos94dynamic.html` and `http://mrl.nyu.edu/~dt/papers/tog94/tog94.pdf`.

J. A. **Thingvold** and E. **Cohen** [1990]. Physical Modeling with B-Spline Surfaces for InteractiveDesign and Animation. In *Computer Graphics*, vol. 24, pp. 129–137. ACM.

Bastiaan N. **Veelo** [2004a]. Shape Modification of Hull Models in H-rep. *Schiffstechnik — Ship Technology Research*, vol. 51, no. 4, pp. 162–172.

Bastiaan N. **Veelo** [2004b]. Shape Modification of Hull Models in H-rep. In *Proceedings of the 3rd International EuroConference on Computer Applications and Information Technology in the Maritime Industries* (edited by Volker **Bertram**), pp. 212–222. Also published as Veelo [2004a].

Bastiaan N. **Veelo** [2004c]. Shape Modification of Sculpted Geometric Models of Arbitrary Topology. In *Proceedings of the 8th International Design Conference* (edited by Dorian **Marjanović**), vol. 1, pp. 525–532. Faculty of Mechanical Engineering and Naval Architecture, Zagreb, and The Design Society, Glasgow.

Todd L. **Veldhuizen** [1998]. Arrays in Blitz++. In *Proceedings of the 2nd International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'98)*, Lecture Notes in Computer Science. Springer-Verlag. See also `www.oonumerics.org/blitz/`.

Huawei **Wang**, Kaihuai **Qin** and Ron **Kikinis** [2000]. Exact Evaluation of NURSS at Arbitrary Parameter Values. In *Proceedings of the IASTED International Conference on Computer Graphics and Imaging*, pp. 169–174. Also available from `http://splweb.bwh.harvard.edu:8000/pages/papers/qin/CGIM00.pdf`.

Eric W. **Weisstein** [2004]. Topology. From MathWorld, `http://mathworld.wolfram.com/Topology.html`.

Eric W. **Weisstein** and Todd **Rowland** [2004]. Geodesic. From MathWorld, `http://mathworld.wolfram.com/Geodesic.html`.

W. **Welch** and A. **Witkin** [1992]. Variational Surface Modeling. In *Computer Graphics Proceedings*, Annual Conference Series, pp. 157–166. ACM SIGGRAPH.

Jarke J. **van Wijk** [1986]. Bicubic Patches for Approximating Non-Rectangular Control-Point Meshes. *Computer Aided Geometric Design*, vol. 3, no. 1, pp. 1–13.

Matthew **Wilson** [2004]. *Imperfect C++: Practical Solutions for Real-Life Programming*. Addison-Wesley Professional.

Franz-Heinrich **Wolter** and Séamus T. **Tuohy** [1992]. Curvature Computations for Degenerate Surface Patches. *Computer Aided Geometric Design*, vol. 9, pp. 241–270.

Yasushi **Yamaguchi** [2000]. Differential Properties at Singular Points of Parametric Surfaces. In *CAD Tools and Algorithms for Product Design* (edited by Pere **Brunet**, Christoph M. **Hoffmann** and Dieter **Roller**), pp. 211–221. Springer.

Ibrahim **Zeid** [1991]. *CAD/CAM Theory and Practice*. Computer Science Series, international edn. McGraw-Hill.

Meijing **Zhang** and Hong **Qin** [2001]. Hierarchical D-NURBS Surfaces and Their Physics-based Sculpting. In *SMI 2001 Proceedings, Conference on Shape Modeling and Applications*, pp. 257–266. IEEE.

J. J. **Zheng** and A. A. **Ball** [1997]. Control Point Surfaces over Non-Four-Sided Areas. *Computer Aided Geometric Design*, vol. 14, pp. 807–821.

Jin Jin **Zheng** and Jian J. **Zhang** [2002]. Interactive Deformation of Irregular Surface Models. In *Computational Science — ICCS 2002: International Conference Proceedings, part II* (edited by P.M.A. **Sloot**, C.J. Kenneth **Tan**, J.J. **Dongarra** and A.G. **Hoekstra**), vol. 2330 of *Lecture Notes in Computer Science*, pp. 239–248. Springer-Verlag Heidelberg.

J.J. **Zheng** [2001]. The *n*-Sided Control Point Surfaces without Twist Constraints. *Computer Aided Geometric Design*, vol. 18, pp. 129–134.

Denis **Zorin**, Peter **Schröder** and Wim **Sweldens** [1996]. Interpolating Subdivision for Meshes with Arbitrary Topology. In *SIGGRAPH'96 Computer Graphics Proceedings*, Annual Conference Series, pp. 189–192. ACM.

Index

CSG tree, 65
2D, xv, 64, 93, 108, 111, 126, 134, 135
3D, xv, 17, 54, 63, 72, 93, 96, 143, 157

anchor, 58
ANSI, xv, xviii
arbitrary continuity, *see* G^k
arbitrary topology, **2**, 7, 21, 31, 38
arc length, 116, 122
artifact, 37, **37**

B-rep, xi, xv, xvi, xviii, 38, 52, 53, 56, 57, 59, 61, 66, 119
B-spline, xii, xv, xviii, xxi, 12, 31–36, 40, 42, 55–58, 63, 64, 75–77, 79–81, 86–88, 99, 108, 113, 116, 126, 142, 153, 161
  hierarchical, 32
Bézier curve, 58
base, 98
basis functions, 110
Baumgart, 59
BlueSurf, 15
boolean combination, 61, 65
boundary representation, *see* B-rep
buttock, 62

C++ programming language, xvii, xix, 167–170
C^0, xv, 51
C^1, xii, xv, 41, 51, 76, 83, 85
C^2, xv, 39, 45, 50, 51, 53, 60
CACD, xvi, xvii, 50, 54, 56
CAD, i, xvi, xviii, xix, 1, 2, 15, 26, 28, 38, 44, 49, 60, 148, 168, 169
CAE, xvi, 1, 2
CAGD, xvi, 1, 4, 28, 37, 50, 77, 159
calculus of variations, 122
CAM, xvi, 1–3, 8, 36, 38, 44, 67
CAS, xvi, 39, 45
Casteljau, de, 24, 59
Catmull-Clark, 33
CDRS, xvi, 50, 54, 56
CFD, xvi, 69, 149
chamfer, 61, 65
circle
  great, 121
class A surface, 15–17, 36
class B surface, 15
class C surface, 15
computational fluid dynamics, *see* CFD
computer aided conceptual design, *see* CACD
computer aided design, *see* CAD
computer aided engineering, *see* CAE

computer aided geometric design, *see* CAGD
computer aided manufacturing, *see* CAM
Computer Algebra System, xvi, *see* CAS, 39, 45
concept evaluation, 54
Conceptual Design and Rendering System, *see* CDRS
conic section, 63
constraint, 106
continuity
  arbitrary, *see* G^k
  curvature, *see* G^2 and ϵG^2, 68
  geometric, xvii, 8, 50
  parametric, xv, *see also* C^0, C^1 and C^2, 50
  positional, *see* G^0
  tangent, *see* G^1, 68
  tangent plane, 38, 41
  visual, *see* geometric continuity
control net, 15, **19**, 21, 23, 28, 31
control point, 38, 55–58, 62–65, 67
control polygon, 31, 35
Coons, 51
Coons patch, 38, 50, 51, 55, 66, 68
Cox-deBoor recursion formula, 108
CSG, xvi, xviii, 47, 61, 65, 66, 150
curvature continuity, *see* G^2 and ϵG^2, 17, 68
curvature discontinuity, 59
curve
  plane, 54
curve fitting, 62
curve fairing, 62

D programming language, 170
D programming language, 168
Dassault Systèmes, 15
data structure
  half-edge, 66
  winged edge, 66
data point, xxi
de Casteljau, 24, 40, 59
de-selection field, 105
de-selection field, 105
decay function, 78–80, 94, 96, 97, 107, 126, 137
deformation, 81
  free-form, *see* FFD
  spatial, 72, *see also* FFD
degenerate, 14
design
  feature based, 65
DESIGNBASE, 38
diagonal, 62
differential equation
  Euler-Lagrange, 122, 124
differential geometry, 39
differential topology, 39
direct manipulation, **27**, 55, **58**, 74, 76, 77, 80
Dirichlet parameterisation, 83, 126
discontinuity, 59
  curvature, 59
  geometric, 8
displacement, 81
distance, xxi
D-NURBS, xvi, 73, 74, 86, 87, 95
Doo-Sabin, 34, 35
dynamic NURBS, *see* D-NURBS

edge blend, 61, 65
ϵG^2, xv, xvii, 60, 68, 145
E&S, xvi, 50
Euler-Lagrange differential equation, 122, 124
Evans & Sutherland Computer Corporation, *see* E&S

Extended Pascal programming language, xvii
extent, xxi, 98
extraordinary point, 33, 35

fair, 56, 58
fairing, **8**, 19
  curve, 62
Fairway, xvii, xix, 61, 62, 68, 91, 96, 100, 130, 137, 138, 141, 142, 145, 146, 150, 165, 168, 170
Fast Shape Designer, *see* FSD
FEA, xvii, 84, 85, 95
feature
  surface, 61
feature based design, 65
feature curves, 49
FEM, xvii, 83–86, 126
FFD, xvii, 72, 74, 75, 78–80, 85, 94, 96, 128, 130, 133, 135–137, 140, 142
field
  de-selection, 105
  selection, 101, 104, 105
finite element analysis, *see* FEA
finite element method, *see* FEM
fitting
  curve, 62
flat of bottom, *see* FoB
flat of side, *see* FoS
FoB, xvii, 7, 8, 65, 160
Fortran-77, 57
FoS, xvii, 7, 8, 65, 160
free-form deformation, xvii, *see* FFD, 72, 74, 94, 130
free-form shape, 15
FSD, xvii, 54–56

G-spline, 31
G^0, xv, xvii, 16, 51, 53
G^1, xv, xvii, 16, 31, 35, 39, 40, 43–45, 50, 51, 53, 55, 59, 60, 88, 145, 151, 161
G^2, xii, xv, xvii, 17, 33, 36, 45–47, 58, 68, 99, 139, 140, 145, 151, 160
GC^0, *see* G^0
GC^1, *see* G^1
GC^2, *see* G^2
GC^k, *see* G^k
generic algorithm, 110
generic programming, *see* GP
genus, **xvii**, 76
geodesic, 121, 122, 124, 125
geometric discontinuity, 8
geometric continuity, xv, xvii, 8, 19, 31, 50, 56, 96, 145
G^k, xviii, 39, 45, 46, 151, 160
Gordon surface, 51, 52, 60, 68
GP, xviii, 107, 167, 168, 170
graph theory, 119
Graphical User Interface, *see* GUI
great circle, 121
Gregory patch, 38
GUI, xviii, 96, 100

H-rep, iii, vi, vii, xii, xviii, 61, 91–93, 95, 96, 99, 100, 102, 126, 128, 131, 133, 135, 137, 140, 143–152, 154, 158, 159, 165, 168, 170
half-edge, 66
hierarchical B-spline, 32
history tree, 65
homogeneous coordinates, 64
hyper patch, 72, 80

I-DEAS, 15
I-ship, 57, 61
ICEM Surf, 26
idea generation, xvi, 54, 58

IGES, xviii, 11, 12
Initial Graphics Exchange Specification, *see* IGES
ISO, xviii, 12

knot insertion, 115
knot refinement, 115
knot vector, 115
knuckle line, 8

lattice, **xviii**, 72–75, 80, 85, 94, 96, 130, 135–137, 140
LeSS, xviii, 76, 77
Lesser General Public License, *see* LGPL
LGPL, xviii, 149
lines plans, 61
localised hierarchy surface spline, *see* LeSS
lofted surface, 27

master/slave, 117
master/slave relation, 58
master/slave relationship, 65
MCAD, xviii, 15–17, 27, 38
mechanical computer aided design, *see* MCAD
MODIF, 38
multi-resolution, 44

naval architecture, 19
Newton-Cotes formula, 117
non-uniform rational B-spline, *see* NURBS
NURBS, i, xi, xv, xvi, xviii, 1, 12, 14–22, 24, 27, 28, 45, 49, 54–57, 63–65, 67, 73, 77, 83, 86, 87, 101, 107, 151
NX, 15

object model, xix, 166
object oriented design, *see* OOD
object oriented programming, *see* OOP
offset point, 58
OOD, xix, 169
OOP, xix, 144, 166–170
Open Inventor, 54, 55
OpenGL, 54, 55, 62, 134
organic shape, 15

parametric continuity, xv, *see also* C^0,C^1 and C^2, 50
Parametric Technology Corporation, *see* PTC
Pascal programming language, xvii, 170
patch, xix, **xix**, 1, 7, 14–26, 31, 38–46, 50, 55–57, 59, 60, 62, 66–68, 75–78, 86–89, 144–146, 151, 157
  Coons, 50, 51, 55, 66, 68
  hyper, 72, 80
patch complex, 31
plane curve, 54
poly-curve, 65
polygon net, *see* control net
positional continuity, *see* G^0, 17, 23
Pro|ENGINEER, xix, 15, 50
programming language
  C++, xvii, xix, 167–170
  D, 170
  Extended Pascal, xvii
  Pascal, 170
PTC, xix, 15, 50

quadrilateral, 14
quality
  surface, 19

rational, 64
rectangular, 14
recursion formula, 108

rigid, 27
root
    selection, **119**
ruled surface, 27

S-patch, 40, 41
SARC, xiii, xvii, xix, 2, 7, 61, 91, 165
sculpted shape, 15
selection root, **119**
selection field, xxi, 101, 104–106
SGI, xix, 54, 80
shift, **81**
shift vector, xxi, 98
ship hull, 7, 8
Silicon Graphics, *see* SGI
Simpson's rule, 117
sketch, 49, 50, 52, 54
solid modelling, 61, 62
SolidEdge, 15
SolidWorks, 16
space curve, 54
space tearing, 82, 143
spatial deformation, 72, *see also* FFD, 78
spline
    surface, **31**, 38–46
        localised hierarchy, *see* LeSS
Standard for the Exchange of Product Model Data, *see* STEP
Standard Template Library, *see* STL
station curve, 62
STEP, xi, xix, 12–14
Stewart, Paul. J., 80
STL, xix, 167, 168, 170
surface feature, 61
surface quality, 19
surface spline, **31**, 38–46, 50, 76
    localised hierarchy, *see* LeSS
surface sweep, 27
swept surface, 27

tangent continuity, *see* G^1, 50, 68
tangent plane continuity, 38, 41
tangent ribbon, 53, 56, 59, 66, 146
tangential continuity, 17
taper angle, 66
tensor product, 16
three dimensional, *see* 3D
topology, **2**
    arbitrary, **2**, 7, 21, 38
transfinite interpolants, 50
two dimensional, *see* 2D
typical shift vector, xxi, 98, 106

UGS PLM Solutions, 15
UML, xix, 170
unified modelling language, *see* UML
Unigraphics, 15

valence, **xix**, 26, 33–35, 37–40, 46
VC^0, *see* G^0
VC^1, *see* G^1
VC^2, *see* G^2
VC^k, *see* G^k
VDAFS, xix, 11, 12
visual continuity, *see* geometric continuity

water line, 8, 62
winged edge, 59, 66
wire-frame, 49, 50, 52, 53, 61

Index of Citations

Abt *et al.* [2001], 8, 138
Allan *et al.* [1989], 78, 79
Bézier [1978], 72, 75
Bill [1994], 79
Boehm [2002], 116
Bolz and Schröder [2002], 36
Booch [1991], 165–169
Borrel and Rappoport [1994], 79, 81, 142, 143
Brönnimann [2001], 66
Bright [2002], 168
Catmull and Clark [1978], 33, 34, 73
Celniker and Gossard [1991], 85, 95
Celniker and Welch [1992], 16, 86, 95
Chiyokura and Kimura [1983], 38, 41, 50
Coad and Yourdon [1991], 166
Coquillart [1990], 72, 73, 136
Cotrina Navau and Pla Garcia [2000], 46
Dachille IX *et al.* [2001], 87
Dierckx [1993], 67
Doo and Sabin [1978], 33, 34
Eck and Hoppe [1996], 44
Farin [2002a], 16, 24, 39, 42, 43, 50, 51, 58
Farin [2002b], xvi, 1, 11, 38, 96
Feng *et al.* [1998], 75
Fjeldaas [1985], 66
Foley *et al.* [1990], 1, 11, 96
Forsey and Bartels [1988], 32, 75–77, 86
Fowler and Bartels [1993], 27, 58
Fowler [2003], 170
Gain [2000], 74, 75, 80, 135, 136
Gibson and Mirtich [1997], 72, 85, 86
Gonzalez-Ochoa and Peters [1999], 44, 76, 161
Gonzalez-Ochoa [1997], 44
Gregory and Hahn [1989], 39
Gregory and Zhou [1999], 45, 46
Gregory [1983], 59, 66
Grimm and Hughes [1995], 46
Guan *et al.* [1997], 87
Guillet and Léon [1998], 88
Hahn [1989], 39
Hsu *et al.* [1992], 73, 130, 135
Ishida [1997], 77, 78
Jensen *et al.* [1991], 50–53, 56–60, 66, 68
Jørgensen [2004], 5
Kang and Kak [1996], 86
Koelman *et al.* [2001], 61
Koelman [1999], 3, 7, 53, 61, 63, 65, 66, 68, 69, 91, 95, 104, 117, 144, 145, 154, 159
Koelman [2002], 145
Koelman [2003], 61

Koelman [2004], 91, 149, 170
Latour [1987], 5
Lawson [1997], xvi
Lazarus *et al.* [1994], 82, 143
Loop and DeRose [1989], 40
Loop and DeRose [1990], 40
Loop [1992], 40, 41
Loop [1994], 43
Léon and Trompette [1995], 2, 87, 88, 161
MacCracken and Joy [1996], 73, 85
Mandal *et al.* [2000], 85
Manocha and Canny [1991], 67
Marsan *et al.* [2001], 79, 83, 84, 126
McDonnell *et al.* [2001], 85
McDonnell and Qin [2001], 85
McDonnell and Qin [2002], 85
McDonnell and Qin [2004], 85
McDonnell [2003], 85
Meyers [1996], 168
Meyers [1998], 168
Meyers [2001], 168
Michelsen [1995], 53, 57–61, 65, 66, 154
Mitcham [1994], 4
Montagnat *et al.* [2001], 72
Mäntylä [1988], 66
Nasri [1997], 35, 161
Nowacki *et al.* [1995], 64
Parent [1977], xi, 78, 79, 142
Peters [1994], 41, 43
Peters [1995a], 44, 49, 153
Peters [1995b], 43, 76
Peters [1996], 45
Peters [2000], 45, 160
Peters [2002a], 45, 46, 160
Peters [2002b], 45, 46
Peters [2003], 36, 37, 45, 46
Piegl and Tiller [1997], 1, 11, 12, 96, 115
Press *et al.* [1992], 45
Qin and Terzopoulos [1996], 86
Qin and Terzopoulos [1997], 14, 24, 87
Rogers and Adams [1990], 1, 11, 96
SARC [2004], 65, 138, 141, 142, 146
Sabin [2002], 37
Sarranga [1987], 39
Sarranga [1989], 39
Schön [1983], 5, 7, 158
Sederberg *et al.* [1998], 36
Sederberg and Parry [1986], 72
Singh and Fiume [1998], xii, 82, 83, 143
Stam [1998], 36, 160
Stoddart *et al.* [1994], 40
Stroustrup [1997], 168
Terzopoulos and Qin [1994], 73, 86, 87, 95
Thingvold and Cohen [1990], 87
Veelo [2004a], 91, 177
Veelo [2004b], 92
Veelo [2004c], 92
Veldhuizen [1998], 168
Wang *et al.* [2000], 36
Weisstein and Rowland [2004], 121, 122
Weisstein [2004], 2
Welch and Witkin [1992], 86, 88
Wilson [2004], 168
Wolter and Tuohy [1992], 25
Yamaguchi [2000], 25
Zeid [1991], xv, 1, 11, 96
Zhang and Qin [2001], 87
Zheng and Ball [1997], 41, 86
Zheng and Zhang [2002], 86
Zheng [2001], 41
Zorin *et al.* [1996], 35
van Dijk [1994], 54–56
van Wijk [1986], 39

Free Software Foundation [2000], xviii, 149

Colophon

This thesis was typeset using the LaTeX typesetting system created by Donald E. Knuth and Leslie Lamport, and the memoir class created by Peter R. Wilson. Bastiaan N. Veelo designed the chapter page lay-out. The body text is set in 10 point Adobe Palatino with a 12 point leading, provided by the PX Fonts package by Young Ryu, on a measure of 304 points. Line drawings were programmed in the MetaPost graphics language and compiled into Postscript by the `mpost` program, created by John D. Hobby.