

# Minimizing Curvature Variation for Aesthetic Surface Design

*Pushkar Prakash Joshi*

Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2008-129

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-129.html>

October 7, 2008



Copyright 2008, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

# Minimizing Curvature Variation for Aesthetic Surface Design

by

Pushkar Prakash Joshi

B.S. (University of Southern California) 2002

M.S. (University of California, Berkeley) 2007

A dissertation submitted in partial satisfaction  
of the requirements for the degree of

Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Carlo Séquin, Chair

Professor Jonathan Shewchuk

Professor Sara McMains

Fall 2008

The dissertation of Pushkar Prakash Joshi is approved.

---

Chair

Date

---

Date

---

Date

University of California, Berkeley

Fall 2008

Minimizing Curvature Variation for Aesthetic Surface Design

Copyright © 2008

by

Pushkar Prakash Joshi

# Abstract

## Minimizing Curvature Variation for Aesthetic Surface Design

by

Pushkar Prakash Joshi

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Carlo Séquin, Chair

We investigate the usability of functional surface optimization for the design of free-form shapes. The optimal shape is subject to only a few constraints and is influenced largely by the choice of the energy functional. Among the many possible functionals that could be minimized, we focus on third-order functionals that measure curvature variation over the surface.

We provide a simple explanation of the third-order surface behavior and decompose the curvature-variation function into its Fourier components. We extract four geometrically intuitive, parameterization-independent parameters that completely define the third order shape at a surface point. We formulate third-order energy functionals as functions of these third-order shape parameters.

By computing the energy minimizers for a number of canonical input shapes, we provide a catalog of diverse functionals that span a reasonable domain of aesthetic styles. The

functionals can be linearly combined to obtain new functionals with intermediate aesthetic styles. Our side-by-side tabular comparison of functionals helps to develop an intuition for the preferred aesthetic styles of the functionals and to predict the aesthetic styles preferred by a new combination of the functionals.

To compare the shapes preferred by the functionals, we built a robust surface optimization system. We represent shapes using Catmull–Clark subdivision surfaces, with the control mesh vertices acting as degrees of freedom for the optimization. The energy is minimized by an off-the-shelf implementation of a quasi-Newton method. We discuss some future work for further improving the optimization system and end with some conclusions on the use of optimization for aesthetic design.

---

Professor Carlo Séquin  
Dissertation Committee Chair

# Contents

<b>Contents</b>	<b>i</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>5</b>
2.1 Optimization for Shape Design . . . . .	5
2.2 Optimization for Surface Fairing . . . . .	7
<b>3 Intuitive Exposition of Third-Order Surface Behavior</b>	<b>10</b>
3.1 Introduction . . . . .	10
3.2 Previous Studies of Third-Order Surface Behavior . . . . .	14
3.3 Third-Order Parameters for a Polynomial Height Field . . . . .	15
3.4 Fourier Analysis of Quadratic Height Function . . . . .	17
3.5 Fourier Analysis of Cubic Height Function . . . . .	18
3.6 Computing Fourier Components for a General Surface Patch . . . . .	20
3.7 Qualitative Description of the Fourier Components . . . . .	24
3.7.1 Expressing Cross Derivatives Using Third-Order Shape Parameters .	25
3.7.2 Expressing Normal Curvature Derivatives in Arbitrary Directions Us- ing Third-Order Shape Parameters . . . . .	28
3.7.3 Application: Classification of Umbilics . . . . .	29
3.8 Summary . . . . .	30
<b>4 Functionals</b>	<b>32</b>
4.1 Requirements of Surface Energy Functionals . . . . .	32



4.2	How to Construct Functionals . . . . .	34
4.2.1	First-Order Functional . . . . .	34
4.2.2	Second-Order Functionals . . . . .	35
4.2.3	Third-Order Functionals . . . . .	39
4.3	Combining Energy Functionals . . . . .	45
4.4	Scale Invariance of Functionals . . . . .	46
<b>5</b>	<b>Surface Representation</b>	<b>49</b>
5.1	Catmull–Clark Subdivision Surfaces . . . . .	50
5.1.1	Removing $C^2$ Discontinuity by Blending . . . . .	51
5.1.2	Boundary Patches . . . . .	53
5.1.3	Maintaining Sharp Features in Input Surfaces . . . . .	54
<b>6</b>	<b>Optimization System</b>	<b>55</b>
6.1	Energy Computation . . . . .	56
6.1.1	Pre-processing . . . . .	56
6.1.2	Computing Surface Energy and Gradient . . . . .	58
6.2	Optimization . . . . .	59
6.2.1	Input . . . . .	60
6.2.2	Increasing Degrees of Freedom . . . . .	60
6.2.3	Optimization Algorithms . . . . .	61
<b>7</b>	<b>Options for Fast Optimization</b>	<b>64</b>
7.1	Discrete Geometry Operators for Energy Queries . . . . .	64
7.2	Addressing Ill-Conditioned Functionals . . . . .	67
7.2.1	Sobolev Gradients . . . . .	69
<b>8</b>	<b>Comparison of Functionals</b>	<b>71</b>
8.1	Experiments on a Torus . . . . .	72
8.1.1	Calibrating Weights for Combined Functionals . . . . .	74
8.2	Comparison of Third-Order Energies . . . . .	75
8.3	Comparison with MVS Energies . . . . .	77
8.4	Comparison with Second-Order Energy . . . . .	80
8.5	Example of Aesthetic Design: Vase . . . . .	83
8.6	Combining Second-Order and Third-Order Energies . . . . .	85

<b>9</b>	<b>Summary, Conclusions, and Future Work</b>	<b>88</b>
9.1	Summary . . . . .	88
9.2	Conclusions . . . . .	89
9.3	Future Work . . . . .	89
	<b>Bibliography</b>	<b>91</b>

## Acknowledgements

I owe my advisor Carlo Séquin a big debt of gratitude for his never-ending guidance, teaching, and enthusiasm through all five years of my research at Berkeley. I also thank Jonathan Shewchuk and Sara McMains for their useful feedback on my thesis.

A special thanks goes out to Eitan Grinspun who invested so much of his time to teach me everything I know about discrete operators and multiresolution preconditioners. Also, Denis Zorin was very generous with helpful code and papers, for which I am grateful.

My favorite research experiences came while working in industry. I thank Tony DeRose and Mark Meyer for giving me the priceless opportunity to work on an ongoing research project at Pixar. I also thank Nathan Carr, Radomir Mech and the rest of the group at Adobe for not only giving me the chance to contribute to a great research project, but also for believing in me enough to induct me full-time into their group.

A big thank-you goes out to my family (Mom, Dad, Hari, Leslie) who gave their love and support throughout my Ph.D. despite not understanding why I liked making blobby shapes. Also, to my friends, both from 'SC and Berkeley: you all are awesome.

And of course, the thesis would not have been possible without the constant encouragement, love, technical guidance and great cooking from Hayley Iben. Thanks to you, finally... the thesis is complete!



# Chapter 1

## Introduction

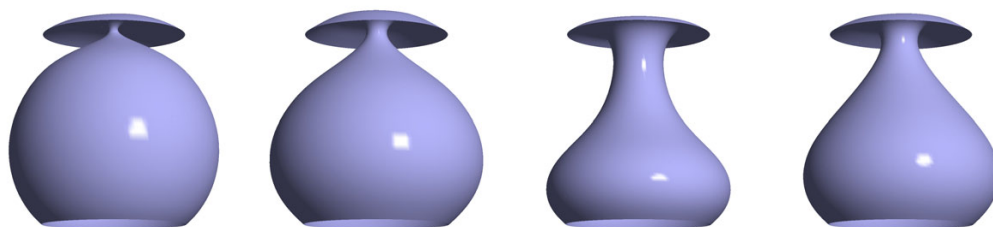


Figure 1.1. We show the optimal shapes obtained by minimizing four different surface functionals. For a detailed discussion of this example, see Section 8.5.

Aesthetically pleasing smooth surfaces are used to generate computer-aided sculpture, to build conceptual models of consumer products, to design mechanical parts (e.g. ship hulls, car hoods), and to fair rough, bumpy surfaces (like noisy point clouds obtained from a range scan). In a typical shape design task, a designer starts with a surface that approximates a desired shape, along with geometric constraints that must be satisfied by the desired shape. By varying parameters that control the shape of the surface, the designer constructs an aesthetically pleasing surface. In most design tasks of practical importance, there are too many control vertices or shape parameters to modify by hand to achieve a smooth desired shape. Instead, designers use numerical optimization to construct the surface. A computer algorithm deforms the initial surface into an aesthetically pleasing,

smooth surface by adjusting the degrees of freedom such that they minimize a functional (a geometric function that maps the surface to a scalar value). In most shape design tasks, optimization is performed as the last step of the process. That is, the input surface is already close in shape to the desired surface, so optimization is used merely to smooth out any unwanted bumps while maintaining the overall shape.

In this thesis, we investigate a different application of surface optimization. We demonstrate the use of surface optimization as a shape *design* tool. That is, we show how a designer can use surface optimization early in the design phase to produce an aesthetically pleasing shape that was not conceived manually.

The surface functional strongly influences the nature of the resulting optimal shapes. Given an initial surface and constraints, optimizing different functionals will result in different optimal shapes. Therefore, it makes sense to provide a *collection* of functionals so that the designer can select one functional to fit the design task. Selecting the proper functional requires an intuitive understanding of the types of shape characteristics favored and penalized by the different functionals. In this thesis, we develop an intuition for the optimal shapes of functionals by comparing their respective minimizers for canonical input surfaces.

The functionals that we have developed measure purely geometric properties and will ignore the influence of external forces and material properties. The functionals are also independent of surface parameterization, scale, and rigid transformations. Our goal is to find a set of functionals that, upon optimization, yield different yet aesthetically pleasing shapes. In the past, researchers have relied mostly on functionals that measure first-order or second-order differential properties (i.e. surface area or bending energy, respectively) to

produce aesthetically pleasing shapes. In this thesis, we consider third-order functionals that include curvature derivatives.

We begin by providing a novel, intuitive exposition of third-order shape behavior — we describe the behavior of the curvature derivative function at a surface point independent of the point’s coordinate system. By minimizing energies that measure curvature variation, we create aesthetically pleasing, high-quality shapes (see Figure 1.1 for an introductory example). We argue for the superiority of third-order functionals over second-order functionals for aesthetic design. We also combine second-order and third-order energies to obtain functionals with combined preferred shapes.

We formulate surface energy functionals by combining differential geometric terms up to the given order. However, it is not necessary to study all the functionals that can be formulated. Compared to the number of functionals that we can formulate, the number of geometric parameters that fully describe the surface behavior up to a given order is small. Therefore, we formulate fundamental functionals that measure surface beauty up to a given order by combining the corresponding geometric parameters: principal normal curvatures (second-order) and Fourier coefficients of the normal curvature derivative function (third-order). We provide four functionals (one second-order functional and three third-order functionals) whose optimal shapes span the range of shapes that can be produced by minimizing second-order and third-order energies. All of our functionals yield aesthetically pleasing but different optimal shapes. Given these basic functionals, a designer can compose new functionals with different preferred shapes by weighted combinations of the basic functionals.

Optimizing surface functionals over a complicated surface is not an easy task. While we

use and recommend off-the-shelf optimization code, it is crucial to select other components of the optimization system so that we can obtain the minimizers at reasonable computational cost. As a reference for future surface optimization system builders, we describe our entire optimization system in detail.



## Chapter 2

# Related Work

We describe previous work in surface design that uses numerical optimization to produce a smooth shape. We can classify the previous work into two categories: one that uses surface optimization to produce a novel shape that was not conceived manually, and the more common category of work that uses optimization to fair (de-noise or smooth out) a given rough surface while maintaining its overall shape.

### 2.1 Optimization for Shape Design

As discussed in Chapter 1, novel aesthetically pleasing shapes can be produced by the optimization of a surface functional. In computer-aided design literature, the most commonly found functional is the second-order bending energy functional (described in Section 4.2.2 in detail). For example, Hsu et al. [HKS92] studied and catalogued the second-order bending energy minimizers for the unconstrained, closed input surfaces of genus zero to five. The emphasis in their work was on the mathematical properties of the bending energy minimizers and not on assessing the suitability of the bending energy functional

for aesthetic design. Around the same time, Rando and Roulier [RR91] demonstrated the notion that different functionals yield different shapes. Their paper describes second-order and third-order functionals for surface design using mean and Gaussian curvatures and their derivatives. Rando and Roulier’s experiments functionals focused only on small surface patches, not on complicated or high-genus surfaces. Therefore, it was difficult to assess the suitability of their functionals for shape design. In this thesis, we use functionals built from fundamental geometric principles and provide a description of their preferred shapes.

Our work follows the work of Moreton [Mor93] who introduced the third-order “Minimum Variation Surface” (MVS) functional. See the paper by Moreton and Séquin [MS92] for a concise description of the MVS functional. MVS optimization minimizes the variation of principal curvatures along their corresponding principal directions. In a subsequent Master’s thesis [Jos07], we enhanced the MVS functional by introducing the  $\text{MVS}_{\text{cross}}$  functional and compared the preferred shapes of the bending energy, MVS energy and  $\text{MVS}_{\text{cross}}$  energy. A more complete functional than MVS or  $\text{MVS}_{\text{cross}}$  was introduced by Mehlum and Tarrou [MT98] that measures the average magnitude of the arc-length derivative of normal curvature (see Section 4.2.3 for more details). There was little discussion of the nature of the preferred shapes of the Mehlum–Tarrou functional in [MT98], but we provide one in this thesis. Finally, Gravesen [GU01, Gra03] presented eighteen third-order surface invariants, each of which can be used as a functional. The invariants are formulated as functions of the coordinate-system dependent first-order parameters (coefficients of the first fundamental form), the second-order parameters (coefficients of the second fundamental form) and third-order parameters (covariant derivatives of the second fundamental form). Similar to Mehlum and Tarrou’s work [MT98], Gravesen did not compute the shapes preferred by

the invariants listed in [GU01, Gra03] — the papers were about the algebra of third-order differential operators. We address the Gravesen functionals further in Section 4.2.3.

## 2.2 Optimization for Surface Fairing

The use of optimization for fairing a shape is more common and has a richer history than the use of optimization for conceiving a shape. Kjellander [Kje83] was one of the first researchers to describe a system for smoothing a B-spline patch network by minimizing an approximation of the second-order bending energy. Bloor and Wilson [BW90] introduced the concept of solving elliptic partial differential equations (PDEs) for surface fairing; for example, solving the Euler–Lagrange equation of the bending energy functional can be used to obtain a fair surface that satisfies the given boundary conditions. Kobbelt et al. [KCVS98] extended Bloor and Wilson’s approach to use triangle meshes to solve PDEs. Schneider and Kobbelt [SK01] extended the system from [KCVS98] to handle  $G_1$  boundary constraints. Xu et al. [XPB06] showed how we can solve non-geometric problems (like texture image sharpening) as PDEs over the surface domain. In a follow-up paper, Xu and Zhang [XZ07] solved sixth-order partial differential equations (the Euler–Lagrange equation of a third-order functional measuring mean curvature variation). A common use of surface fairing is for removing noise from a range scan data set. Towards that application, Desbrun et al. [DMSB99] described a triangle mesh-based system that uses implicit time integration for solving the elliptic PDE to yield a smooth mesh.

The work of Desbrun et al. [DMSB99] was one of several papers to use the currently popular “discrete differential geometry” operators that provide simple expressions for local geometric properties like normal curvature in terms of mesh vertices and edges. Unlike the

“vertex-based” discrete operator in Desbrun et al. [DMSB99], Bridson et al. [BMF03] and Grinspun et al. [GHDS03] introduced an “edge-based” discrete operator for computing the mean curvature across an edge of a triangle mesh. While discrete operators are not new (energy queries in Brakke’s “Surface Evolver” [Bra92] used discrete operators) and number of papers on discrete operators in the literature is vast, we point the reader to recent work that describes operators that maintain topological invariants [BS05] and those that are more robust to bad mesh quality [GGRZ06]. The papers listed in this section describe different options for computing energy over a surface. However, none of the papers were intended to *design a novel surface* — the main application was de-noising, and in some cases, simple hole-filling.

In this thesis, we will not discuss in detail any methods that convert the non-linear surface optimization problem into a linear one by building a quadratic approximation — the central idea of the approach is to introduce interactivity in the surface modeling at the cost of accuracy. The quadratic approximation depends on non-geometric information like parameterization [CG91, WW92] or on a separately defined reference surface (Greiner’s “data-dependent” approach [Gre94]). All these approximations may be suitable for surface fairing, but not for the more difficult problem of shape design using optimization. When we want to construct a novel shape from the energy optimization of an initial shape, we can use only minimal information from the initial shape: topological type (genus), symmetry, and constraints (if any). The optimal shape may be significantly different from the initial shape, and the optimization must be performed with as much independence from the initial shape as possible. Therefore, we cannot assume that the user-provided parameterization remains constant (and thus cannot use quadratic approximations to the energy functionals, as was done by Celniker and Gossard [CG91] and Welch and Witkin [WW92]) or that the

optimal shape’s relation to another fixed surface remains constant (and thus cannot use Greiner’s [Gre94] data-dependent approach).

There is a vast amount of literature in the use of non-linear surface optimization for tasks other than aesthetic design. Some important examples include the simulation of elasticity of cell membranes [SBL91] and of the interface between two different liquids [CCF91]. Surface optimization is also used for design tasks that consider external factors (like shape design that considers air drag [EP97]). Since we focus on the use of optimization for aesthetic design, we will not discuss the other applications in more detail.

## Chapter 3

# Intuitive Exposition of Third-Order Surface Behavior

As mentioned in Chapter 1, our third-order functionals are formulated by combining the parameters that describe third-order surface behavior. In this chapter, we provide an explanation of third-order surface behavior and list the four parameters that completely define third-order shape. Besides surface optimization, our third-order shape parameters are useful for any application that requires a concise description of the third-order behavior at a surface point.

### 3.1 Introduction

Surface analysis (also known as shape interrogation) is a useful tool for understanding the geometric behavior of a surface near a given point. In the general case of a smooth surface, one can analyze its geometry up to a given order by performing a Taylor expansion of the surface. As an example, the zero<sup>th</sup>-order surface analysis near a given point yields the position of that point. The first-order analysis adds the tangent plane, the second-order

the curvature tensor, and the third-order a rank-3 tensor that describes the derivatives of curvature. The higher the order of surface analysis, the more information about the shape is extracted.

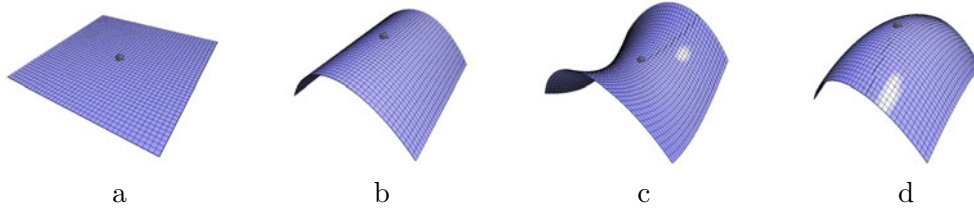


Figure 3.1. Up to second order, we can intuitively classify a surface point as (a) flat, (b) parabolic, (c) hyperbolic, and (d) elliptic.

Surface analysis using Taylor expansion produces shape information that is compactly stored in tensors. To extract this information from the tensors, we must formulate an input query in the tensor's coordinate system. For instance, consider the second-order curvature tensor. The curvature tensor is a rank-2 tensor, which means it takes two vectors as input and produces the normal curvature in the direction specified by the vectors. To compute the normal curvature in a given direction at a surface point, we first express the direction as a vector in the point's tangent plane, provide the same vector as both inputs to the curvature tensor, and re-scale the result by the area metric (multiply by the inverse of the first fundamental form). We perform a similarly complicated sequence of operations to extract derivatives of surface curvature; we need to provide three directions to the rank-3 tensor that encapsulates the curvature derivative information. Extracting precise shape information at a surface point thus requires us to understand how to query the shape tensors at that point.

However, most people, especially novices to linear algebra, are more apt to extract shape information from simple geometric primitives. For instance, up to second order we can easily classify a surface point as flat, elliptic, hyperbolic or parabolic (Figure 3.1), without having

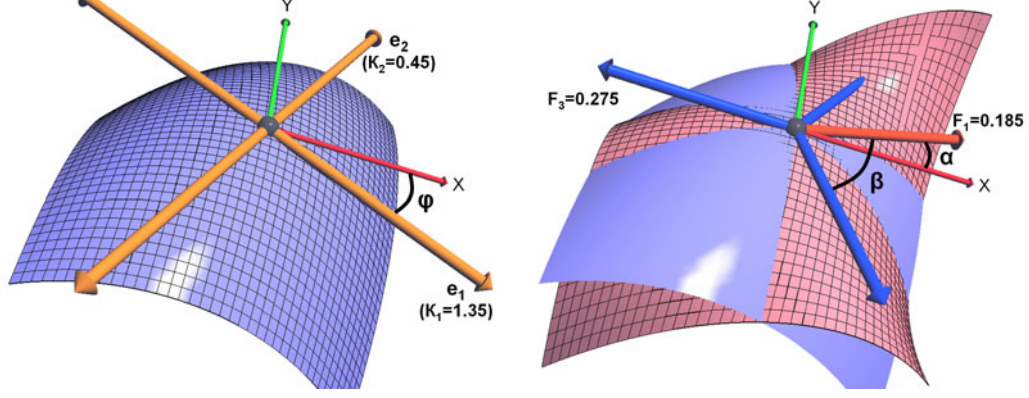


Figure 3.2. The above figures show the parameters that fully describe the second-order (left) and third-order (right) shape behavior. All vectors are in the tangent plane of the point of analysis and are unit vectors.

**Left:** Second-order frame comprised of principal directions and their associated principal curvatures. The angle  $\phi$  indicates the rotation of the frame from the user provided  $x$ -axis in the tangent plane. The entire second-order behavior is described by three numbers:  $\kappa_1$ ,  $\kappa_2$  and  $\phi$ .

**Right:** Third-order frame comprised of four directions: one indicating the peak of the first Fourier component and the other three indicating equally spaced peaks of the third Fourier component. Angle  $\alpha$  indicates the rotation of the frame from the user provided  $x$ -axis, and angle  $\beta$  indicates the rotation of the third Fourier component from the first Fourier component. The entire third-order behavior is described by four numbers:  $F_1$ ,  $F_3$ ,  $\alpha$  and  $\beta$ . The cubic surface in pink (with the grid) is superimposed on the original quadratic surface in blue (without the grid) to show the undulatory third-order behavior.

to query the curvature tensor. Instead, we can extract the principal curvatures  $\kappa_1$  and  $\kappa_2$  (maximum and minimum values of the normal curvatures at the surface point) from the curvature tensor, and use just those two scalar values to intuitively classify the second-order behavior of a surface point. When the product  $\kappa_1\kappa_2$ , also known as the Gaussian curvature, is positive, negative, or zero, the surface is elliptic, hyperbolic, or parabolic, respectively. In the special case where  $\kappa_1$  and  $\kappa_2$  are equal, the surface point is umbilic. Of course, when both  $\kappa_1$  and  $\kappa_2$  are zero, the surface is flat. Euler's theorem tells us that the principal directions ( $e_1$  and  $e_2$ ) corresponding to the principal curvatures ( $\kappa_1$  and  $\kappa_2$  respectively) are mutually orthogonal. In fact, we can completely describe the second-order shape of a surface point by three intuitive parameters: the two principal curvatures,  $\kappa_1$  and  $\kappa_2$ , and the



angle  $\phi$  made by the  $e_1$  principal direction with an arbitrary direction in the tangent plane (Fig. 3.2). At an umbilic point, all normal curvatures are equal, therefore, the principal curvatures and principal directions are not defined.  $\kappa_1$ ,  $\kappa_2$ , and  $\phi$  represent exactly the same information as that in the curvature tensor, but are more accessible to novices and to visual and geometrical thinkers. We believe that this geometrical analysis results in a more intuitive and widespread understanding of second-order shape behavior.

For many surface design tasks, geometric analysis only up to second order is not sufficient because it ignores too much shape behavior. Therefore, we need to study and understand higher-order shape behavior. As one step towards that goal, we focus on third-order analysis. We have not been able to find an intuitive description for third-order surface behavior in the differential geometry literature — all the third-order surface analysis we have seen so far uses the algebra of rank-3 tensors. As a result, a thorough understanding of third-order surface behavior is typically limited to those people who are comfortable with tensor algebra.

**Contribution** in this chapter, we provide an intuitive, geometric description of third-order surface behavior. Our description is similar in its intuitive nature to the readily accessible second-order description using principal curvatures and directions. We extract four shape parameters that completely describe the third-order shape behavior at a surface point. Our shape parameters are independent of any coordinate system and are obtained by decomposing the third-order shape function into its Fourier components.

### 3.2 Previous Studies of Third-Order Surface Behavior

While not as commonly studied as second-order surface behavior, third-order surface behavior has been studied for selected applications. In computer graphics, the most common application is to convey shape information via line drawings such as suggestive contours [DFRS03] or other salient features such as perceptually-based curvature extrema [WB01]. Rusinkiewicz [Rus04] describes how the construction of the rank-3 tensor can be used to interrogate the derivatives of normal curvature in arbitrary directions. These curvature derivatives provide shape information that is perceptually important to the visual system. While the rank-3 tensor yields precise curvature derivative information, it is not easy to understand.

As explained in Section 2.1, surface designers optimize third-order surface energies to produce smooth surfaces used in computer-aided geometric design. Formulating such energies typically requires understanding some aspect of third-order surface behavior. For instance, Mehlum and Tarrou [MT98] formulate an expression that provides the arc-length derivative of the normal curvature in a given direction. They introduce four third-order shape parameters,  $P$ ,  $Q$ ,  $S$ ,  $T$ . These terms essentially encode the normal components of parametric surface derivatives. While useful for computing the energy values, these parameters do not easily provide a qualitative description of the third-order shape at a given point because they depend on the particular parameterization used at that point.

Umbilic points (surface points with equal principal curvatures) have received a lot of third-order analysis. Understanding the behavior of a surface near umbilics is useful for manufacturing thin shell parts [MWP96] and studying geometrical optics [BH77]. As a result, numerous researchers have explored the exact geometric nature of umbilic points.

A common method of characterizing an umbilic point is Darboux’s classification according to the pattern of lines of curvature near the point (*star*, *monstar* and *lemon* — see [BH77] for a visual description and [Por01] for a detailed description). Maekawa et al. [MWP96] analyze the local surface geometry near an umbilic point to compute curvature lines that pass through that point. The initial setup for the surface analysis near the umbilic point is similar to ours, but further analysis focuses on the umbilic classification and lacks the intuitive, qualitative description we seek.

In a nutshell, previously, researchers have extensively studied specific aspects of third-order surface behavior corresponding to particular applications, but an *intuitive, purely geometric description is missing*. Informally speaking, the “algebra of third-order behavior” has been studied sufficiently; the “geometry of third-order behavior” needs to be raised to a corresponding level of understanding. We hope that the following exposition serves as a significant step towards that goal.

### 3.3 Third-Order Parameters for a Polynomial Height Field

To introduce the intuition behind the necessary mathematical concepts, we will restrict our attention to a smooth surface patch centered at a given point. Assume that the surface near the point is fully described by a third-order height field above the tangent plane at that point. The height field is a function of the two independent variables  $x$  and  $y$  such that the  $x$ - $y$  coordinate frame forms a parameterization of the surface near the point. The

height is defined by

$$\begin{aligned}
z(x, y) &= C_0 x^3 + C_1 y^3 + C_2 x^2 y + C_3 x y^2 \\
&+ Q_0 x^2 + Q_1 y^2 + Q_2 x y \\
&+ L_0 x + L_1 y + K.
\end{aligned} \tag{3.1}$$

We assume that the directions corresponding to  $x$  and  $y$  are mutually orthogonal and that the first-order ( $L_0, L_1$ ) and constant parameters ( $K$ ) are zero. This assumption is an oversimplification and is not always valid for a surface patch. However, we found it easier to first develop an intuition for the third-order parameters using this restricted analysis of the patch. In Section 3.6 we describe how to extract the third-order shape parameters for a general surface patch.

As a first step, we convert the cubic height function  $z(x, y)$  into polar coordinates  $z_p(r, \theta)$ , where  $r = \sqrt{x^2 + y^2}$  and  $\theta = \tan^{-1}(y/x)$ . We then separate the height field function (Equation 3.1) into two equations that describe only the second-order (quadratic) and third-order (cubic) behavior:

$$z_{p_q}(r, \theta) = r^2 [Q_0 \cos^2 \theta + Q_1 \sin^2 \theta + Q_2 \cos \theta \sin \theta] \tag{3.2}$$

$$z_{p_c}(r, \theta) = r^3 [C_0 \cos^3 \theta + C_1 \sin^3 \theta + C_2 \cos^2 \theta \sin \theta + C_3 \cos \theta \sin^2 \theta]. \tag{3.3}$$

Previous work follows a similar setup up to this step. At this point, people solve for the extremal values of  $\theta$  by solving the quadratic equation  $\frac{dz_{p_q}(r, \theta)}{d\theta} = 0$  and cubic equation  $\frac{dz_{p_c}(r, \theta)}{d\theta} = 0$  (e.g. see [MT98, MWP96]). The roots of the quadratic equation yield the principal curvature directions. The number of real roots of the cubic equation (1 or 3) and their distribution with respect to each other is used to classify umbilic points or to study maxima of curvature variation. We have obtained a more intuitive understanding

of the third-order behavior by decomposing the functions  $z_{p_q}$  and  $z_{p_c}$  into their Fourier components.

### 3.4 Fourier Analysis of Quadratic Height Function

As an introductory exercise, we analyze the Fourier components of the quadratic height function and show how the amplitudes and phase shifts of the Fourier components yield the well-known second-order shape parameters. The Fourier components of the functions that comprise  $z_{p_q}(r, \theta)$  can easily be extracted:

$$\cos^2 \theta = 0.5 + 0.5 \cos 2\theta, \quad (3.4)$$

$$\sin^2 \theta = 0.5 - 0.5 \cos 2\theta, \quad (3.5)$$

$$\cos \theta \sin \theta = 0.5 \sin 2\theta. \quad (3.6)$$

Therefore,  $z_{p_q}$  can be expressed as a constant term plus a linear combination of the Fourier components  $\cos 2\theta$  and  $\sin 2\theta$ , which can be further simplified as an equation using a single phase-shifted cosine function. That is,

$$z_{p_q}(r, \theta) = r^2[F_0 + F_2 \cos(2(\theta + \phi))], \quad (3.7)$$

where  $F_0$  represents the mean value of  $z_{p_q}$ , and  $F_2$  represents the amplitude of the cosine component that gets added to the mean. The cosine term is a symmetric function that produces four equally spaced extremal values in the range  $[0, 2\pi)$ . The maxima and minima correspond to the well-known principal curvatures and the mutually orthogonal principal directions. The angle  $\phi$  is the phase shift that is measured with respect to an arbitrary, user-provided direction (usually the  $x$ -axis or the  $u$ -direction). Therefore, the entire second-order shape information can be compactly described in a parameterization-independent manner

by three terms ( $F_0$ ,  $F_2$  and  $\phi$ ). By computing  $\kappa_1 = F_0 + F_2$  and  $\kappa_2 = F_0 - F_2$  we get the three familiar terms:  $\kappa_1$ ,  $\kappa_2$ ,  $\phi$ . At an umbilic point, the  $F_2$  component is zero.

### 3.5 Fourier Analysis of Cubic Height Function

Similar to the quadratic height function, we extract the Fourier components of the functions that make up  $z_{pe}(r, \theta)$ :

$$\cos^3 \theta = 0.75 \cos \theta + 0.25 \cos 3\theta, \quad (3.8)$$

$$\sin^3 \theta = 0.75 \sin \theta - 0.25 \sin 3\theta, \quad (3.9)$$

$$\cos^2 \theta \sin \theta = 0.25 \sin \theta + 0.25 \sin 3\theta, \quad (3.10)$$

$$\cos \theta \sin^2 \theta = 0.25 \cos \theta - 0.25 \cos 3\theta. \quad (3.11)$$

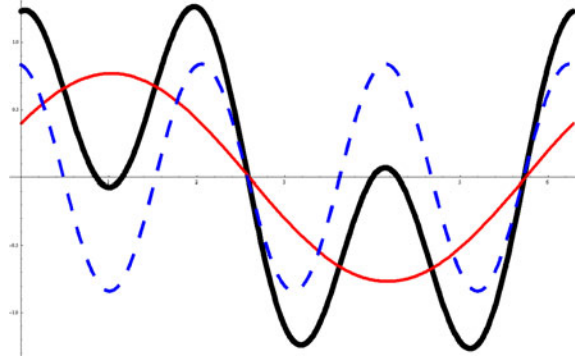


Figure 3.3. Third-order height function from Eqn. 3.3 (thick black) is a sum of two cubic sinusoidal height functions:  $\cos \theta$  (solid red) and  $\cos 3\theta$  (dashed blue)

The cubic shape function  $z_{pe}$  can then be expressed as a linear combination of two Fourier components,  $\cos \theta$  and  $\cos 3\theta$  by the function

$$z_{pe}(r, \theta) = r^3 [F_1 \cos(\theta + \alpha) + F_3 \cos 3(\theta + \delta)], \quad (3.12)$$

where  $F_1$  and  $F_3$  are the amplitudes of the Fourier components, and  $\alpha$  and  $\delta$  are the phase shifts from the x-axis. Instead of the x-axis, we could pick an arbitrary, user-provided

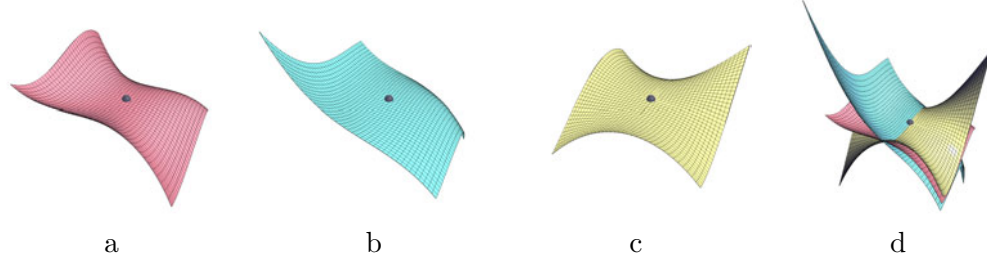


Figure 3.4. The third-order surface is a combination of two sinusoidal functions ( $\cos \theta$  and  $\cos 3\theta$ ) which are the Fourier components of the third-order shape function. We show (a) the original cubic surface, (b) only the first Fourier component, (c) only the third Fourier component, and (d) the original cubic surface sandwiched between constituent Fourier components with twice their original amplitudes. Clearly, the cubic surface is the average of the twice the Fourier components, and therefore is equal to the sum of the Fourier components.

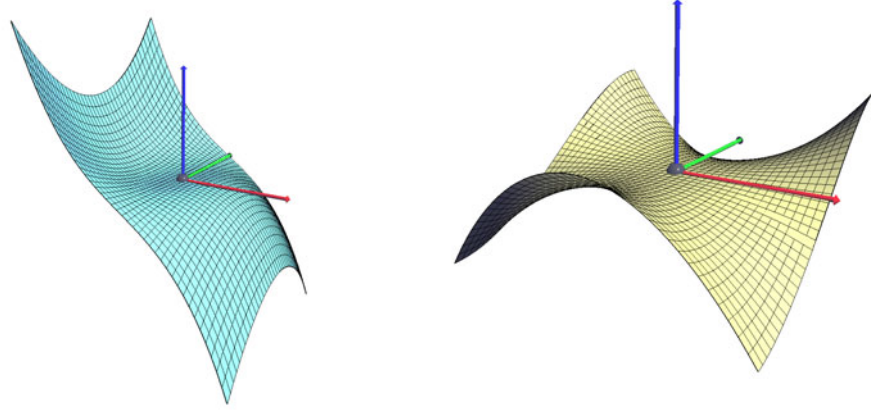


Figure 3.5. The first and third Fourier components of the third-order shape function — all third-order surface behavior can be expressed as properly scaled and rotated combinations of these two shapes.

direction to measure the phase shifts. Fig. 3.3 illustrates this linear combination for a fixed value of  $r$ . Fig. 3.4 illustrates the combination of these Fourier components to form the cubic surface.

We can consider the two phase shifts  $\alpha$  and  $\delta$  independently of each other. However, we find it more instructive to consider the direction corresponding to the (single) maximum of  $F_1 \cos(\theta + \alpha)$  as a “third-order principal direction.” Then, the phase shift  $\delta$  can be expressed as  $\alpha + \beta$ , where  $\beta$  is the phase shift with respect to the third order principal direction. Therefore, we get our final equation for describing the cubic behavior of the

surface,

$$z_{p_c}(r, \theta) = r^3[F_1 \cos(\theta + \alpha) + F_3 \cos 3(\theta + \alpha + \beta)]. \quad (3.13)$$

We use the amplitudes and phase shifts of the Fourier components from Eqn. 3.13 as our four parameterization-independent, geometrically intuitive shape parameters (illustrated in Fig. 3.2). These parameters can be extracted from the original third-order parameters  $C_0$ ,  $C_1$ ,  $C_2$ ,  $C_3$  (Eqn. 3.3) of the polynomial height field:

$$F_1 = \frac{\sqrt{(3C_0 + C_3)^2 + (3C_1 + C_2)^2}}{4}, \quad (3.14)$$

$$F_3 = \frac{\sqrt{(C_0 - C_3)^2 + (C_2 - C_1)^2}}{4}, \quad (3.15)$$

$$\alpha = \tan^{-1} \left( \frac{3C_1 + C_2}{3C_0 + C_3} \right), \quad (3.16)$$

$$\beta = \frac{1}{3} \tan^{-1} \left( \frac{C_2 - C_1}{C_0 - C_3} \right) - \alpha. \quad (3.17)$$

Similarly, given our third-order parameters  $F_1$ ,  $F_3$ ,  $\alpha$  and  $\beta$ , we can extract the parameterization-dependent third-order parameters for the idealized surface patch:

$$C_0 = F_1 \cos \alpha + F_3 \cos \beta, \quad (3.18)$$

$$C_1 = F_1 \sin \alpha - F_3 \sin \beta, \quad (3.19)$$

$$C_2 = F_1 \sin \alpha + 3F_3 \sin \beta, \quad (3.20)$$

$$C_3 = F_1 \cos \alpha - 3F_3 \cos \beta. \quad (3.21)$$

### 3.6 Computing Fourier Components for a General Surface Patch

In this section we describe how to compute the third-order shape parameters for a point on a general surface patch. Unlike the approach taken in Section 3.3, we can no longer



ignore the effect of lower-order shape parameters (namely, first- and second-order parameters) on the third-order shape parameters. Therefore, we cannot extract parameterization independent shape parameters simply by analyzing a height function. Instead, we need to perform a Fourier analysis of the function that denotes the arc-length derivative of normal curvature. The Fourier coefficients can then be combined as above to yield the required shape parameters.

Consider that we have a bi-variate tensor product surface patch (e.g. a bi-cubic B-spline patch) parameterized by  $u, v$ . Given a point  $(u, v)$  in parameter space, let  $\mathbf{S}(u, v)$  denote the 3D position of the point,  $\mathbf{n}$  denote the unit normal, and  $\mathbf{S}_{\mathbf{u}}(u, v)$ ,  $\mathbf{S}_{\mathbf{v}}(u, v)$ ,  $\mathbf{S}_{\mathbf{uu}}(u, v)$ , etc. denote the 3D parametric surface derivatives with respect to  $u$  and  $v$ . Our task is to efficiently and exactly compute the  $F_1$ ,  $F_3$ ,  $\alpha$  and  $\beta$  parameters for any point  $(u, v)$  on the patch.

First, compute the parameterization-dependent third order shape parameters  $P$ ,  $Q$ ,  $S$ , and  $T$  introduced by Mehlum and Tarrou [MT98]:

$$P = \mathbf{S}_{\mathbf{uuu}} \cdot \mathbf{n} + 3\mathbf{S}_{\mathbf{uu}} \cdot \mathbf{n}_{\mathbf{u}}, \quad (3.22)$$

$$Q = \mathbf{S}_{\mathbf{uuv}} \cdot \mathbf{n} + 2\mathbf{S}_{\mathbf{uv}} \cdot \mathbf{n}_{\mathbf{u}} + \mathbf{S}_{\mathbf{uu}} \cdot \mathbf{n}_{\mathbf{v}}, \quad (3.23)$$

$$S = \mathbf{S}_{\mathbf{uvv}} \cdot \mathbf{n} + 2\mathbf{S}_{\mathbf{uv}} \cdot \mathbf{n}_{\mathbf{v}} + \mathbf{S}_{\mathbf{vv}} \cdot \mathbf{n}_{\mathbf{u}}, \quad (3.24)$$

$$T = \mathbf{S}_{\mathbf{vvv}} \cdot \mathbf{n} + 3\mathbf{S}_{\mathbf{vv}} \cdot \mathbf{n}_{\mathbf{v}}. \quad (3.25)$$

Then, use the formula from [MT98] that expresses the arc-length derivative of normal curvature as a function of the angle  $\theta$  from any given reference direction

$$\begin{aligned} \kappa'_n(\theta) = \frac{1}{\sigma^3} [ & PG^{3/2} \sin^3 \theta + 3QGE^{1/2} \sin^2 \theta \cos(\theta + \psi) \\ & + 3SEG^{1/2} \sin \theta \cos^2(\theta + \psi) + TE^{3/2} \cos^3(\theta + \psi) ], \end{aligned} \quad (3.26)$$

where  $\theta$  is measured from the  $u$  direction,  $E$ ,  $F$  and  $G$  are coefficients of the first fundamental form (the metric tensor), and  $\sigma = \sqrt{F^2 - EG}$  is the area element at the point of analysis. We maintain the label  $\kappa'_n(\theta)$  for the arc-length derivative of normal curvature  $\kappa_n(\theta)$  as was done by Mehlum and Tarrou [MT98].  $\psi$  denotes the complement to the angle between the  $u$  and  $v$  directions and is given by  $\tan(\psi) = F/\sqrt{EG}$ . (For the polynomial height field of Section 3.3, the coordinate axes were mutually orthogonal and therefore  $\psi$  was zero.)

Eqn. 3.26 can be written as an expression similar to Eqn. 3.3:

$$\kappa'_n(\theta) = A \cos^3(\theta + \psi) + B \sin^3 \theta + C \sin \theta \cos^2(\theta + \psi) + D \sin^2 \theta \cos(\theta + \psi) \quad (3.27)$$

where the coefficients  $A$ ,  $B$ ,  $C$ , and  $D$  can easily be written as functions of  $P$ ,  $Q$ ,  $S$ ,  $T$ , and  $E$ ,  $F$ , and  $G$ :

$$A = \frac{TE^{3/2}}{\sigma^3} \quad , \quad B = \frac{PG^{3/2}}{\sigma^3}, \quad (3.28)$$

$$C = \frac{3SEG^{1/2}}{\sigma^3} \quad , \quad D = \frac{3QGE^{1/2}}{\sigma^3}. \quad (3.29)$$

As described in Section 3.5, we can perform a Fourier analysis of the sinusoidal functions in Eqn. 3.27:

$$\begin{aligned} \cos^3(\theta + \psi) &= 0.75 \cos \psi \cos \theta - 0.75 \sin \psi \sin \theta \\ &+ 0.25 \cos 3\psi \cos 3\theta - 0.25 \sin 3\psi \sin 3\theta, \\ \sin^3 \theta &= 0.75 \sin \theta - 0.25 \sin 3\theta, \\ \cos^2(\theta + \psi) \sin \theta &= -0.25 \sin 2\psi \cos \theta - 0.25(\cos 2\psi - 2) \sin \theta \\ &+ 0.25 \sin 2\psi \cos 3\theta + 0.25 \cos 2\psi \sin 3\theta, \\ \cos(\theta + \psi) \sin^2 \theta &= 0.25 \cos \psi \cos \theta - 0.75 \sin \psi \sin \theta \\ &- 0.25 \cos \psi \cos 3\theta + 0.25 \sin \psi \sin 3\theta. \end{aligned}$$

By grouping coefficients, we express the arc-length derivative of normal curvature as a sum of first-order and third-order sinusoidal functions

$$\kappa'_n(\theta) = F_{1\cos} \cos \theta + F_{1\sin} \sin \theta + F_{3\cos} \cos 3\theta + F_{3\sin} \sin 3\theta, \quad (3.30)$$

where

$$F_{1\cos} = 0.25(3A \cos \psi - C \sin 2\psi + D \cos \psi), \quad (3.31)$$

$$F_{1\sin} = 0.25(-3A \sin \psi + 3B - C(\cos 2\psi - 2) - 3D \sin \psi), \quad (3.32)$$

$$F_{3\cos} = 0.25(A \cos 3\psi + C \sin 2\psi - D \cos \psi), \quad (3.33)$$

$$F_{3\sin} = 0.25(-A \sin 3\psi - B + C \cos 2\psi + D \sin \psi). \quad (3.34)$$

Finally, we can combine the sine and cosine functions to formulate the arc-length derivative of normal curvature as a sum of phase-shifted sinusoidal functions of the angle  $\theta$

$$\kappa'_n(\theta) = F_1 \cos(\theta + \alpha) + F_3 \cos(3(\theta + \alpha + \beta)), \quad (3.35)$$

where the parameterization independent third order shape parameters can be expressed in closed-form as:

$$F_1 = \frac{\sqrt{F_{1\cos}^2 + F_{1\sin}^2}}{4}, \quad F_3 = \frac{\sqrt{F_{3\cos}^2 + F_{3\sin}^2}}{4}, \quad (3.36)$$

$$\alpha = \tan^{-1} \left( \frac{-F_{1\sin}}{F_{1\cos}} \right), \quad \beta = \frac{1}{3} \tan^{-1} \left( \frac{-F_{3\sin}}{F_{3\cos}} \right) - \alpha. \quad (3.37)$$

To summarize, to compute the third-order shape parameters for any point  $u, v$  on a general surface patch, we need to compute the parameterization dependent third-order  $(P, Q, S, T)$  and first-order  $(E, F, G)$  parameters. Algebraic manipulation of these parameters yields the coefficients  $F_{1\cos}$ ,  $F_{1\sin}$ ,  $F_{3\cos}$ , and  $F_{3\sin}$  of the four sinusoidal components of arc-length derivative of normal curvature. These four coefficients then readily yield the  $F_1$ ,  $F_3$ ,  $\alpha$  and  $\beta$  parameters.

### 3.7 Qualitative Description of the Fourier Components

The shapes of the first and third Fourier components are shown in Fig. 3.5. Both functions are anti-symmetric with respect to  $\pi$ , which leads to their combination being anti-symmetric as well,  $z_{p_c}(r, \theta) = -z_{p_c}(r, \pi + \theta)$  — a fact pointed out by Berry and Hannay [BH77] in their study of umbilics and Mehlum and Tarrou [MT98] in their study of normal curvature variation.

In the range  $[0, 2\pi)$ , the first Fourier component has one maximum and minimum. The shape of this component is given by the height field  $z = x^3 + xy^2$  and can be understood as a lateral extrusion of the cubic curve  $z = x^3$  in the  $y$  direction, enhanced by a linear component whose slope increases as the square of  $y$  (see Fig. 3.5). When  $F_1$  is zero, the first Fourier component is flat and the angle  $\alpha$  cannot be uniquely determined (in this case we set  $\alpha$  to zero in our implementation). We consider such a point a third-order equivalent of the umbilic. Unlike the umbilic where the normal curvature is equal in all directions, at the third order equivalent of the umbilic the normal curvature derivative does not necessarily behave the same — it is influenced by the non-zero third Fourier component. In fact, as shown by [MT98], the *only* situation when the normal curvature derivative is equal in all directions is when it is zero, meaning the surface is flat in third order (i.e. both  $F_1$  and  $F_3$  are zero).

In the range  $[0, 2\pi)$ , the third Fourier component has three equally spaced maxima and minima. The shape of this component is similar to that of the height field  $z = x^3 - 3xy^2$ . This is the well-known “monkey saddle” with three peaks and troughs, each  $\pi/3$  radians apart. The angle  $\beta$  denotes the rotation of the third Fourier component with respect to the  $\alpha$  direction given by the first Fourier component. As shown in Fig. 3.6, given a fixed  $\alpha$

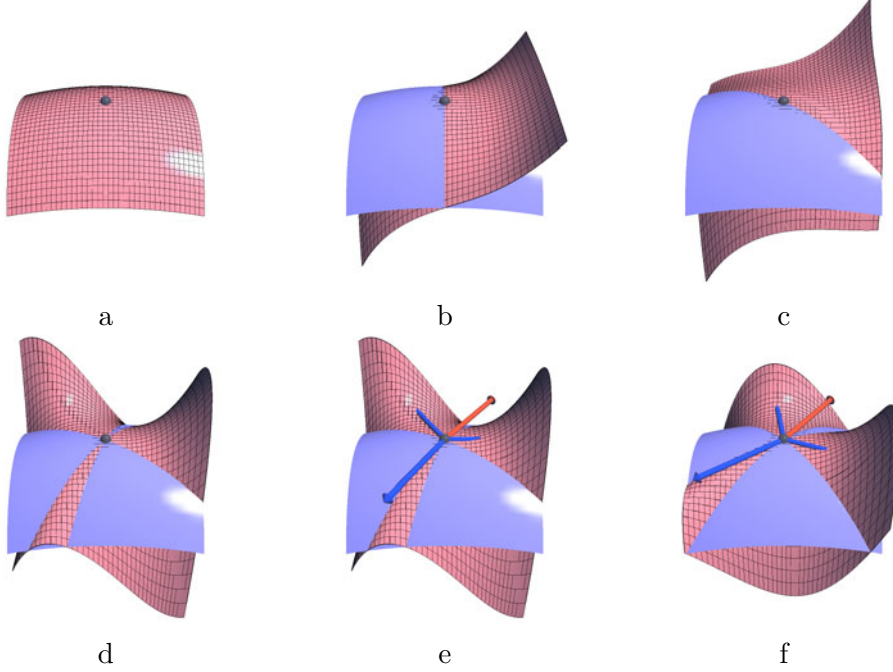


Figure 3.6. **Sequence of third-order shape edits:** starting from a purely second order surface patch where  $F_1$  and  $F_3$  are zero (a), we increase the amplitude  $F_1$  of the first Fourier component (b), rotate it about the  $z$ -axis by increasing the value of  $\alpha$  (c), and increase the amplitude  $F_3$  of the third Fourier component (d). (e) shows the same shape as (d) but with the third-order frame indicating the directions of  $\alpha$  and  $\beta$ . Finally, we rotate only the third Fourier component about the  $z$ -axis by increasing the value of  $\beta$  (f). The blue surface (without the grid) is the best-fitting (and unchanged) quadratic surface at the point of analysis.

and  $F_1$ , we can vary  $\beta$  and  $F_3$  to change the undulatory behavior of the third order height function. When  $F_3$  is zero,  $\beta$  cannot be uniquely determined. In this case, we set it to zero in our implementation.

### 3.7.1 Expressing Cross Derivatives Using Third-Order Shape Parameters

Equation 3.35 gives an expression for the *inline* derivative of curvature ( $\kappa'_n$ ) — the change of curvature is analyzed along the line for which normal curvature is measured. Alternately, we can consider *cross* derivatives of curvature ( $\kappa_n^\times$ ), where the change of curvature is analyzed in a direction perpendicular to the line along which the normal curvature is measured. For example, in an earlier Master's thesis [Jos07], we introduced the MVS<sub>cross</sub>

functional that contains cross derivative terms in principal directions:  $d\kappa_1/de_2$  and  $d\kappa_2/de_1$ .

Here we use our third-order parameters  $F_1$ ,  $F_3$ ,  $\alpha$ , and  $\beta$  to obtain an expression for the cross derivative of normal curvature.

Suppose we are given a surface point with normal curvature  $\kappa_n(\theta)$  in a direction given by angle  $\theta$  in the tangent plane. The cross derivative  $\kappa_n^\times(\theta)$  is a directional derivative of  $\kappa_n(\theta)$  along the direction denoted by  $\theta + \pi/2$ . We can show that the cross derivative is given by the formula

$$\begin{aligned}\kappa_n^\times(\theta) &= \frac{F_1}{3} \cos(\theta + \pi/2 + \alpha) - F_3 \cos 3(\theta + \pi/2 + \alpha + \beta) \\ &= -\frac{F_1}{3} \sin(\theta + \alpha) - F_3 \sin 3(\theta + \alpha + \beta)\end{aligned}\tag{3.38}$$

The above equation is similar to Equation 3.35 which expresses the normal curvature derivative ( $\kappa'_n$ ) using the third-order shape parameters. There are three differences: (1) the  $F_1$  component is reduced to a third of its original value, (2) the  $F_3$  component switches sign and (3) the angles are shifted by  $\pi/2$  radians.

The derivation for Equation 3.38 proceeds as follows: we can express the third-order surface information near the point of analysis by the cubic height field function used in Section 3.3

$$z_{pc}(x, y) = C_0x^3 + C_1y^3 + C_2x^2y + C_3xy^2\tag{3.39}$$

We can use this description in a small neighborhood around a surface point where the first fundamental form is the identity matrix and the second fundamental form is zero.

Suppose we are interested in the cross derivative  $\kappa_{n_y}^\times = \frac{d\kappa_n(\pi/2)}{dx} = \frac{d}{dx} \frac{d^2z_{pc}}{dy^2} = \frac{d^3z_{pc}}{dy^2dx} = 2C_3$ . We will show how this cross derivative is closely related to the inline curvature derivative  $\kappa'_{n_x} = \frac{d\kappa_n(0)}{dx} = \frac{d^3z_{pc}}{dx^3} = 6C_0$ .

Consider the situation when  $F_1$  is non-zero and  $F_3$  is zero. Without loss of generality, we can define the  $x$ - $y$  coordinate system around such a surface point such that  $C_0 = C_3 \neq 0$  and  $C_1 = C_2 = 0$  (the  $x$ -axis is along the maximal direction of the  $F_1$  component). In this case,  $\frac{d^3 z_{pc}}{dx dy^2} = \frac{1}{3} \frac{d^3 z_{pc}}{dx^3}$ , which implies that the value of the cross derivative of normal curvature is equal to one third the value of the inline derivative of normal curvature, where both curvature derivatives are in the same direction. For a general direction denoted by  $\theta$ , the cross derivative in the direction  $\phi = \theta + \pi/2$  of the normal curvature  $\kappa_n(\theta)$  is

$$\begin{aligned}
\frac{d\kappa_n(\theta)}{d\mathbf{e}_\phi} &= \frac{1}{3} \frac{d\kappa_n(\phi)}{d\mathbf{e}_\phi} \\
&= \frac{1}{3} F_1 \cos(\phi + \alpha) \\
&= \frac{1}{3} F_1 \cos(\theta + \pi/2 + \alpha) \\
&= -\frac{1}{3} F_1 \sin(\phi + \alpha).
\end{aligned} \tag{3.40}$$

Now consider the situation when  $F_1$  is zero and  $F_3$  is non-zero. Without loss of generality, we can define the  $x$ - $y$  coordinate system around such a surface point such that  $C_0 = -3C_3 \neq 0$  and  $C_1 = C_2 = 0$  (the  $x$ -axis is along one of the maximal directions of the  $F_3$  component). In this case,  $\frac{d^3 z_{pc}}{dx dy^2} = -\frac{d^3 z_{pc}}{dx^3}$  which implies that the value of the cross derivative of normal curvature is equal to the negative value of the inline derivative of normal curvature, where both curvature derivatives are in the same direction. For a general direction denoted by  $\theta$ , the cross derivative in the direction  $\phi = \theta + \pi/2$  of the normal curvature  $\kappa_n(\theta)$  is

$$\begin{aligned}
\frac{d\kappa_n(\theta)}{d\mathbf{e}_\phi} &= -\frac{d\kappa_n(\phi)}{d\mathbf{e}_\phi} \\
&= -F_3 \cos 3(\phi + \alpha + \beta) \\
&= -F_3 \cos 3(\theta + \pi/2 + \alpha + \beta) \\
&= -F_3 \sin 3(\theta + \alpha + \beta).
\end{aligned} \tag{3.41}$$

Just like the inline curvature derivative function  $\kappa'_n$ , we can express the cross curvature derivative function  $\kappa_n^\times$  as a sum of its first and third Fourier components. By combining Equations 3.40 and 3.41, we get the expression for Equation 3.38.

### 3.7.2 Expressing Normal Curvature Derivatives in Arbitrary Directions Using Third-Order Shape Parameters

The inline and cross derivatives are only two of the infinitely many directions in which we can compute directional derivatives of normal curvature. Given a surface point and a normal curvature  $\kappa_n(\theta)$  measured along a direction given by  $\theta$ , we should be able to compute the directional derivative  $d\kappa_n(\theta)/d\mathbf{e}_\psi$  for an arbitrary direction  $\mathbf{e}_\psi$ . At any surface point, up to third order, we can define a rank-3 tensor that takes 3 directions as input: two (equal) directions to query the curvature tensor and specify the normal curvature and a third direction to specify the direction of normal curvature derivative [Rus04, GU01]. We now show that the normal curvature derivatives in all directions are simple linear combinations of inline and cross curvature derivatives.

Recall the rule of directional derivatives. Suppose  $f$  is a scalar function over a domain spanned by directions  $\hat{\mathbf{x}}$  and  $\hat{\mathbf{y}}$ . Let the direction  $\mathbf{m}$  also be spanned by the  $x$ - $y$  basis ( $\mathbf{m} = m_x\hat{\mathbf{x}} + m_y\hat{\mathbf{y}}$ ). Then, the directional derivative  $\frac{\partial f}{\partial \mathbf{m}} = \mathbf{m} \cdot (\frac{\partial f}{\partial x}\hat{\mathbf{x}} + \frac{\partial f}{\partial y}\hat{\mathbf{y}})$ .

Let the direction of the inline derivative be along the  $x$  axis, and the direction corresponding to the cross derivative be along the  $y$  axis. A vector along an arbitrary direction given by angle  $\psi$  can be written as  $(\cos \psi)\hat{\mathbf{x}} + (\sin \psi)\hat{\mathbf{y}}$ . Therefore, using the above rule of directional derivatives and given the inline and cross derivatives of normal curvature,  $\kappa'_n(\theta)$



and  $\kappa_n^\times(\theta)$ , we can express the directional derivative of  $\kappa_n(\theta)$  along the direction of  $\psi$  as:

$$\begin{aligned}\frac{d\kappa_n(\theta)}{d\mathbf{e}_\psi} &= ((\cos \psi)\hat{\mathbf{x}} + (\sin \psi)\hat{\mathbf{y}}) \cdot (\kappa_n(\theta)'\hat{\mathbf{x}} + \kappa_n(\theta)^\times\hat{\mathbf{y}}) \\ &= \kappa_n'(\theta) \cos \psi + \kappa_n^\times(\theta) \sin \psi\end{aligned}\tag{3.42}$$

where  $\psi$  is computed as the offset angle from the direction of  $\theta$ .

### 3.7.3 Application: Classification of Umbilics

As one example, we show how to use our third-order shape parameters to characterize the surface behavior near umbilic points (points with equal principal curvatures). As mentioned before, generic umbilic points on surfaces are classified according to the pattern made by lines of curvature as they pass through the point. Since the surface behavior up to second order is uniform in all directions, we need a third-order analysis to classify umbilics. As presented by Berry and Hannay [BH77], based on the pattern of lines of principal curvature near the point, there are three types of generic (stable) surface umbilics: *lemon*, *monstar* and *star* (see Figure 3.7). The pattern of lines of principal curvature depends on the number of real, distinct roots of the cubic equations  $z_{p_c}(r, \theta) = 0$  ( $z_{p_c}$  from Section 3.3) and  $\frac{dz_{p_c}(r, \theta)}{d\theta} = 0$ . The roots can be obtained by computing the discriminants of the two cubic equations (the third order-height function and  $\frac{dz_{p_c}(r, \theta)}{d\theta} = 0$ ). Computing the roots is useful if one needs to find the exact location of the lines of principal curvature, but the discriminants and roots by themselves do not provide a quick geometric understanding of how the surface behaves near the umbilic point. Instead, our third-order shape parameters offer a more intuitive explanation of when and how different types of umbilics are formed. When the first Fourier component dominates the overall third-order behavior, we get only one maximum and minimum for  $z_{p_c}(r, \theta)$ . In that case, we have the *lemon* type of umbilic.

When the third Fourier component is strong enough that its derivatives (slope) exceed those of the first component, we get three distinct maxima and minima (six real roots for the equation  $\frac{dz_{pc}(r,\theta)}{d\theta} = 0$ ) and obtain the *monstar* umbilic. If the third Fourier component dominates the third-order height function and creates six zero crossings (instead of two), we get the *star* type of umbilic. Figure 3.7 compares the network of curvature lines to the number of zeros and extrema of the third-order height function evaluated along a small circle around the umbilic point.

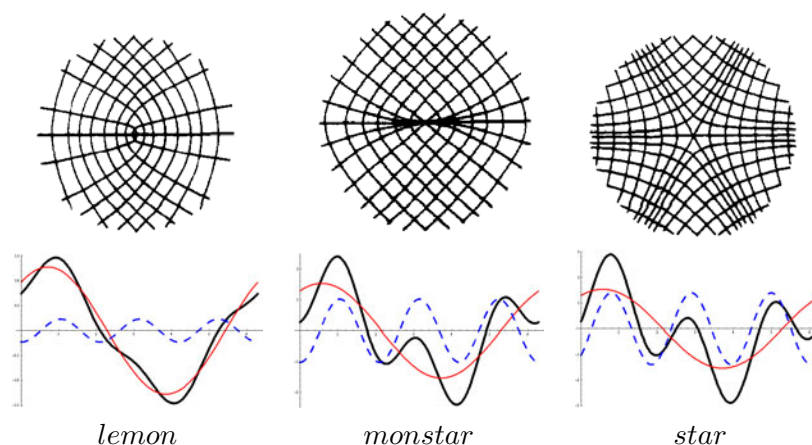


Figure 3.7. Principal curvature lines near the three types of umbilic points with the graphs of corresponding third-order height functions (thick black). The top row of figures is from Berry and Hannay [BH77]. The number of extrema (two or six) and zero crossings (two or six) of the height function together determine the type of the generic umbilic point. Notice how the first Fourier component (solid red) dominates the overall third-order behavior for the *lemon* umbilic, while the third Fourier component (dotted blue) creates local extrema in the *monstar* umbilic and additional zero crossings of the height function in the *star* umbilic.

## 3.8 Summary

We have presented an intuitive analysis of third-order surface behavior in terms of Fourier components of the third-order height function. We hope our exposition will be useful as a tool for studying and characterizing third order geometry. In the next chapter,

we will use our understanding of third-order surface behavior to define functionals built from the fundamental building blocks  $F_1$  and  $F_3$ .

## Chapter 4

# Functionals

### Introduction

Our goal is to use numerical optimization to construct smooth surfaces suitable for aesthetic shape design. We optimize the shape of an input surface so that it minimizes an energy defined as a function over the surface. The energy is also called a “functional”. We focus on functionals that, when minimized, lead to smooth, aesthetically pleasing shapes. Informally speaking, the functional returns a numerical value of the beauty of the surface; the lower the value, the more beautiful the surface. In this chapter we describe some of the commonly used functionals and also introduce some new ones.

### 4.1 Requirements of Surface Energy Functionals

Before listing the formulae for the functionals, it is instructive to consider the requirements of all functionals useful for aesthetic design. While the exact formulation of the ideal functional depends on the particular design task, we can postulate that for the types of shapes we wish to design, all functionals should:

- be bounded from below: all energy functionals should have a finite lower bound so that the optimization can converge in a finite number of steps,
- penalize sharp shape changes: unless specified otherwise, the optimization should remove sharp features like kinks and creases and not introduce new ones,
- penalize unnecessary surface bending: the optimization should remove any unnecessary wiggles, undulations or other smooth extraneous shape changes and not introduce new ones,
- be local operators: the energy near a surface point should be efficiently computable by considering only the local neighborhood of the point,
- be invariant to parameterization, rigid transformation and uniform scaling: different designers using different surface parameterizations at different scales should obtain the same optimized shape, and
- be numerically stable: the functional should be “smooth” (have continuous first and second derivatives with respect to the degrees of freedom) so that we can use commonly found optimization algorithms.

All the functionals selected for aesthetic design in this thesis satisfy the above requirements. While evaluating a functional for aesthetic design, we found that along with the above requirements, we would like the ideal functional to:

- reward symmetry: the functional should reward symmetry by yielding lower energy for more symmetric shapes, and
- be numerically well-conditioned: to compute the energy minimum more conveniently and quickly.

## 4.2 How to Construct Functionals

We use Birkhoff’s principle [Bir33], which states that the mathematical measure of a shape’s beauty is inversely proportional to its complexity. One can argue that the complexity of a surface is inversely proportional to its smoothness. That is, a smooth surface is less complicated, while a rough, bumpy surface is more complex. To make the least complex shape, we consider energy functionals that reward smooth shapes and penalize unnecessary variation in the normals or curvatures in the shape. The variation is computed by measuring the arc-length derivatives of the position of a surface point. Each order of the derivative corresponds to a particular class of functionals. In this thesis, we will consider functionals up to order three. We compute the derivative terms at surface points and formulate the functionals as area integrals over the entire surface ( $\int \bullet dA$ ) of the  $L^2$  norm of the derivative terms. Only a subset of all the functionals that can be formulated by combining the derivative terms are useful from the perspective of aesthetic design. Therefore, we need to study only a subset of all the functionals that we list in this chapter.

### 4.2.1 First-Order Functional

Up to first order, we can formulate the energy functional that measures surface area as

$$\text{surface area} = \int dA. \tag{4.1}$$

This functional is used to solve the design task of constructing the simplest surface that interpolates a closed, non-planar boundary curve (i.e. Plateau’s problem, see [Rad30]). The optimal surface corresponding to the surface area functional is called a *minimal surface*. Constructing a surface of least area is inspired by nature: soap films naturally form minimal surfaces, and Plateau’s formulation of the problem was inspired by his study of soap films.

The surface area functional has some drawbacks when used for aesthetic design. First, the functional is not scale invariant (uniformly doubling the scale will quadruple surface area). Second, the functional cannot be applied to unconstrained surfaces since the optimization will simply collapse the surface to a single point. Even in the case of surfaces with constraints, any largely unconstrained regions may collapse to a single point or line.

Despite its drawbacks and due to its simplicity, the surface area functional has been used for aesthetic design — the boundary and rough approximation of the desired surface is given, and the final surface is computed by minimizing the surface area. There is a rich history behind the use of this functional for constructing minimal surfaces; we will not repeat that information in this thesis. For more information on minimal surfaces, refer to Osserman’s book on the topic [Oss02].

#### 4.2.2 Second-Order Functionals

Second-order functionals are functions of the normal curvature at a surface point. Recall that the normal curvature  $\kappa_n$  is the arc-length derivative of the surface normal, where the derivative is computed along a normal section curve. Depending on the direction in which the arc-length is varied, we get a different value for the normal curvature. Euler’s Theorem states that the normal curvature in a given direction at a surface point is a function of the maximum ( $\kappa_1$ ) and minimum ( $\kappa_2$ ) normal curvatures at that point,

$$\kappa_n(\theta) = \kappa_1 \cos^2 \theta + \kappa_2 \sin^2 \theta, \quad (4.2)$$

where  $\kappa_1$  and  $\kappa_2$  are called the *principal* curvatures and  $\theta$  is the angle between the given direction and the principal direction corresponding to  $\kappa_1$ . Using  $\kappa_1$  and  $\kappa_2$ , we can define

two common curvature terms,

$$\text{mean curvature (H)} = \frac{1}{\pi} \int_0^\pi \kappa_n(\theta) d\theta = \frac{\kappa_1 + \kappa_2}{2}, \text{ and} \quad (4.3)$$

$$\text{Gaussian curvature (K)} = \kappa_1 \kappa_2. \quad (4.4)$$

We are now ready to define some second-order functionals:

### Mean Curvature Energy

The mean curvature energy integrated over the surface is

$$\text{mean curvature energy} = \int H \, dA. \quad (4.5)$$

For aesthetic design, the mean curvature energy functional is similar to the surface area functional, with the same benefits and drawbacks. The only difference between the surface area and the mean curvature energy is in the type of constraints we can specify. While minimizing surface area, we can specify only position ( $C^0$ ) constraints, but while minimizing mean curvature energy, we can specify both, position and tangent ( $C^1$ ) constraints. With the absence of tangent constraints, the minimal surface obtained as a result of area minimization will have *zero mean curvature* at all interior points. Therefore, without any tangent constraints, the mean curvature energy functional and the surface area functional have the same minimizers. This is not surprising as the gradient of the mean curvature energy (the direction of maximal mean curvature change) is parallel to the gradient of surface area. In terms of the nature of preferred shapes for aesthetic design, study of the mean curvature energy adds no new information than that already included in the study of surface area functional. Therefore, in this thesis, we will not discuss the use of the mean curvature energy for aesthetic design.



## Gaussian Curvature Energy

The Gaussian curvature energy integrated over the surface is

$$\text{Gaussian curvature energy} = \int K dA = \int \kappa_1 \kappa_2 dA. \quad (4.6)$$

According to the Gauss–Bonnet theorem, this integral is a topological constant of the surface and its value depends only on the topological type (i.e. genus) of the surface, not its shape. Since we are not changing the surface topology during optimization, the genus of the input surface remains constant, and therefore the Gaussian curvature energy remains constant. Thus, we ignore this energy for aesthetic design.

## Willmore Energy (Square of Mean Curvature)

We can compute the area integral of the square of mean curvature, also known as Willmore energy [Wil71] as

$$\text{Willmore energy} = \int H^2 dA. \quad (4.7)$$

An alternative formulation of the Willmore energy measures the deviation of a surface point from an umbilic point (a point with equal normal curvature in all directions). That is, we can measure the derivative of the normal curvature with respect to the direction given by angle  $\theta$ ,

$$\begin{aligned} \text{umbilic deviation energy} &= \int \left( \frac{d\kappa_n(\theta)}{d\theta} \right)^2 dA, \\ &= \int (\kappa_1 - \kappa_2)^2 dA \text{ (by Eqn. 4.2) }, \\ &= 4 \int \left( \frac{\kappa_1 + \kappa_2}{2} \right)^2 dA - 4 \int \kappa_1 \kappa_2 dA, \\ &= 4(\text{Willmore energy}) - 4(\text{Gaussian curv. energy}). \end{aligned} \quad (4.8)$$

Since the Gaussian curvature energy is a topological constant, we need to minimize only the Willmore energy to minimize umbilic deviation.

The Willmore energy is well studied and can be used for aesthetic design. However, as we show next, minimizing the Willmore energy is equivalent to minimizing a more complete and common functional, the bending energy. Therefore, we consider the bending energy instead of the Willmore energy for aesthetic design.

### Bending Energy

Consider a functional that measures the average value of the square of the normal curvature by integrating the square of normal curvature over all directions. That is, we wish to find

$$\begin{aligned}
 \text{total curvature energy} &= \int \frac{1}{\pi} \left( \int_0^\pi \kappa_n(\theta)^2 d\theta \right) dA, \\
 &= \frac{3}{8} \int \kappa_1^2 + \kappa_2^2 dA + \frac{2}{8} \int \kappa_1 \kappa_2 dA \text{ see [MT98]}, \\
 &= \frac{3}{8} \int \kappa_1^2 + \kappa_2^2 dA + \frac{2}{8} (\text{Gaussian curv. energy}). \quad (4.9)
 \end{aligned}$$

As before, we can ignore the constant Gaussian curvature, giving

$$\text{bending energy} = \int \kappa_1^2 + \kappa_2^2 dA. \quad (4.10)$$

This energy is the most popular functional in surface optimization. Example applications are constructing  $G^1$ -continuous surfaces to fill holes, blends between two or more surfaces, fairing a noisy surface obtained from range scans and simulating the interface between lipids and microscopic cell membranes.

As mentioned earlier, minimizing the bending energy is equivalent to minimizing the

Willmore energy ( $\int H^2 dA$ ).

$$\begin{aligned}
\text{Willmore energy} &= \int H^2 dA = \int \left( \frac{\kappa_1 + \kappa_2}{2} \right)^2 dA, \\
&= \frac{1}{4} \int (\kappa_1^2 + \kappa_2^2) dA + \frac{1}{2} \int \kappa_1 \kappa_2 dA, \\
&= \frac{1}{4} (\text{bending energy}) + \frac{1}{2} (\text{Gaussian curv. energy}). \quad (4.11)
\end{aligned}$$

where, as before, we can ignore the Gaussian curvature energy and focus on minimizing only the bending energy.

To summarize, while we can formulate a large number of second-order surface energies, we need to consider only the bending energy (Equation 4.10) as a second-order functional for purposes of aesthetic design.

### 4.2.3 Third-Order Functionals

Up to third order, we can formulate functionals as functions of the arc-length derivative of normal curvature (i.e.  $\kappa'_n$ ). Unlike second-order expressions (involving  $\kappa_n$ ), which could be simplified using the Gauss–Bonnet theorem and Euler’s theorem, we no longer have any convenient theorems to simplify expressions involving  $\kappa'_n$ . Instead, we use the third-order shape parameters  $F_1$ ,  $F_3$ ,  $\alpha$ , and  $\beta$  (from Chapter 3) to simplify expressions for third-order functionals.

### Minimum Variation Surface (MVS and MVS<sub>cross</sub>) Energies

The MVS energy was introduced by Moreton and Séquin [MS92] as an alternative to bending energy. The MVS functional measures the derivative of the principal curvatures in their principal directions. In this thesis, we consider the scale-invariant (“SI-”) version of

the MVS energy. That is,

$$\text{SI-MVS energy} = \int \left( \frac{d\kappa_1}{de_1} \right)^2 + \left( \frac{d\kappa_2}{de_2} \right)^2 dA \cdot \int dA \quad (4.12)$$

where the additional  $\int dA$  makes the MVS energy scale invariant [SCM95]. See Section 4.4 for more details on scale invariance.

The SI-MVS functional favors surfaces that have constant change of curvature along principal curvature lines. These surfaces, called cyclides, have circles or straight lines as lines of curvature (e.g. spheres, cylinders, cones, tori, Horn tori). The SI-MVS energy is zero for *all* cyclides: it treats all cyclides as equally beautiful and cannot distinguish between them. To fix this drawback, we [Jos07] enhanced the SI-MVS functional by adding cross derivative terms to get

$$\text{SI-MVS}_{\text{cross}} \text{ energy} = \int \left( \frac{d\kappa_1}{de_1} \right)^2 + \left( \frac{d\kappa_2}{de_2} \right)^2 + \left( \frac{d\kappa_1}{de_2} \right)^2 + \left( \frac{d\kappa_2}{de_1} \right)^2 dA \cdot \int dA. \quad (4.13)$$

Unlike the SI-MVS functional, the SI-MVS<sub>cross</sub> functional is zero for only two cyclides: spheres and cylinders. The SI-MVS<sub>cross</sub> functional distinguishes between different tori — the infinitely thin torus has the least energy [JS07].

**Drawbacks:** Both the SI-MVS and the SI-MVS<sub>cross</sub> functionals assign a special significance to the principal curvatures and directions, measuring the curvature changes only in principal directions. However, the principal directions (a second-order property) are not relevant in third-order space. Choosing to measure curvature variation in only the principal directions could ignore significant curvature variation in other directions. As pointed out by Mehlum and Tarrou [MT98], the maximum value of curvature variation occurs in directions independent of the principal directions. Moreover, at umbilic points where the normal curvature is equal in all directions, we cannot uniquely compute the principal curvature

directions and have to pick two arbitrary, mutually perpendicular directions as principal directions. Therefore, we think a more complete and stable functional is one that considers normal curvature variation in all directions, as proposed by Mehlum and Tarrou.

### Mehlum and Tarrou's Energy

Mehlum and Tarrou [MT98] argue that to measure the total curvature variation at a surface point, we should compute the average magnitude of the arc-length derivative of normal curvature across all directions.

$$\text{Mehlum-Tarrou energy} = \frac{1}{\pi} \int \left( \int_0^\pi \kappa'_n(\theta)^2 d\theta \right) dA \quad (4.14)$$

where  $\kappa'_n(\theta)$  is a directional derivative of  $\kappa_n(\theta)$  in the direction denoted by angle  $\theta$  in the tangent plane. The Mehlum-Tarrou energy has a complicated closed-form expression (Equation 34 in [MT98]), but can be simply expressed by using our third-order shape parameters from Chapter 3. In particular, we show that the Mehlum-Tarrou energy is the sum of squares of the amplitudes of the two Fourier components that define third-order surface behavior.

Mehlum-Tarrou energy

$$\begin{aligned} &= \frac{1}{\pi} \int \left( \int_0^\pi \kappa'_n(\theta)^2 d\theta \right) dA, \\ &= \frac{1}{\pi} \int \left( \int_0^\pi (F_1 \cos(\theta + \alpha) + F_3 \cos 3(\theta + \alpha + \beta))^2 d\theta \right) dA, \\ &= \frac{1}{2} \int F_1^2 + F_3^2 dA. \end{aligned} \quad (4.15)$$

In the above derivation, we use the orthogonality of the cosine function:  $\int_0^\pi \cos(\theta + \gamma) \cos 3(\theta + \delta) d\theta = 0$  for any constants  $\gamma$  and  $\delta$ . Thus, the Mehlum-Tarrou energy is independent of the phase shift  $\beta$  of the third Fourier component.

In this thesis, we will study the scale-invariant version of the Mehlum–Tarrou functional and refer to it as

$$\text{SI-Mehlum–Tarrou energy} = \frac{1}{2} \int F_1^2 + F_3^2 dA \int dA. \quad (4.16)$$

The SI-Mehlum–Tarrou energy can also be used to differentiate between different cyclides. Like the SI-MVSCross energy, the SI-Mehlum–Tarrou energy is zero for two cyclides: spheres and cylinders, and also designates the infinitely thin torus as the optimal genus-1 shape.

We can modify Equation 4.16 to obtain a weighted form of the SI-Mehlum–Tarrou energy, where the weights are used to favor or ignore the first or third Fourier components of the curvature derivative function,

$$\text{weighted SI-Mehlum–Tarrou energy} = \frac{1}{2} \int w_1 F_1^2 + w_3 F_3^2 dA \int dA. \quad (4.17)$$

Similarly, we can obtain an energy functional that measures the amplitude of only one of the two components of the curvature derivative function:

$$\text{SI-}F_1^2 \text{ energy} = \int F_1^2 dA \int dA, \quad (4.18)$$

$$\text{SI-}F_3^2 \text{ energy} = \int F_3^2 dA \int dA. \quad (4.19)$$

### Other Third-Order Energies

There are other third-order energies that we do not consider as candidate functionals for shape design. We will list some of them here and explain why we chose not to investigate them further.

**Cross Curvature Energy:** Similar to the Mehlum–Tarrou energy that measures the average magnitude of inline derivative of normal curvature, we could formulate an energy that measures the average magnitude of the *cross* derivative of normal curvature.

Cross curvature energy

$$\begin{aligned}
&= \frac{1}{\pi} \int \left( \int_0^\pi \kappa_n^\times(\theta)^2 d\theta \right) dA \int dA, \\
&= \frac{1}{\pi} \int \left( \int_0^\pi \left( -\frac{F_1}{3} \sin(\theta + \alpha) - F_3 \sin 3(\theta + \alpha + \beta) \right)^2 d\theta \right) dA \int dA, \\
&= \frac{1}{2} \int \frac{F_1^2}{9} + F_3^2 dA \int dA.
\end{aligned} \tag{4.20}$$

We used Equation 3.38 to simplify the expression for  $\kappa_n^\times(\theta)$ . The cross curvature energy is equal to the weighted SI-Mehlum–Tarrou energy (Eqn 4.17) for a fixed choice of  $w_1$  and  $w_3$  and does not offer any new information not already provided by the weighted SI-Mehlum–Tarrou energy. Therefore we do not consider the cross curvature energy as a functional for aesthetic design.

**Average Curvature Derivative Energy:** We can go one step beyond the cross curvature energy and define an energy that measures, in essence, the average value of all normal curvature derivatives at a surface point. That is, we can measure the average value of the average value of the square of the directional derivative of normal curvature over all directions. That is,

average curvature derivative energy

$$\begin{aligned}
&= \int \left( \int_0^\pi \frac{1}{\pi} \left( \int_0^\pi \frac{1}{\pi} \left( \frac{d\kappa_n(\theta)}{d\mathbf{e}_\psi} \right)^2 d\psi \right) d\theta \right) dA \int dA, \\
&= \int \left( \int_0^\pi \frac{1}{\pi} \left( \int_0^\pi \frac{1}{\pi} (\kappa_n(\theta)' \cos \psi + \kappa_n(\theta)^\times \sin \psi)^2 d\psi \right) d\theta \right) dA \int dA, \\
&= \int \left( \frac{1}{\pi} \int_0^\pi \kappa_n'(\theta)^2 + \kappa_n^\times(\theta)^2 d\theta \right) dA \int dA, \\
&= \text{SI-Mehlum-Tarrou energy} + \text{cross curv. energy}. \tag{4.21}
\end{aligned}$$

Again, since the cross curvature energy is equivalent to a weighted SI-Mehlum-Tarrou energy, the average curvature derivative energy does not offer any new information. Therefore, we will not consider the average curvature derivative energy as a functional for aesthetic design.

**Gravesen's Energies:** Gravesen [Gra03] describes 18 third-order invariants on the surface, each of which can be used as a functional. These are variations of the Mehlum-Tarrou energy, with different weights and multiplicities with which the first-order and second-order terms are considered. Gravesen's work also describes the algebra that can be used to define other third-order energies like the variation of mean curvature or the variation of Gaussian curvature. All the functionals presented by Gravesen [Gra03] can be expressed as a combination of  $F_1$  and  $F_3$  terms. Therefore, we do not need to study Gravesen's functionals further in this thesis.

One way to classify third-order functionals is to consider the task of minimizing the curvature variation of *curves* on the surface. Depending on the subset of the surface curves that we choose to optimize, we can classify third-order surface functionals. In this thesis, we have paid special attention to normal section curves. The MVS and MVS<sub>CROSS</sub> functionals measure curvature variation along of lines of curvature, while the Mehlum-Tarrou functional



measures curvature variation along all normal section curves. The functionals in [Gra03] that contain information not already considered by the MVS,  $MVS_{\text{cross}}$  or Mehlum–Tarrou functionals measure curvature variation over *all* surface curves (including those that are not normal section curves) [GU01]. Since we care only about the shape of the surface and not about its metric distortion, we do not try to optimize curvature variation over curves that are not normal section surface curves. Therefore, we chose to ignore the functionals in [GU01] and [Gra03].

To summarize, we need to study the  $SI-F_1^2$  energy (Eqn. 4.18),  $SI-F_3^2$  energy (Eqn. 4.19), and their combination, the weighted SI-Mehlum–Tarrou energy (Eqn. 4.17) as third-order functionals for shape design.

### 4.3 Combining Energy Functionals

In [Jos07], we show that a designer can combine second-order and third-order functionals to produce new functionals. For instance, in [Jos07], we wanted to produce a functional that designated a torus of radius ratio 0.5 as the most beautiful genus-1 surface. We formulated the combined functional,  $0.95 \text{ bending energy} + 0.05 SI-MVS_{\text{cross}} \text{ energy}$ . Similarly, we can combine the second-order bending energy with the third-order energies described in this thesis to produce a combined functional with different preferred shapes. In Section 8.6 we show some examples of combined energy functionals.

## 4.4 Scale Invariance of Functionals

A surface energy functional is scale invariant when the same shape at different scales produces the same energy. Without scale invariance, the optimization system could keep decreasing the surface energy by infinitely increasing (or decreasing) the size of the surface without changing its shape. Therefore, scale invariance is a desirable property of a surface functional.

The area integral of the sum of squares of the (second-order) principal curvatures is scale invariant. The dimensions in terms of mass (M), length (L), and time (T) of the  $\kappa_1^2 + \kappa_2^2$  term are  $L^{-2}$ , while the dimensions of the area element  $dA$  are  $L^2$ . Their product,  $(\kappa_1^2 + \kappa_2^2)dA$  (see Eqn. 4.10), is dimension-less, and therefore the bending energy is scale invariant.

On the other hand, the third-order terms consisting of squares of  $F_1$  and  $F_3$  have dimensions of  $L^{-4}$ . Their product with the area element is *not* scale invariant, but has dimensions of  $L^{-2}$ . As a result, as shown by Séquin et al. [SCM95], any third-order energy like  $\int F_1^2 + F_3^2 dA$  needs to be multiplied by an additional  $\int dA$  term to make it scale invariant (as in Equation 4.18, 4.19, 4.17).

Ideally, we would be able to formulate third-order surface functionals without the extra  $\int dA$  term. It is unclear how much influence the extra  $\int dA$  term has on the optimal shapes, and whether the optimization tends to favor “skinny” shapes (with lower surface area) due to the extra  $\int dA$  term. We can build such a functional by computing the square root of the curvature derivative terms. For instance, we can construct a scale-invariant form of the SI-Mehlum–Tarrou energy (Equation 4.16) by noting that the term  $\sqrt{F_1^2 + F_3^2}dA$  is

dimension-less.

$$\sqrt{F_1^2 + F_3^2} \text{ energy} = \int \sqrt{F_1^2 + F_3^2} dA. \quad (4.22)$$

However, this functional is not always smooth: at surface points where both  $F_1$  and  $F_3$  are zero, the energy gradient is undefined. Most optimization routines, including the one we used in Chapter 6, assume that the energy space is smooth and that the energy gradient is defined at all surface configurations [BMMS05]. The behavior of the optimization process when the surface reaches a configuration with an undefined gradient is difficult to predict. Therefore, smooth functionals are usually necessary for numerical stability during the optimization process.

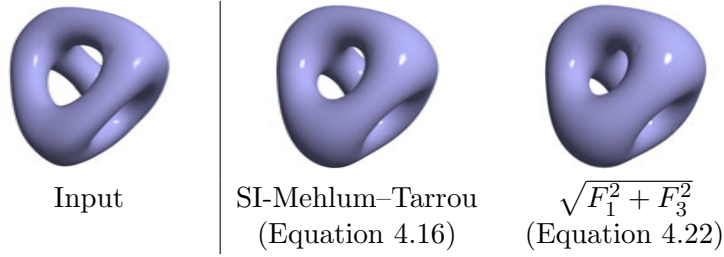


Figure 4.1. For the same input shape (left), we show the optimal shapes with respect to the SI-Mehlum-Tarrou energy (middle) and the  $\sqrt{F_1^2 + F_3^2}$  energy (right).

As a test, we optimized the  $\sqrt{F_1^2 + F_3^2}$  energy (Equation 4.22) for a few canonical input shapes. We ignored the surface points where both  $F_1$  and  $F_3$  were zero while computing the energy gradient. This is equivalent to setting the energy gradient at those points to zero. This strategy is not theoretically sound but seemed to have no adverse effects on the optimization. Our optimization system (described in Chapter 6) remained stable throughout the optimization process and we were able to obtain the optimal shapes without the extra  $\int dA$  term. However, the optimal shapes with respect to the  $\sqrt{F_1^2 + F_3^2}$  energy (Eqn. 4.22) were very similar to the optimal shapes with respect to the SI-Mehlum-Tarrou energy (Equation 4.16) with the additional  $\int dA$  term (see Figure 4.1 for one example).

This indicates that the additional  $\int dA$  did not strongly influence the optimization to favor skinny shapes with smaller surface area. Therefore, in the rest of this thesis, we will consider only the smooth versions of the third-order functionals (with the  $\int dA$  term).

## Chapter 5

# Surface Representation

For any surface optimization system, the choice of the surface representation is crucial. The surface representation affects how easily a designer can specify the input shape and set constraints. The surface representation is also an important factor in the computational cost (memory and time) of optimization — a poorly chosen surface representation can make even simple optimization tasks extremely slow and memory intensive. In this chapter, we describe the method we use to represent surfaces in our optimization system and provide an explanation of our choices.

Depending on the degree of control we want over the surface, we can choose different methods of surface representation. For low-level, detailed control, we could use a densely tessellated polygon mesh or a point cloud, where the mesh vertices or the points are the shape control parameters. Unfortunately, meshes or point clouds need to be rather dense to accurately represent smooth shapes. On the other hand, for high-level, global control, we could use implicit surfaces or a Boolean combination of simple primitives (i.e. constructive solid geometry), where complex shapes can be represented using only a handful of control

parameters. Unfortunately, in such a representation, specifying low-level control such as position constraints can be difficult.

When choosing a shape representation for optimization-based shape design, we need to pick a representation with low-level control as well as the ability to represent smooth shapes using a relatively few control parameters. We found that a parametric surface representation (i.e. spline patches) was convenient and efficient. We could follow the work of Kjellander [Kje83] and represent surfaces using a  $C^2$  continuous, bi-cubic B-spline patch network. However, to model a wide variety of shapes, we would need to handle the inconvenient topological restrictions imposed by a B-spline patch network. On the other hand, Catmull–Clark subdivision surfaces [CC78] offer the freedom to model shapes of arbitrary topology while maintaining the benefits of a B-spline patch network. Therefore, we chose to use Catmull–Clark subdivision surfaces in our optimization system.

## 5.1 Catmull–Clark Subdivision Surfaces

The control polyhedron (i.e. coarse mesh) of the subdivision surface is stored on disk as a single, connected quadrilateral mesh. Each quadrilateral and its immediate neighbors denote the control structure for one surface patch (Fig. 5.1). We compute the surface patch corresponding to any quadrilateral that contains an extraordinary vertex (number of neighbors, i.e. valence  $\neq 4$ ) by using Stam’s method of exact evaluation of the Catmull–Clark limit surface [Sta98]. This allows us to treat our extraordinary patch like any other B-spline patch such that the surface position at any point within the patch can be computed as a closed-form linear combination of the control points. Note that the Catmull–Clark limit surface corresponding to a quadrilateral containing only regular vertices (valence =

4) is exactly the same as the bi-cubic B-spline patch corresponding to that quadrilateral. Therefore, any regular patch is evaluated as a bi-cubic B-spline patch. The control vertices of the subdivision limit surface (i.e. the mesh vertices) are provided to the optimization system as degrees of freedom.

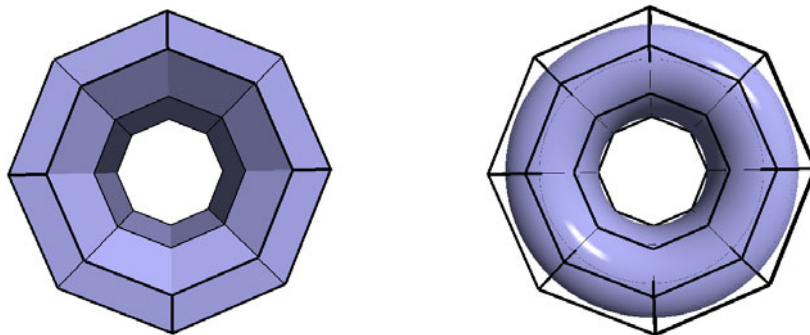


Figure 5.1. The input shape is read in as a quadrilateral mesh (left) which is used as a control polyhedron for the surface (right). This surface is used for all energy computations during surface optimization.

As mentioned before, the main advantages of Catmull–Clark subdivision surfaces are the freedom to model shapes of arbitrary topology and the ability to represent smooth shapes using relatively few control parameters. Another key advantage, which is particularly crucial for surface optimization, is that routines that provide the energy and energy gradient values for each patch are efficient, closed-form, and easy to code.

### 5.1.1 Removing $C^2$ Discontinuity by Blending

Unfortunately, the  $C^2$  discontinuity near an extraordinary vertex of the Catmull–Clark limit surface means we cannot reliably compute curvature derivatives near that vertex. Therefore, we need to modify the the Catmull–Clark limit surface by smoothly blending it with a  $C^2$ -continuous surface. The resulting blended surface is smooth enough to yield reliable curvature derivative values.

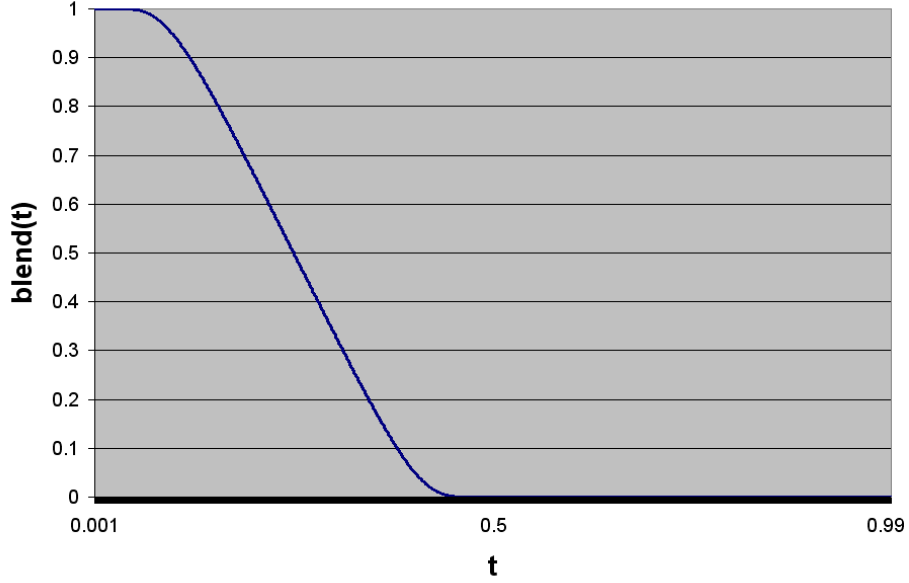


Figure 5.2. The graph shows the weight function from [YZ04, BK01] used to blend the flat spot with the Catmull–Clark surface near the extraordinary vertex. The function starts with a value of one at the extraordinary vertex and smoothly drops to zero halfway along the edges adjacent to the vertex.

We use the simple solution of “ironing” out the  $C^2$  surface discontinuity, similar to the *flatness* parameter proposed in Biermann et al. [BLZ00]. First, we compute the projection of the subdivision limit surface near an extraordinary vertex on that vertex’s limit tangent plane. The new surface is then defined as a smooth blend between the subdivision limit surface and its projection on the limit tangent plane of the extraordinary vertex. To blend the two surfaces, we use the infinitely smooth ( $C^\infty$ ) blend function [YZ04, BK01]

$$\text{blend}(t) = \frac{e^{\frac{2e^{-1/(2t)}}{2t-1}}}{e^{\frac{2e^{-1/(2t)}}{2t-1}} + e^{\frac{2e^{-1/(1-2t)}}{-2t}}} \quad (5.1)$$

In  $u, v$  parameter space, the extraordinary vertex is assigned  $(u, v)$  values of  $(0, 0)$ . Since the  $C^2$  discontinuity affects the Catmull–Clark limit surface to a radius of 0.5 in  $u, v$  space [Lev06], the blend function approaches a limit of 1.0 (as  $t$  approaches zero) at the extraordinary vertex, and a limit of zero (as  $t$  approaches 0.5) halfway towards the ring of the neighboring vertices (see Figure 5.2). At a point with parameter values  $u, v$ , the blend



weight is

$$\text{weight} = \text{blend}(u) \cdot \text{blend}(v) \quad (5.2)$$

and the modified,  $C^2$ -continuous surface at that same point is computed as

$$\mathbf{S}(u, v) = (1 - \text{weight}) \cdot \mathbf{S}_{\text{Catmull-Clark}}(u, v) + (\text{weight}) \cdot \mathbf{S}_{\text{Flat Spot}}(u, v). \quad (5.3)$$

As a result of this modification, the surface is unchanged beyond the midpoint of the edges adjacent to the extraordinary vertex. Near the extraordinary vertex, the surface is a blend of the tangent plane projection and the Catmull–Clark subdivision limit surface. Similar to the requirement for exact evaluation in Stam’s system [Sta98], we need to isolate the extraordinary vertices so that each face is a quadrilateral with a maximum of one extraordinary vertex.

The main advantage of the above surface representation is simplicity: the routines to compute the subdivision limit surface, flat surface, and blending function are very easy to code. Energy queries on these blended patches are as efficient as those on the regular B-spline patches. Any undesirable, unattractive artifacts of the flat spot are reduced during optimization. Subdividing the mesh reduces the influence of the flat spot even further. However, a better starting surface (with the same efficiency) may be obtained by blending a quadric surface near the extraordinary vertex with the subdivision limit surface, as proposed by Levin [Lev06] and Zorin [Zor06].

### 5.1.2 Boundary Patches

Catmull–Clark subdivision surfaces can be modified so as to interpolate a given boundary curve. Biermann et al. [BLZ00] describe a system where the subdivision limit surface

near an open boundary can be specified with position and normal constraints. Such a system can be used to interpolate a given boundary curve during a design task.

In our current implementation, the patches corresponding to the quadrilaterals on the boundary are ignored and do not contribute to the energy of the surface. For all our examples with an open boundary, the vertices on the boundary and their two-ring neighbors are held fixed as constraints. This amounts to constraining the position, tangents and curvatures of the limit surface points on the boundary (i.e. specifying  $C^2$  constraints).

### 5.1.3 Maintaining Sharp Features in Input Surfaces

Sharp features were introduced in subdivision surfaces by Hoppe et al. [HDD<sup>+</sup>94]. DeRose et al. [DKT98] showed how to apply the same sharp features (creases, spikes, darts) to Catmull–Clark subdivision surfaces. They also explained how to model semi-sharp features such that the feature is not infinitely sharp but has a non-zero fillet. Biermann et al. [BLZ00] further improved the method of specifying sharp features by improving the surface behavior of sharp features near the boundary. Overall, the current technology of Catmull–Clark subdivision surfaces has evolved to the point where it can be used for design tasks that require the surface to interpolate any given sharp features such as creases or spikes.

Since we focus on the design of smooth shapes, we assume that there are no sharp features (like creases) that are intentionally placed by the user. Therefore, in our system, the implementation of Catmull–Clark surfaces cannot handle sharp features.

## Chapter 6

# Optimization System

### Introduction

In this chapter we will describe the numerical optimization system we used for minimizing surface energy functionals. Our goal was to build a system that was convenient and fast enough to compute the functional minimizers of some canonical input shapes so that we could compare the shapes preferred by the functionals. As a result, robustness (to bad mesh quality) and accuracy were more important than speed. For the construction of an optimization system for *interactive* design, in Chapter 7 we discuss some options to speed up the system.

The surface optimization system can be separated into two independent components: methods for computing the surface energy and gradient (Section 6.1) and optimization routines (Section 6.2) that modify the degrees of freedom to obtain the optimal shape.

## 6.1 Energy Computation

The optimization routines may require the energy and gradients several times during energy minimization. For that reason, fast energy and gradient computations are essential. We can speed up our energy computation by pre-computing most of the information needed to obtain the energy or gradient at a surface point.

### 6.1.1 Pre-processing

As described in Chapter 5, we represent our shapes using Catmull–Clark subdivision surfaces. The surface is stored on disk as a quadrilateral mesh and read into memory as a half-edge [Wei85] data structure. Each quadrilateral in the mesh denotes a bi-cubic B-spline patch parameterized by the standard  $u$ - $v$  unit-square parameterization. We use a fixed sampling scheme for all patches (see Section 6.1.2); that is, each patch is sampled at the same  $(u, v)$  values. As a pre-process, immediately after the mesh is loaded into memory, we compute the 16 control points and B-spline basis weights for the position  $\mathbf{S}(u, v)$  and 9 parameteric derivatives ( $\mathbf{S}_u(u, v)$ ,  $\mathbf{S}_v(u, v)$ ,  $\mathbf{S}_{uu}(u, v)$ ,  $\mathbf{S}_{vv}(u, v)$ ,  $\mathbf{S}_{uv}(u, v)$ ,  $\mathbf{S}_{uuu}(u, v)$ ,  $\mathbf{S}_{uuv}(u, v)$ ,  $\mathbf{S}_{uvv}(u, v)$ ,  $\mathbf{S}_{vvv}(u, v)$ ) corresponding to each  $(u, v)$  sample point for each quadrilateral patch. After doing so, to extract the position or parametric derivatives at parameter values  $(u, v)$  for a given patch, we need to simply linearly combine positions of the control polygon of the patch.

### Memory Consumption of Pre-processing

For all our examples, the coarse control meshes for the input surfaces were designed by hand instead of being produced by a surface scanner. Most of our coarse meshes start

with less than 500 vertices, and the largest meshes we can handle have about 10,000 vertices. For optimizing a surface with a much larger control mesh, the Catmull-Clark surface representation as described above may require a prohibitively large amount of memory.

To compute energy and energy gradients efficiently, each patch must store pointers to all the control points along with the weights that are pre-computed from the B-spline basis functions. For example, for all regular patches sampled with  $N \times N$  quadrature points ( $N = 8$  in our case), we need to allocate memory for 16 integers as indices of the control points and  $9 \times 16 \times N \times N$  (=9216 in our case) double-precision floating point numbers as B-spline weights used for computing the 9 parametric surface derivatives up to third order ( $\mathbf{S}_{uu}$ ,  $\mathbf{S}_{uv}$ ,  $\mathbf{S}_{uuu}$ , etc.). Additionally, to compute the energy gradients efficiently, each patch must also allocate memory for  $9 \times 16 \times 3 \times 3 = 1296$  double-precision floating point numbers to store the gradients with respect to the 16 control points of the 9 parametric surface derivatives up to third order. Overall, each regular patch needs about 84KB of memory to store its pre-processing data. The memory consumption of a patch with an extraordinary vertex is even greater. In a nutshell, for fast energy and gradient queries, we need to store a relatively large “stencil” of neighboring information for every parametric surface patch.

We cannot avoid the high memory consumption of a parametric surface patch representation if we need efficient energy and gradient queries. In our implementation, the memory footprint of optimizing a mesh with 8,546 vertices is about 1GB, of which 720MB is due to the pre-processing. The memory cost may become prohibitively large for dense meshes (vertex count much larger than 10,000). If such a surface needs to be optimized, there are two options: (1) use mesh simplification to decimate the input mesh down to a smaller size or (2) replace the pre-computation with computing the neighborhood information and basis weights for the patch on the fly *every time* the patch energy or gradient is required. This

second option will make the energy/gradient computation much slower, but the overall system will still be faster for not needing to access virtual memory on disk. If the optimization task demands that we perform the pre-computation even for a large mesh, we will need to minimize the number of virtual memory accesses. We could do so by re-organizing the patch data so that the data of adjacent patches is stored in nearby memory locations.

### 6.1.2 Computing Surface Energy and Gradient

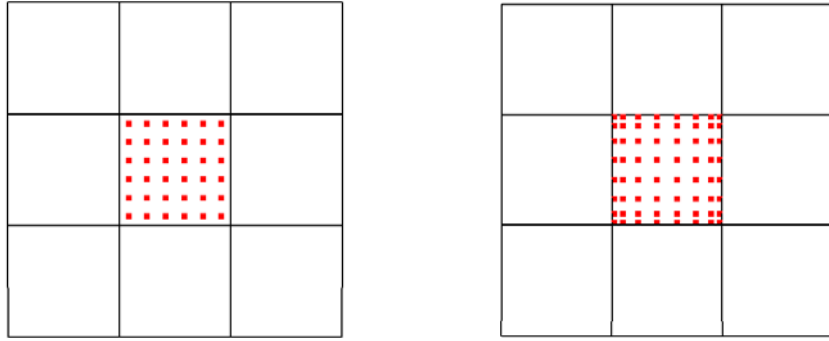


Figure 6.1. A surface patch and its control polygon — instead of uniform sampling (left), we use the Gauss-Legendre sampling (right) to compute surface energy and energy gradients.

The total surface energy is computed as an area integral of the energy of an infinitesimal area element over the entire surface. For instance, if the energy of an infinitesimal area  $dA$  is equal to  $E$ , the total surface energy for the surface domain  $\Omega$  is computed as the integral  $\int_{\Omega} E dA$ .

In practice, the above integral is too difficult to compute analytically; instead, we must compute the integral numerically. That is, the integral is approximated as an area-weighted linear combination of sample values. Numerical integration (i.e. “quadrature”) of the energy of each patch is performed by sampling the patch at fixed  $(u, v)$  values for the energy  $E$ , scaling the energy with the proper area weighting and adding it to the energy of the patch. The total surface energy is the sum over the patches of each patch’s integral.

Compared to a uniform sampling scheme (placing samples at uniform intervals), we experimentally found that the Gauss–Legendre scheme converged faster to a steady number as we increased the sampling density. Additionally, while the Gauss–Legendre scheme places more samples near the boundary of the domain, it does not place any samples on the exact boundary (Figure 6.1). This makes it a suitable scheme for sampling the surface near an extraordinary vertex, as we need to avoid sampling the patch boundaries corresponding to edges containing an extraordinary vertex [Sta98]. Therefore, the Gauss–Legendre quadrature scheme is a good choice for sampling the surface patch domain.

To compute the total surface energy,

$$\text{Energy} = \sum_{p=1}^{\#ofpatches} \sum_{i=1}^N \sum_{j=1}^N w_{i,j} E(p, i, j) \quad (6.1)$$

where  $N$  is the sampling resolution of the patch (in our implementation,  $N=8$ , i.e. 64 samples per patch).  $w_{i,j}$  is the Legendre weight of the  $(i,j)$  sample, computed by solving the orthogonal Legendre polynomials (see [PFTV92] for an implementation).  $E(p, i, j)$  is the energy for the  $(i,j)$  sample in patch  $p$ . For example, if we want to compute the SI- $F_1^2$  energy of a patch  $p$ ,  $E(p, i, j) = \frac{F_1^2}{2} dA$ , where  $F_1$  is computed for the sample point  $(i,j)$ . Similarly, the surface energy gradient is computed by summing up the gradients computed for every sample of every patch.

## 6.2 Optimization

The surface representation provides the shape parameters and the energy and energy gradients as a function of the shape parameters. The optimization routines must modify the parameters to minimize the energy and produce an aesthetically pleasing shape.

### 6.2.1 Input

The unconstrained vertices of the user-provided surface mesh constitute the degrees of freedom for optimization. Let  $\mathbf{X}$  be the vector of vertex coordinates,  $len(\mathbf{X}) = 3 \times \#$  of mesh vertices (100 to 10000 in our case). Given position  $\mathbf{X}$  with individual values  $X_i$ , we can provide routines that compute the energy  $E(\mathbf{X})$  and the gradient vector  $\mathbf{G}(\mathbf{X}) = \partial E(\mathbf{X}) / \partial X_i$ . We set the gradient entries corresponding to constrained vertices to zero, which ensures that the positions of the constrained vertices remain unchanged during optimization. The vector  $\mathbf{X}$  (the initial condition) and the routines for  $E(\mathbf{X})$  and  $\mathbf{G}(\mathbf{X})$  are provided to the optimization function as input.

The optimization routines may modify  $\mathbf{X}$  frequently. To compute  $E(\mathbf{X})$  and  $\mathbf{G}(\mathbf{X})$  efficiently, we store  $\mathbf{X}$  as a continuous one-dimensional array of floating point values in memory rather than as individual floating point triplets within each mesh vertex. We assign an index to the mesh vertices. Given the index and a pointer to  $\mathbf{X}$ , the vertices can access their current position.

### 6.2.2 Increasing Degrees of Freedom

Many of our canonical input shapes are unconstrained, high genus, complicated shapes. We can obtain a considerable speedup during optimization if we use a multi-resolution surface representation, where we start with a coarse control mesh, but during later stages of the optimization, the same surface is represented using a finer control mesh. We use the coarse meshes to bring about large-scale, gross shape changes. We then subdivide the mesh using Catmull–Clark subdivision — the limit surface before and after the subdivision remains the same everywhere except near the extraordinary vertices. Optimization using the



fine meshes allows the surface to more closely approach the ideal, final shape. For example, suppose we are optimizing the bending energy of a coarsely-sampled ellipsoid so that it should deform into a sphere. Catmull–Clark surfaces with a coarse control polyhedron cannot approximate the sphere to sufficient accuracy, so the optimization will stop at a rough approximation of the sphere. However, upon subdivision of the coarse control mesh, the new, finer control mesh will have the necessary degrees of freedom to better approximate a sphere, thereby assisting the optimization to find the true optimal surface.

In all our examples, the minimum energy for a surface with one level of subdivision was very close to the minimum energy for the surface with two levels of subdivision. The difference in energy was less than 0.5% of the energy of the surface with one level of subdivision. Beyond two levels of subdivision, there are usually too many degrees of freedom for the optimization to find the minimum fast enough, and the reduction in the energy value will probably be very small (less than 0.5% of the energy at the previous level of subdivision). Therefore, we perform two levels of subdivision during optimization.

### 6.2.3 Optimization Algorithms

The optimization algorithm needs to modify the degrees of freedom to find the local energy minimum, starting from the initial condition. Most gradient based non-linear optimization algorithms try to build a quadratic model (usually a high-dimensional paraboloid) of the energy space. The quadratic model is given by the energy Hessian at the current configuration. The energy Hessian is the matrix of double derivatives of the energy with respect to the degrees of freedom. That is, given the position vector  $\mathbf{X}$ , the energy Hessian  $= \frac{\partial^2 E(\mathbf{X})}{\partial X_i \partial X_j}$ . The configuration ( $\mathbf{X}$ ) corresponding to the bottom of the paraboloid is a good estimate of the energy minimum; the optimization routine will step towards that configu-

ration. After taking the step, the algorithm will compute the new Hessian and the process will iterate until the energy value converges to a stable number.

Our non-linear optimization task has a relatively large number of degrees of freedom and a small number of simple, fixed-position constraints. A good (usually default [BMMS05]) choice for such a task is a limited memory, quasi-Newton method for optimization. A quasi-Newton method (also known as a “variable metric method” [BMMS05]) iteratively builds an approximation of the energy Hessian when the exact Hessian is not available. In our case, the exact Hessian is very complicated to compute and computationally expensive; not requiring the exact Hessian is an important advantage. A *limited-memory* quasi-Newton method builds the approximation by using only a fixed number of previous gradient values (usually much smaller than the total degrees of freedom), thereby reducing the memory cost of storing the information needed to approximate the Hessian.

**Implementation Details:** We employ the Broyden-Fletcher-Goldfarb-Shanno (BFGS) flavor of the limited-memory quasi-Newton method. We use the Toolkit for Advanced Optimization (TAO) [BMMS05] that contains a reliable implementation of the BFGS algorithm<sup>1</sup>.

Since we are minimizing non-linear, third-order functionals, the energy landscape can be very complicated. Despite having to traverse such a difficult energy space, our optimization system, while not fast enough to be interactive, is not unreasonably slow. Most of the examples in this thesis were computed in less than 3 hours. Most of the discernible shape changes (including the large, global shape changes) took place in the first few hundred

---

<sup>1</sup>Further information for TAO users: we first run the BFGS method using the default Moré–Thuente line search routine. For complicated energy landscapes (especially for third-order functionals) this routine often fails — in that case, we also run BFGS with the Armijo line search routine. We specify 10 as the number of previous energy and gradient values (the “limited memory size”) used for approximating the Hessian.

iterations, which, depending on mesh size, took about 2 to 10 minutes. All the computation was performed on a single-core Intel 2.8GHz. CPU with 2.5GB of RAM. In Chapter 7, we will describe some ideas to speed up the optimization.

## Chapter 7

# Options for Fast Optimization

So far, we have described our robust benchmark system that is useful for studying the nature of the shapes preferred by functionals. Speeding up the optimization is essential if optimization is to be used as a tool for *interactive* design. In this chapter, we will describe some options towards that goal, with our experiences when applicable.

### 7.1 Discrete Geometry Operators for Energy Queries

Computing the surface energy and gradient using numerical integration over parametric spline patches can be slow. A faster option is *directly* approximating the relevant geometric information (like mean curvature) at a control mesh vertex as the function of the vertex and its neighbors. The recently popular [Gri06] discrete differential geometry operators provide this option. We used these operators for second-order (bending energy) minimization — our hope was that using the discrete operators would considerably speed up optimization. We used the edge based operator from Grinspun et al. [GHDS03] that computes the bending at a mesh edge (i.e. the mean curvature of that mesh edge) as a weighted function of the

dihedral angle made by the adjacent two faces at that edge. We adapted the triangle-based discrete operator to handle quadrilaterals by computing the average bending across the diagonals of the quadrilateral, in addition to computing the bending across the existing mesh edges<sup>1</sup>. This amounts to creating a “virtual triangulation” of the quadrilateral mesh and computing the bending energy using that triangulation.

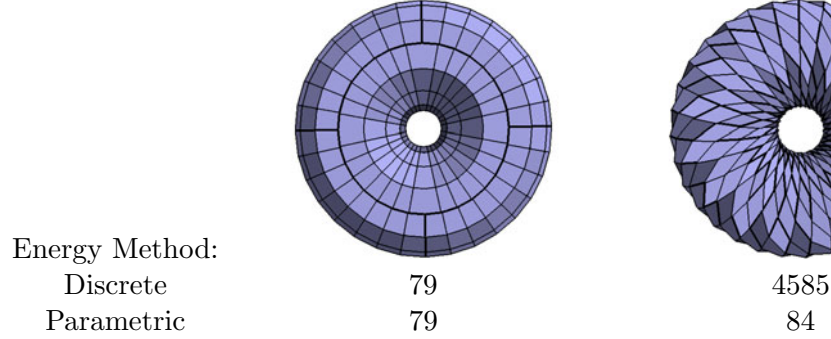


Figure 7.1. Comparison of the calculated energy values for two different mesh structures on the same torus

Unfortunately, discrete operator energy values of a coarse mesh with large dihedral angles are not always reliable. For instance, Figure 7.1 shows the effect of mesh quality on the energy returned by discrete operators for a coarse mesh. We uniformly sampled a “Clifford” torus (the genus-1 bending energy minimizer, with a known bending energy of  $25\pi \approx 78.54$  [HKS92]) and produced two different meshes by changing the connectivity of the vertices. The energy returned by discrete operators matched the energy returned by our method of parametric spline patches if the mesh had the straight-forward quadrilateral connectivity. However, the energy returned by the discrete operators was significantly higher if the mesh had rhombic connectivity with spiraling edge lines. Furthermore, in this case, optimization using discrete energy deforms the mesh into an infinitely thin torus, which is clearly different from the expected minimum, the Clifford torus. On the other hand, the

---

<sup>1</sup>thanks to Prof. Eitan Grinspun for this idea

optimization our method of using parametric patches yields a torus very close in shape and energy to the Clifford torus.

Therefore, with a coarse mesh, the discrete operators show a large dependence on the mesh quality. Two meshes with vertices in the same position (i.e. the same “shape”) but different connectivity can produce significantly different energy values. The accuracy of the energy approximation for the smooth surface improves with the fineness of the mesh and also when the mesh edges coincide with the principal directions.

In our optimization system, using discrete energy operators produced the incorrect optimal shapes for coarse meshes with badly aligned edges. Also, our input shapes may be significantly different from the optimized shapes. Therefore, even if we started with a coarse mesh of good quality, during the course of the optimization, the mesh could get deformed such that it no longer had the proper connectivity to return reliable energy values to steer the surface towards its optimal shape. Since the deformed mesh would produce incorrect discrete energy values, the optimization would return an incorrect optimal shape.

Discrete operators can produce reliable energy values if the mesh is dense, such that large dihedral angles are rare. However, using a dense mesh produces another problem: ill-conditioning due to too many degrees of freedom. As described in the next section, a dense, smooth mesh will move much more slowly towards the optimal shape than a coarse mesh. While using a dense mesh improves the quality of the energy approximation, the increased degrees of freedom cause the optimization to converge much slower to the optimum. Therefore, using discrete operators with dense meshes produced results much more slowly than using the spline patches with coarse (control) meshes.

We believe that using discrete operators to compute surface energy is assigning undue

importance to the mesh elements. Mesh elements like vertices, faces and edges are the wrong modeling primitives for the kinds of surfaces we wish to produce. The optimal shapes with respect to the functionals mentioned in this thesis can be described at a high level as a blend of spheres, toroidal arms, and generalized cylinders. Therefore, we believe that a promising area of future work (Section 9.3) is to use such higher-level modeling primitives to represent surfaces with fewer but directly relevant shape parameters (such as the radius of a sphere) and thereby speed up optimization without sacrificing accuracy. Perhaps we could use implicit surfaces to model the primitives and their unions. The main challenge will be to ensure that with these primitives, we can still efficiently model and interrogate surfaces with arbitrarily detailed constraints (just like we can with spline patches).

## 7.2 Addressing Ill-Conditioned Functionals

All the functionals used in this thesis are ill-conditioned: a small change in the position of the control vertices results in a relatively large change in the energy values. This property, in the context of optimization or partial differential equations, may also be called “ill-posed.” The third-order functionals are even more ill-conditioned than the second-order functionals. As a result, a control vertex that is in a locally optimal (i.e. smooth) configuration will be allowed only a small step towards the shape that is globally smoother. For example: upon optimization of the  $SI-F_3^2$  energy of an ellipsoid, we should obtain a sphere. However, if the ellipsoid mesh is dense (large number of degrees of freedom), the deformation towards the sphere will be much slower than if the ellipsoid mesh was coarse (fewer degrees of freedom). In our application, once most of the vertices are in locally optimal configurations, the

optimization takes a long time to deform the global shape into an aesthetically pleasing one.

Our current approach of tackling ill-conditioning is to use multi-resolution surfaces, where the same surface is represented using different resolutions (mesh densities) of control polyhedra. The gross shape of the optimal surface is obtained by manipulating the coarse control polyhedron; after subdivision, the additional degrees of freedom can be used to deform the shape even closer to the optimum. However, even with a multi-resolution surface representation, we often encounter problems due to ill-conditioning. If we start with too coarse a mesh, optimization will produce a poor approximation of the expected optimal shape. After mesh refinement by subdivision, the new degrees of freedom will allow the system to deform the surface towards a better approximation of the optimal shape. But the additional degrees of freedom will significantly slow down the rate of energy minimization and the progress towards the optimal shape by deforming the increased degrees of freedom will be very slow.

After the first hundred or so iterations of the quasi-Newton solver (Section 6.2), the optimization slows down considerably, allowing only small step sizes that produce small energy reductions. One might argue that the convergence slows down because we are using a quasi-Newton method with an approximate Hessian, and not Newton's method with the exact Hessian, and therefore we are seeing a slow, linear convergence rate as opposed to the faster, quadratic convergence rate of Newton's method. However, in any optimization task, the quadratic rate of Newton's method does not emerge until the energy value is very close to the minimum. In our case, since the input shape is usually much different than the optimal shape, even getting close to the optimum can be time-consuming. Therefore, using a Newton method (rather than a quasi-Newton method) may not be worth the effort of



writing the complicated routines and additional computational cost to compute the exact Hessian.

In addition to the multi-resolution surface representation, we need to use a gradient conditioning technique specifically designed for ill-conditioned functionals. Renka [Ren04] explains that in the case of the ill-conditioned functionals typically used for constructing smooth curves and surfaces, the slow convergence might be due to the loss of smoothness during the computation of the energy gradient. The energy functionals typically involve squares of derivative terms up to some order that are integrated over the entire surface. That is, the functionals are in a *Sobolev* space  $H^{k,2}$ , where  $k(=3$  in our case) is the order of differentiation.  $H^{k,2}$  is a subset of the  $L^2$  space (space of square integrable functions). If we use the standard Euclidean gradient to compute the descent direction, we compute gradient norms (that determine a unit change in the functional value) using the Euclidean,  $L^2$  norm. The  $L^2$  norm measures the change in functional value, but ignores information about the change in the derivatives. As a result, the Euclidean gradient of energies in  $H^{k,2}$  at a control vertex can be very “rough” [Ren04] and is independent of the gradients of the vertex neighbors. During optimization, the mesh vertices are moved *simultaneously* towards the optimum, so the gradient at each vertex should also be computed while keeping the neighboring gradients in consideration. This amounts to maintaining the smoothness of an  $H^{k,2}$  function for its gradient, which is accomplished by computing the energy gradient using the higher order Sobolev norm.

### 7.2.1 Sobolev Gradients

For energies in Sobolev space  $H^{k,2}$ , Renka and Neuberger [RN95] describe the use of the “Sobolev” gradient which is calculated by using the underlying, higher-order Sobolev

norm. Their experiments indicate that the number of iterations required to minimize an ill-conditioned functional (such as our functionals) is reduced by up to two orders of magnitude when using the Sobolev gradient as opposed to the usual, Euclidean gradient. While the nature of the optimization problem was different (Renka and Neuberger computed minimal surfaces and minimized curvature variation of curves), we believe that we can significantly improve the time to optimize third-order surface functionals. Therefore, as future work, we plan to use the Sobolev gradient in our system.

## Chapter 8

# Comparison of Functionals

In this chapter we compare the shapes preferred by the energy functionals from Chapter 4 and discuss the suitability of the functionals for shape design. We compare the shapes optimized by each of the functionals for the same input shape. Most of the examples are unconstrained, which ensures that the shape of the final surface is influenced only by the choice of the functional and not by any geometric constraints. We also provide some examples with boundary constraints. For any open surface boundary, we specify  $C^2$  constraints by clamping (holding fixed) the control mesh vertices on the boundary and their two-ring neighbors. This ensures that the position, tangents and curvatures at the boundary of a patch are constrained.

The comparison of optimal shapes in a side-by-side tabular form also helps a designer to develop an intuitive understanding of the different functionals. The designer can anticipate the effect of optimization of a given energy functional by studying the kinds of optimal shapes produced for some canonical input cases. This intuitive understanding is useful when the designer optimizes a different input shape and needs to choose the proper functional or combination of functionals.

As a first step, we optimized some well-known, simple, mathematical surfaces whose shape can be defined with very few shape parameters. For genus 0, optimization of all the functionals studied in this thesis yield a sphere of an arbitrary radius, as all the functionals are scale invariant. We cannot draw any distinctions between the functionals from optimal sphere configurations. For genus 1, we get different torus configurations from different functionals. We find that studying the optimal torus configurations provides us with a better understanding of the different functionals and their combinations.

## 8.1 Experiments on a Torus

A torus is a surface of revolution and has two degrees of freedom: the radius of the circular cross-section and the radius of revolution of the cross-section. For the sake of simplicity, we do not consider any tori where the radius of cross section may vary as it is revolved, nor do we consider tori with self-intersections. Since all our functionals are scale invariant, we can fix the radius of revolution to 1 and investigate which values of the radius of the circular cross section (i.e. a “radius ratio” of the torus) produce energy minimizing configurations. As expected [HKS92], the bending energy minimizer is the Clifford torus, which has a radius ratio of  $1/\sqrt{2}$ . As Moreton and Séquin [MS92] mention, all torus configurations are cyclides; therefore the SI-MVS energy is zero for all torus configurations. The SI-MVS<sub>cross</sub> energy is minimized by an infinitely thin torus [Jos07]. This is because the SI-MVS<sub>cross</sub> energy is zero for all cylinders and the infinitely thin torus is the best approximation of a cylinder that the optimization system can make while still maintaining the topology of a torus. The same, infinitely thin torus is preferred by our third-order energies: SI- $F_1^2$  energy, SI- $F_3^2$  energy, and their sum, the SI-Mehlum–Tarrou energy. For

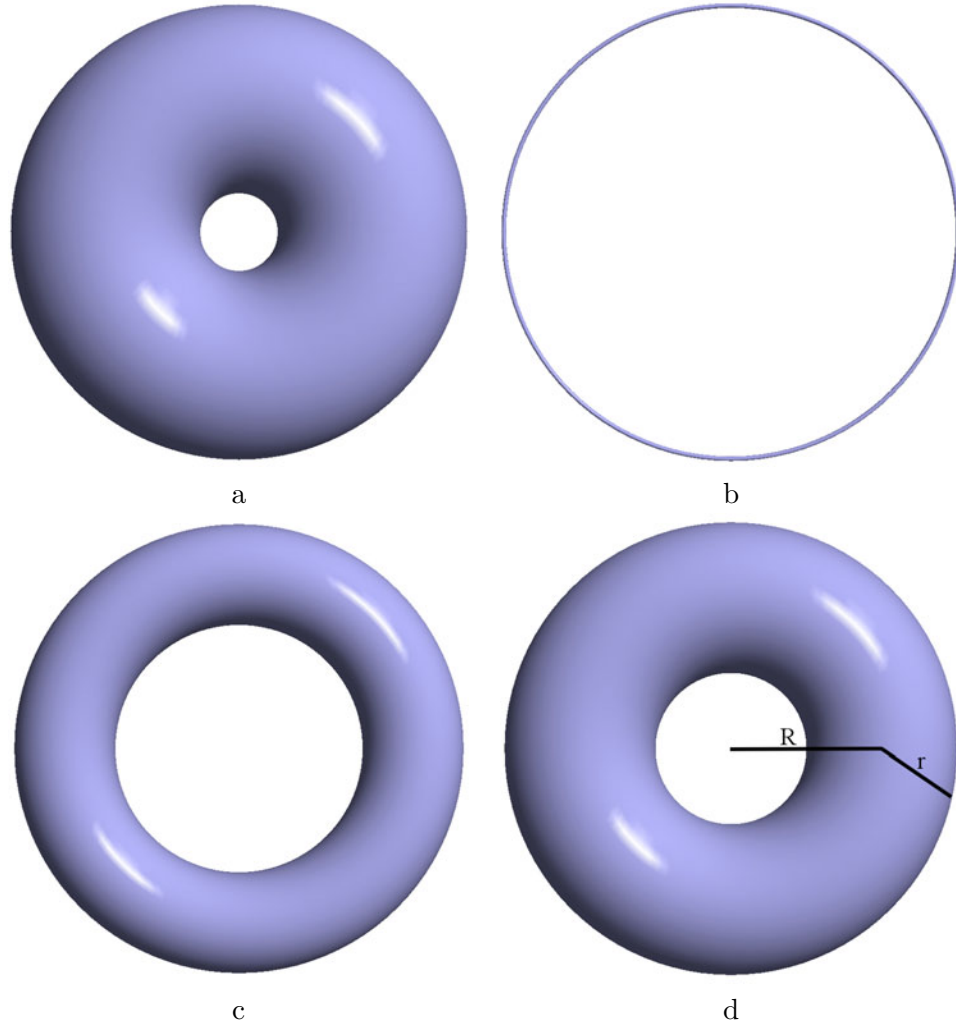


Figure 8.1. (a) The Bending energy (BE) minimizer, i.e. the “Clifford” torus, (b) the  $\text{SI-MVS}_{\text{cross}}$  and SI-Mehlum–Tarrou energy minimizer (an infinitely thin torus), (c) the minimizer of the combined energy:  $\text{BE} + \text{SI-Mehlum–Tarrou}$  (torus radius ratio = 0.285), and (d) the minimizer of the combined energy:  $0.92 \text{ BE} + 0.08 \text{ SI-Mehlum–Tarrou}$  (torus radius ratio = 0.5). (d) also shows the two radii, the radius of cross section (“ $r$ ”) and the radius of revolution (“ $R$ ”) which give the torus radius ratio ( $=r/R$ ).

any torus, the  $\text{SI-}F_1^2$  energy is equal to the  $\text{SI-}F_3^2$  energy, which means the weighted SI-Mehlum–Tarrou energy with any combinations of weights  $w_1$  and  $w_3$  will also yield the infinitely thin torus as the optimizer.

### 8.1.1 Calibrating Weights for Combined Functionals

We obtain different tori when we combine the second-order and third-order energy. We can use the combinations that produce desirable torus configurations to design new combined functionals that have desirable properties. For example, if we consider the bending energy and SI-Mehlum–Tarrou energy with equal weight, we obtain a torus with a radius ratio of 0.285. However, if we want a functional that yields the torus with a radius ratio of 0.5, we need to combine bending energy with a weight of 0.92 and the SI-Mehlum–Tarrou energy with a weight of 0.08. We provide examples of energy minimization using these combined functionals in Section 8.6.

## 8.2 Comparison of Third-Order Energies

Recall that the  $\text{SI-}F_1^2$  energy (Eqn. 4.18) measures the amplitude of the first Fourier component of the third-order curvature derivative function while the  $\text{SI-}F_3^2$  energy (Eqn. 4.19) measures the amplitude of the third Fourier component. Because  $\text{SI-}F_1^2$  energy minimization ignores the undulatory “monkey-saddle”-like surface behavior, the optimal shapes (second column of Figure 8.2) show a relatively high amount of undulation. On the other hand, because the  $\text{SI-}F_3^2$  energy minimization will try to minimize the undulatory behavior, the optimal shapes (third column of Figure 8.2) are relatively simple, with lesser shape variation.

The SI-Mehlum–Tarrou energy measures the average derivative of normal curvature in all directions at all surface points. Observe that many of the corresponding optimal shapes (fourth column of Figure 8.2) are very close to those of the  $\text{SI-}F_3^2$  energy. In the case where a designer uses a weighted SI-Mehlum–Tarrou functional (Equation 4.17), specifying a higher weight for  $F_3$  than that for  $F_1$  will not yield shapes that are significantly different from those found by SI-Mehlum–Tarrou or  $F_3$  optimization. However, if we specify a lower weight for  $F_3$  than for  $F_1$ , we obtain a functional that produces shapes (last column of Figure 8.2) similar to those of SI-Mehlum–Tarrou, but with more undulation.

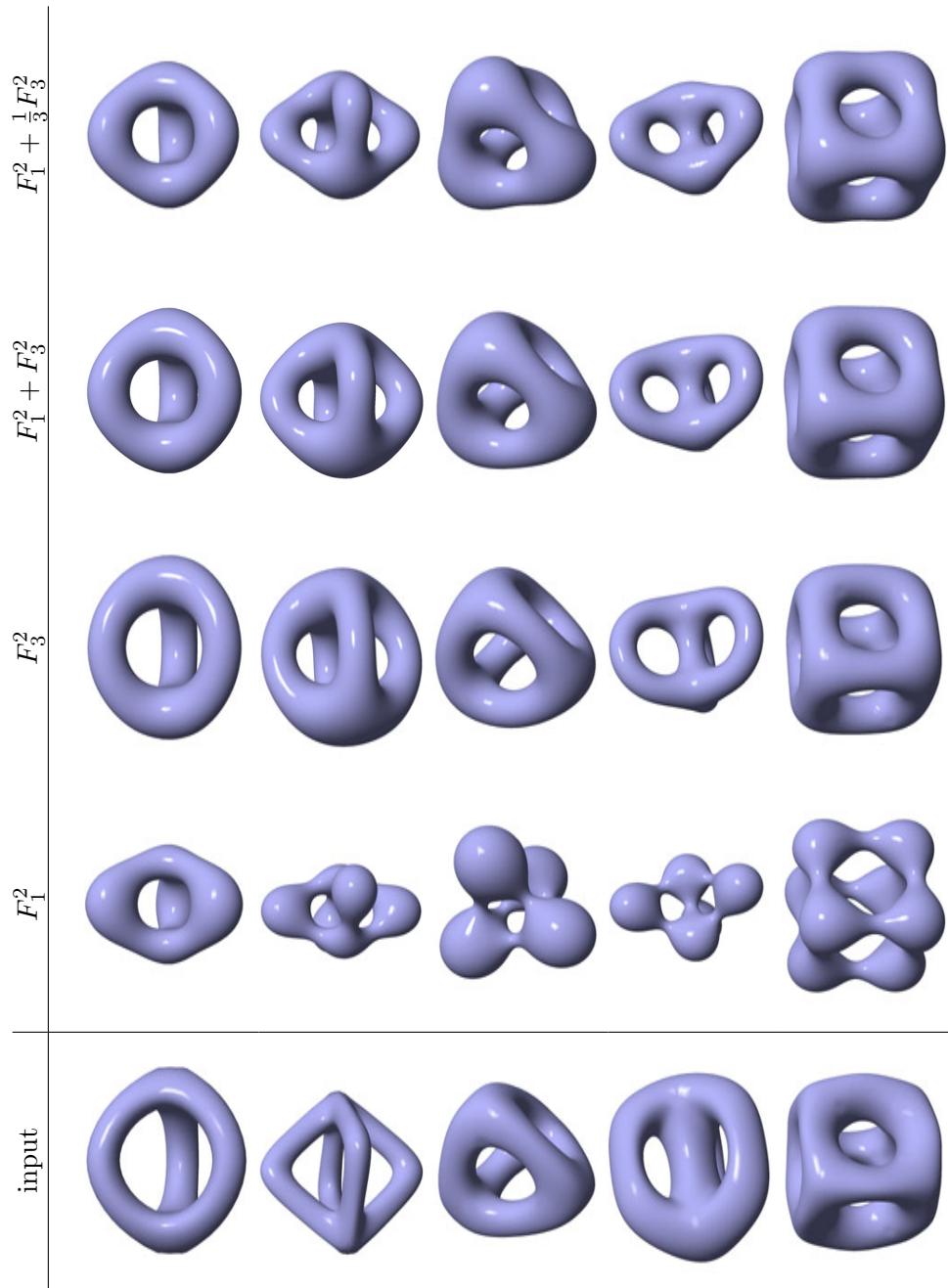


Figure 8.2. Comparison of the shapes preferred by  $SI-F_1^2$  energy,  $SI-F_3^2$  energy, and their combination, the  $SI$ -Mehlum-Tarrrou energy. Specifying a lower weight for  $F_3$  produces a weighted  $SI$ -Mehlum-Tarrrou functional (far-right) with slightly different preferred shapes.



### 8.3 Comparison with MVS Energies

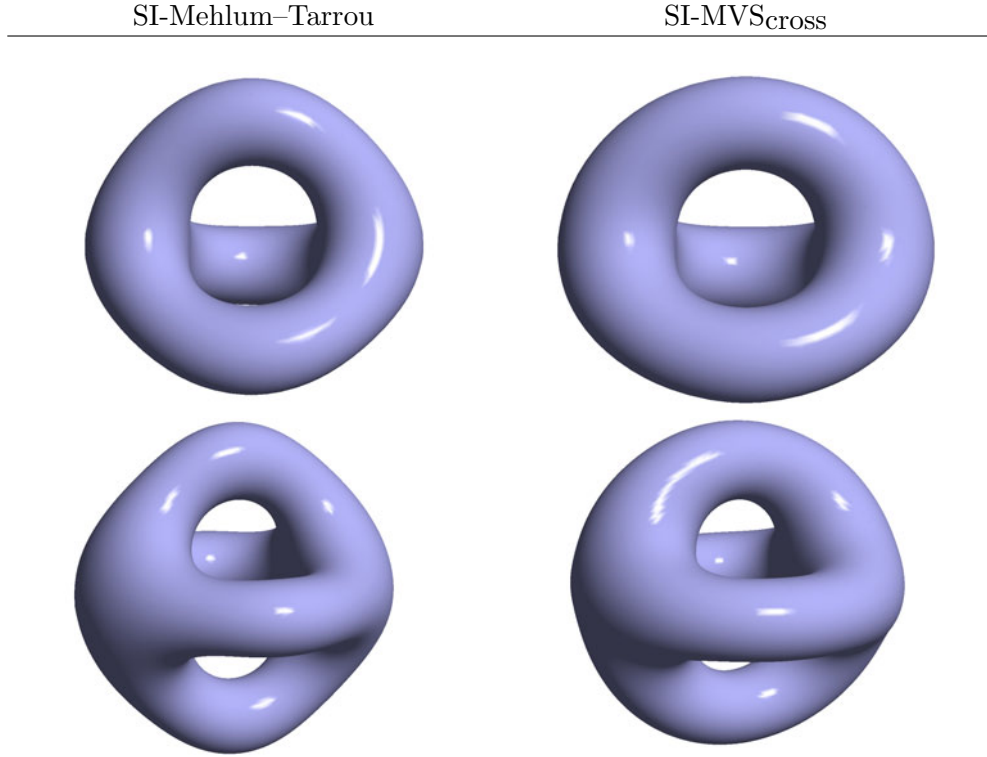


Figure 8.3. Comparison of the SI-Mehlum–Tarrou energy and SI-MVS<sub>cross</sub> energy minimizers. For simple shapes, the SI-MVS<sub>cross</sub> minimizers have rounder envelopes than the SI-Mehlum–Tarrou minimizers.

In Figure 8.4, we compare the minimizers of the SI-Mehlum–Tarrou functional with the third-order MVS functionals (SI-MVS<sub>cross</sub> [Jos07] and the original SI-MVS [MS92]). We compared the SI-MVS<sub>cross</sub> and SI-MVS functionals in my earlier Master’s thesis [Jos07]. We showed that the SI-MVS<sub>cross</sub> functional is a more complete third-order functional than the SI-MVS. Here, we use the SI-MVS<sub>cross</sub> functional as the reference functional to compare the SI-Mehlum–Tarrou functional with.

For simple shapes (see Figure 8.3), the SI-MVS<sub>cross</sub> minimizers may have more circular or toroidal envelopes than those of SI-Mehlum–Tarrou. However, for more complicated

shapes (in Figure 8.4), the minimizers of the SI-Mehlum–Tarrou energy are essentially indistinguishable from those of the SI-MVS<sub>Cross</sub> energy.

Recall that the SI-MVS and SI-MVS<sub>Cross</sub> functionals measure normal curvature variation only along the principal directions. The principal directions are given by the second-order behavior of the surface point and are independent of the third-order behavior. Therefore, giving a special importance to the principal directions for a third-order functional (as done by SI-MVS and SI-MVS<sub>Cross</sub> optimization) is appropriate only if the principal directions are particularly significant for the design task. For a general task of minimizing curvature variation, we recommend the more complete, stable and easier to compute SI-Mehlum–Tarrou functional.

Figure 8.4 also lists the SI-Mehlum–Tarrou and SI-MVS<sub>Cross</sub> energy values. We can follow the analysis of energy values performed in our earlier Master’s thesis [Jos07] and compute the SI-Mehlum–Tarrou energy per blended  $n$ -way junction of toroidal arms, where  $n$  in Figure 8.4 varies from 3 to 4. Like the SI-MVS<sub>Cross</sub> energy, the SI-Mehlum–Tarrou energy favors smaller  $n$  for the  $n$ -way junctions, and the energy increases very rapidly as  $n$  increases. Thus, the SI-Mehlum–Tarrou functional favors a larger number of blended  $n$ -way junctions with a low  $n$  rather than fewer  $n$ -way junctions with a high  $n$ . In a nutshell, like the SI-MVS<sub>Cross</sub> functional, the SI-Mehlum–Tarrou energy returns a lower energy for more symmetric shapes. The detailed analysis is similar to the analysis of the SI-MVS<sub>Cross</sub> energy previously carried out in [Jos07] and is not repeated here.

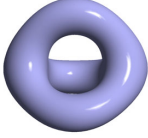
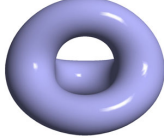
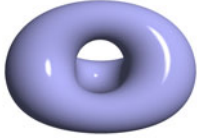






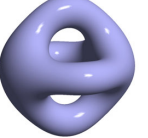
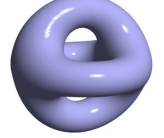

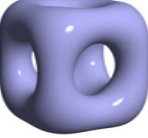
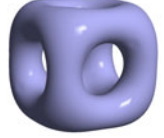
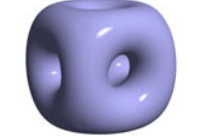
Genus	$n$	SI-Mehlum–Tarrou	SI-MVS <sub>cross</sub>	SI-MVS
2	3			
		1903	2408	343
3	3			
		3320	4020	975
3	4			
		4182	6965	840
3	4			
		4437	6416	1548
5	3			
		6862	9109	2217

Figure 8.4. Optimal shapes preferred by the SI-Mehlum–Tarrou energy (which measures curvature variation along all directions) and the SI-MVS<sub>cross</sub> and SI-MVS energies (which measure curvature variation only along principal directions). Under each optimal shape, we provide the energy value. Both the SI-Mehlum–Tarrou and SI-MVS<sub>cross</sub> energies depend on the valence  $n$  of the blended  $n$ -way junctions.

## 8.4 Comparison with Second-Order Energy

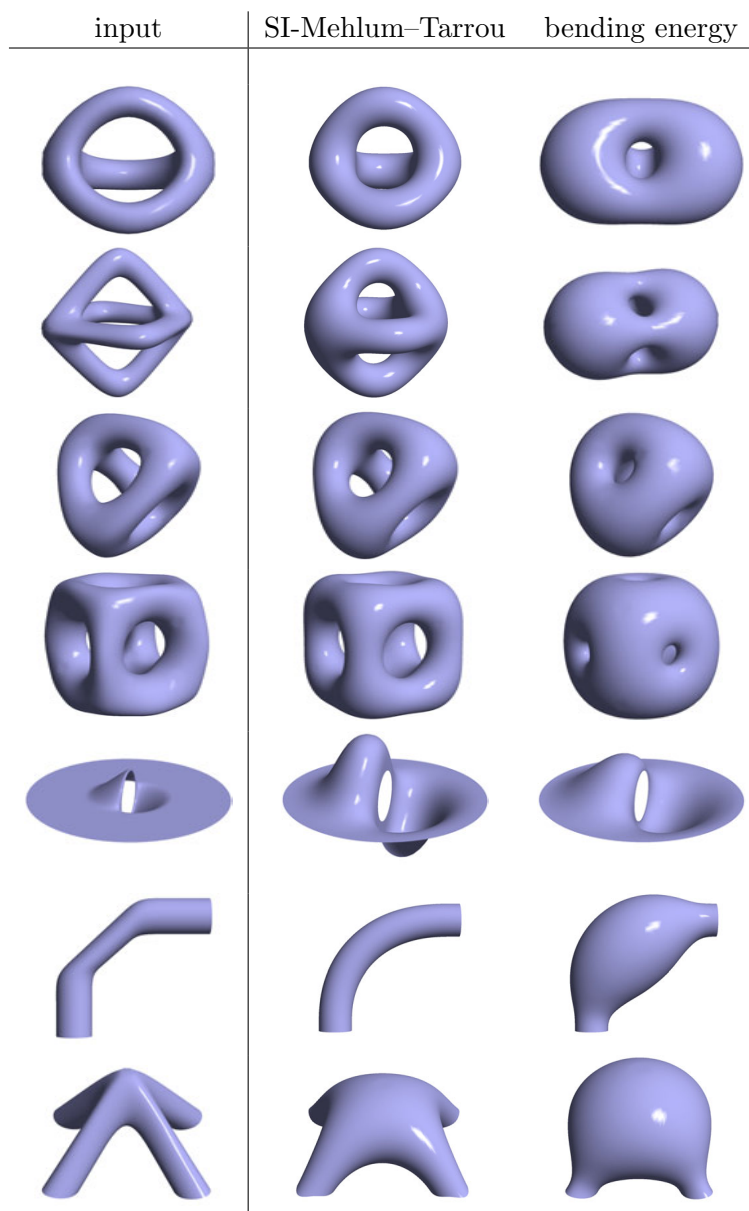


Figure 8.5. Minimizers of the third-order SI-Mehlum–Tarrou energy and the second-order bending energy.

We compare the shapes preferred by the third-order SI-Mehlum–Tarrou energy with those preferred by the second-order bending energy (Figure 8.5). The bending energy optimization tries to make every surface point as umbilic as possible [JS07]. Thus, the optimal shapes look “blobby.” We consider this a drawback of the bending energy, as the effort to

make all surface points umbilic can actually make the optimal shape *more complex* and potentially undesirable. Figure 8.6 shows boundary constrained junctions of two and four toroidal arms, respectively. The bending energy minimization does not penalize large curvature variation but lowers the total curvature by inflating the shape so it has a larger radius of curvature. Thus, the unconstrained, interior regions of the bending energy minimizers bulge out and produce small areas of high curvature to connect smoothly with the constrained surface patches on the boundary.

On the other hand, SI-Mehlum–Tarrou optimization minimizes the *variation* of curvature by trying to maintain the same curvature along the whole shape. This decreases the range of curvature values for points on the optimized surface. Therefore, optimization of an input shape with a third-order functional usually results in simpler and more attractive shapes.

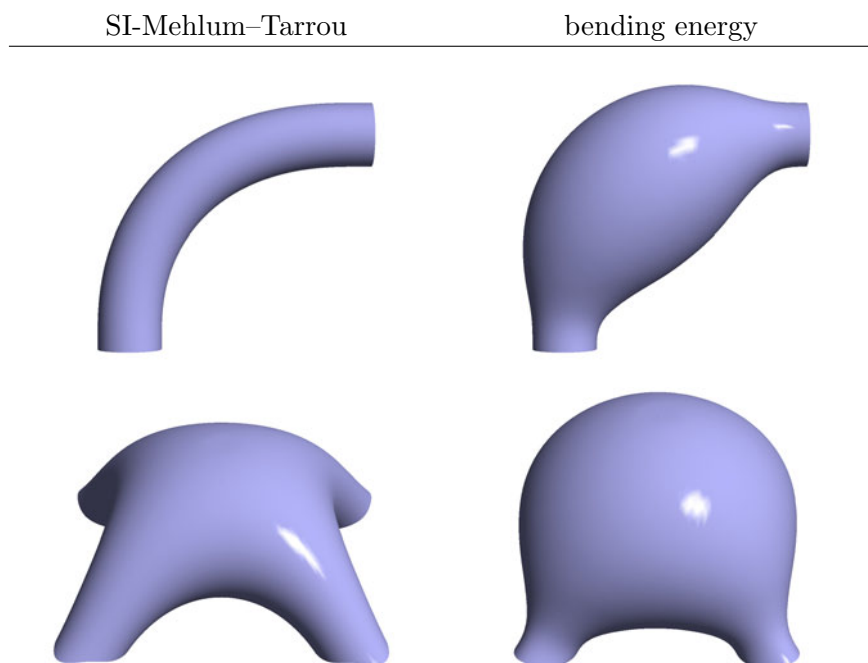


Figure 8.6. Optimization of the third-order SI-Mehlum–Tarrou energy produces simpler shapes with more uniform curvature distribution than those produced by optimizing the second-order bending energy.

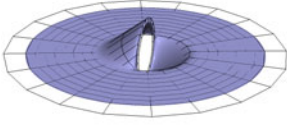
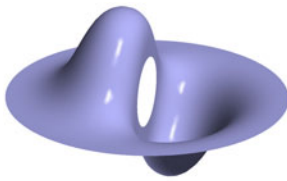
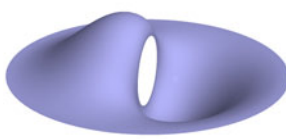
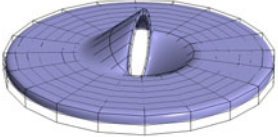
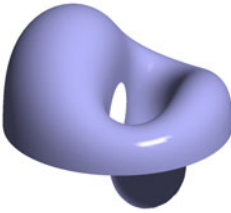
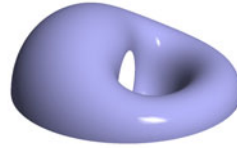
Input	SI-Mehlum–Tarrou	Bending Energy
		
		

Figure 8.7. Optimal shapes with respect to the third-order SI-Mehlum–Tarrou energy and the second-order bending energy for a complicated open surface with an orthogonal, circular hole. The SI-Mehlum–Tarrou energy minimization produces a more complex shape that bulges out near the boundary constraints.

However, depending on the input shape and boundary conditions, the second-order bending energy can produce simpler shapes than the third-order SI-Mehlum–Tarrou functional. For instance, in Figure 8.7, a surface with complicated boundary constraints is optimized. Optimization using the third-order SI-Mehlum–Tarrou functional, in an attempt to satisfy the boundary constraints with a higher level of geometric smoothness (curvature smoothness as opposed to only tangent smoothness), creates a complex optimal shape that bulges out near the boundary constraints. On the other hand, optimization using the second-order bending energy functional produces a simpler and therefore more desirable optimal shape.

## 8.5 Example of Aesthetic Design: Vase

Here we describe how a designer can construct an aesthetically pleasing vase by optimizing the same input shape with respect to the four functionals we have discussed so far: the second-order bending energy and the third-order SI- $F_1^2$  energy, SI- $F_3^2$  energy, and SI-Mehlum–Tarrou energy.

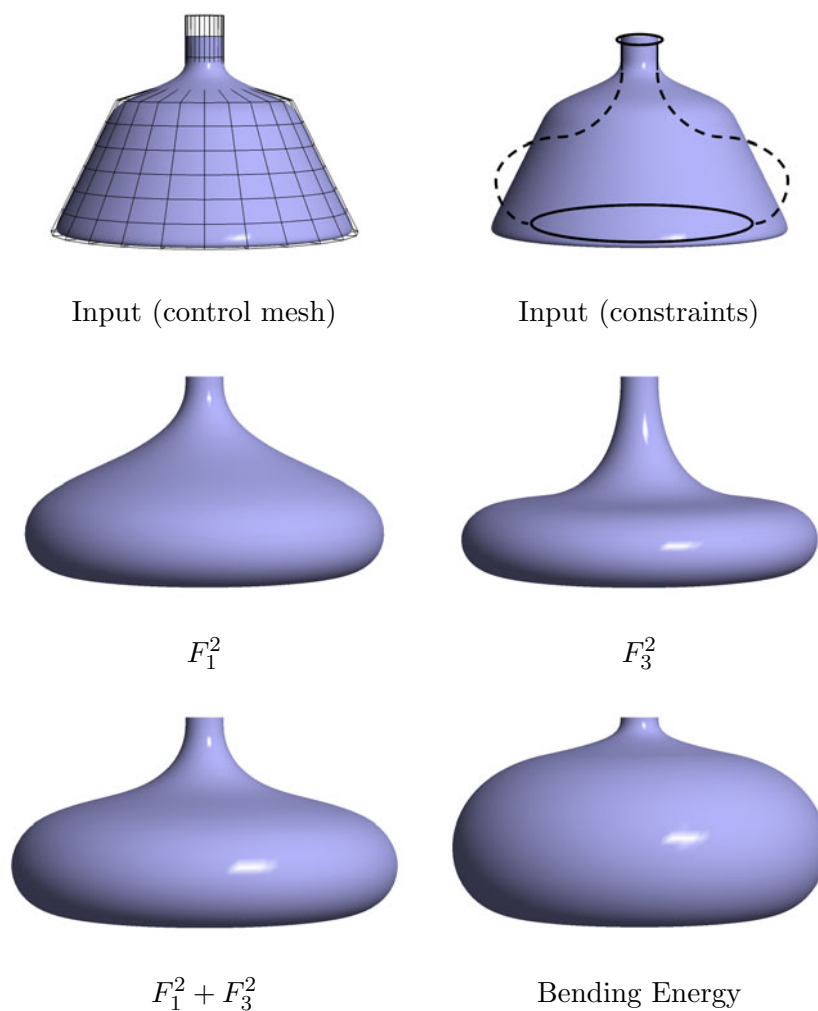


Figure 8.8. Designing a vase. The first row shows the input shape overlaid with the control mesh, and a sketch describing the nature of the boundary constraints and the approximate desired shape. The second row shows the optimal shapes with respect to the SI- $F_1^2$  and SI- $F_3^2$  energies. The third row shows the optimal shapes with respect to the SI-Mehlum–Tarrou energy and the bending energy.

In the first example (Figure 8.8), we start with a cone-like surface open at the tip, with

vertical boundary conditions at the top and horizontal boundary conditions at the bottom. The goal is to construct a bell-shaped vase that smoothly interpolates the constraints. In the second example (Figure 8.9), we start with a cylindrical surface that connects to the rim at the boundaries at angles of  $\pm 45$  degrees. For both examples, we can clearly see that by changing the functional, we obtain different shapes for the same input surface and constraints. Given Figures 8.8 and 8.9, a designer can glean an intuitive understanding of the functionals and can choose a functional to optimize when faced with a similar design task.

The second-order bending energy favors the emergence of spherical umbilical shapes. The unconstrained regions of the surface get deformed into spherical bulges that connect to the upper boundary constraints with a small region of high curvature. From the differences between the  $\text{SI-}F_1^2$  and  $\text{SI-}F_3^2$  optimal shapes (second row of Figure 8.8), it is apparent that the  $\text{SI-}F_3^2$  energy favors cylinders or toroidal regions more strongly than the  $\text{SI-}F_1^2$  energy.



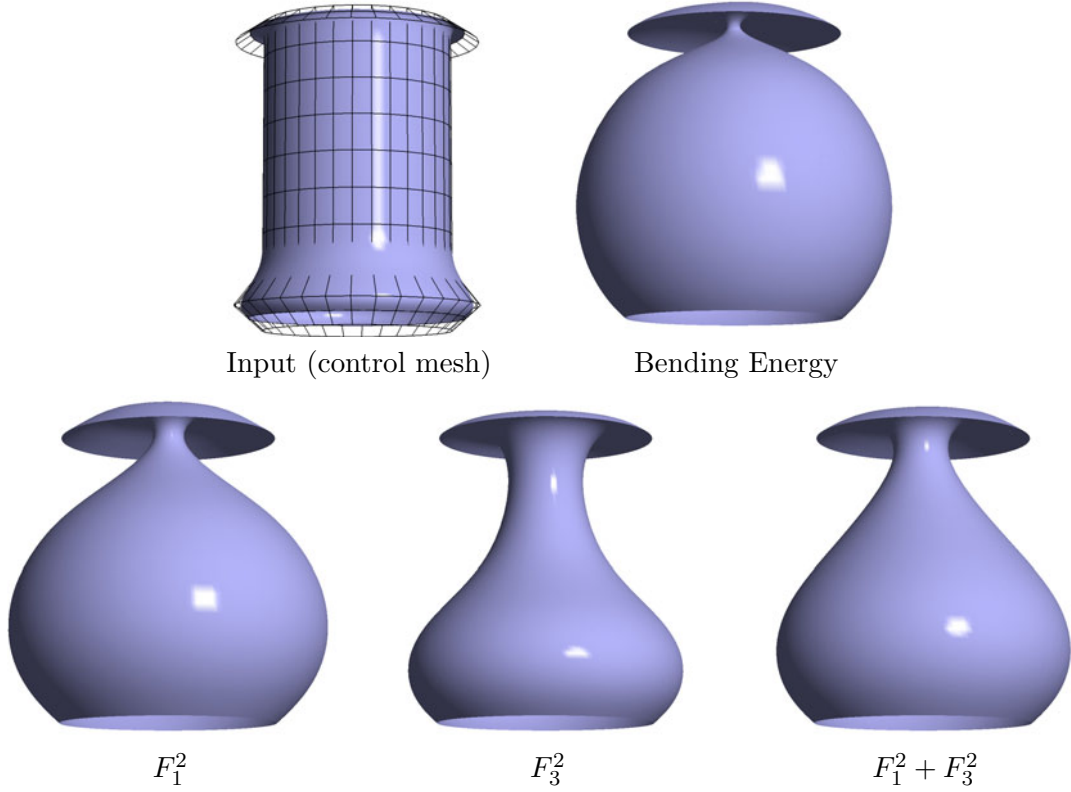


Figure 8.9. We specify a “lamp-shade”-like vase by setting 45-degree boundary constraints at the top and bottom of an open cylinder. The unconstrained regions of the bending energy optimal shape exhibits a dominant spherical bulge.  $\text{SI-}F_3^2$  energy minimization yields the closest approximation of a cylinder. The other optimizers ( $\text{SI-}F_1^2$  energy and  $\text{SI-Mehlum-Tarrou}$  energy) have shapes that are combinations of the bending energy and the  $\text{SI-}F_3^2$  energy optimal shapes.

## 8.6 Combining Second-Order and Third-Order Energies

As we explain in Section 4.3, we can formulate a mixture of second-order and third-order functionals by linearly combining the  $\text{SI-}F_1^2$ ,  $\text{SI-}F_3^2$ ,  $\text{SI-Mehlum-Tarrou}$  and bending energies. As one example, we pick the combination whose optimal genus-1 surface is a torus whose radii have a ratio of 0.5 (first row of Figure 8.10). The resulting optimal shapes have are a combination of those preferred by the third-order functional and the second-order functional. In particular, the optimal shapes in Figure 8.10 have thicker toroidal arms

$$\begin{array}{ccc} 0.85BE + 0.15F_1^2 & 0.85BE + 0.15F_3^2 & 0.92BE + 0.08(F_1^2 + F_3^2) \end{array}$$


---

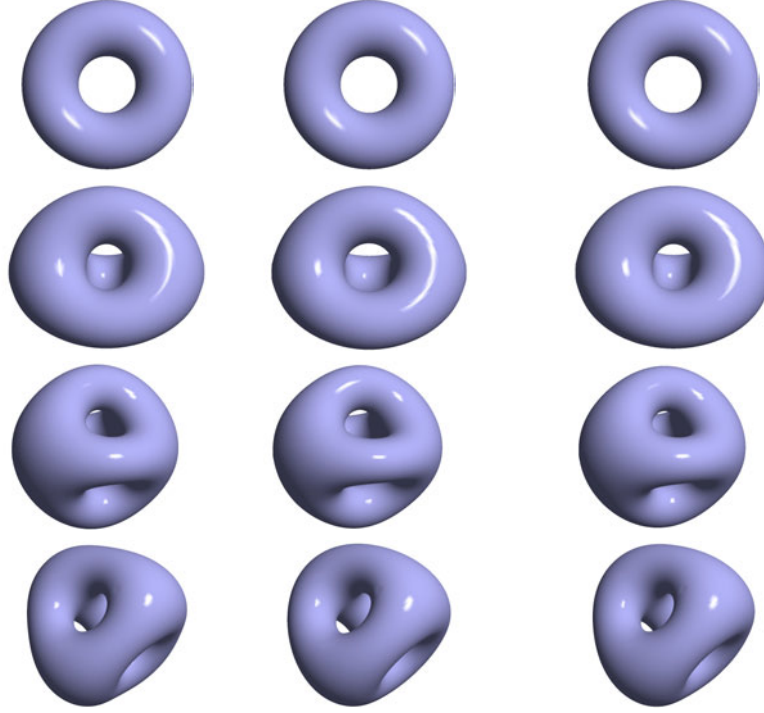


Figure 8.10. Combining second-order and third-order energies produces functionals with intermediate preferred shapes. In this example, we assign a higher weight to the second-order energy to construct a functional whose optimal genus-1 surface is the torus of radius ratio 0.5.

than the SI-Mehlum–Tarrou minimizers from Figure 8.5 and but are less “blobby” than the bending energy minimizers from Figure 8.5.

The combination of bending energy with either the  $SI-F_1^2$  or  $SI-F_3^2$  energy yields similar looking optimal shapes in Figure 8.10. This might be because the undulatory behavior that is a characteristic of  $SI-F_1^2$  energy minimizers is penalized by the second-order bending energy. Since the second-order energy is factored with a much higher weight than the third-order energy, the major difference between the  $SI-F_1^2$  and  $SI-F_3^2$  minimizers is neutralized by the bending energy, thereby yielding very similar looking minimizers for all the functionals in Figure 8.10.

As another test, we performed the optimization using combined functionals that spec-

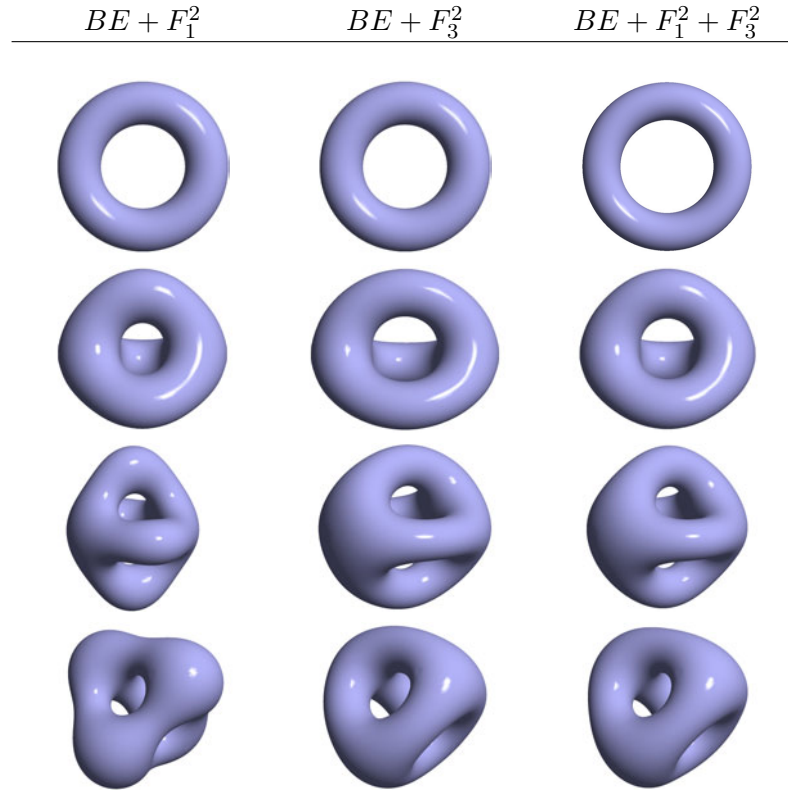


Figure 8.11. Combining second-order and third-order energies produces functionals with intermediate preferred shapes. In this example, we assign a equal weights to the energies to construct the combined functionals.

ify equal proportions of the second-order and third-order energies. The optimal shapes (in Figure 8.11) show some of the undulation that is a characteristic of the  $SI-F_1^2$  energy optimization. With equal weights, we see a difference in the optimizers of the combined functionals.

## Chapter 9

# Summary, Conclusions, and Future Work

### 9.1 Summary

We have explained how optimization can be used to produce a diverse set of aesthetically pleasing, smooth shapes. We showed how the local third-order surface behavior at a surface point can be intuitively expressed as a function of four parameterization-independent, purely geometric shape parameters. Our third-order functionals are formulated as functions of these intuitive shape parameters. By comparing the optimal shapes for some canonical input surfaces with respect to different functionals, we were able to compare the shapes preferred by the various functionals. Based on the results so far, we can draw the following conclusions.

## 9.2 Conclusions

1. We can use surface optimization as a shape design tool

With a better understanding of surface behavior, we can formulate a number of different functionals that produce a range of aesthetically pleasing optimal shapes in reasonable computation time.

2. Third-order functionals have desirable properties for shape design

The higher-order smoothness addressed by third-order functionals usually yields simpler shapes than those obtained by minimizing second-order functionals.

## 9.3 Future Work

As mentioned in Chapter 7, our optimization system is fast enough for study the shapes preferred by functionals, but not fast enough for interactive shape design. With interactive aesthetic design as our next goal, our future work will explore different methods to speed up the optimization process.

As the next step, we plan to implement the Sobolev gradient method (Section 7.2.1) developed by Renka and Neuberger [RN95]. For a curve optimization task, Renka and Neuberger show how the Sobolev gradients allow the optimization routine to find a minimum in drastically fewer iterations, albeit at an increased cost per iteration. For our surface optimization task, we want to investigate whether the Sobolev gradients significantly reduce the running time.

Another promising area of future work is to use high-level parameters to define the input shape. As mentioned in Section 7.1, by representing the shape as a blend of simple

primitives such as spheres, cylinders, and toroidal arms, we can greatly reduce the number of parameters that describe the surface. Consequently, the optimization routines will have a much smaller number of parameters, reducing the time taken to find the minimum.

However, it is not yet fully clear whether any such high-level surface representation will be able to represent the wide variety of shapes that need to be modeled with the same efficiency as our current representation by subdivision surfaces. In particular, we seek efficient methods to model sharp features and constraints. When, for instance, using implicit surfaces, the state-of-the-art method for modeling is the work of Turk and O'Brien [TO02]. Turk and O'Brien describe a system that constructs a smooth implicit surface that smoothly interpolates the given constraint points. An interesting challenge is to extend Turk and O'Brien's system so that it can efficiently handle sharp features and boundary constraints.

Once we can model a variety of shapes as a blend of implicit surfaces, the next challenge is to exactly compute the energy of the surface. Chopp and Sethian [CS93] describe a method for approximating the curvature of an implicit curve by using a finite difference discretization of the volume enclosed by the bounding box of the implicit curve. A finite difference discretization accurate enough for our purpose might be very fine, making the method computationally expensive. A challenging problem is to efficiently compute the exact energy of an implicit surface.

# Bibliography

- [BH77] M. Berry and J. Hannay. Umbilic points on Gaussian random surfaces. *Journal of Physics A: Mathematical and General*, 10(11):1809–1821, 1977.
- [Bir33] G. Birkhoff. *Aesthetic Measure*. Harvard University Press, 1933.
- [BK01] O. Bruno and L. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: Basic implementation, tests, and applications. *Journal of Computational Physics*, 169(1):80–110, 2001.
- [BLZ00] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *SIGGRAPH 2000: Proceedings of the 27th annual conference on computer graphics and interactive techniques*, pages 113–120, New York, NY, USA, 2000. ACM Press/Addison–Wesley Publishing Co.
- [BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 28–36, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [BMMS05] S. Benson, L. McInnes, J. Moré, and J. Sarich. TAO user manual (revision

- 1.8). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2005. <http://www.mcs.anl.gov/tao>.
- [Bra92] K. Brakke. The surface evolver. *Experimental Mathematics*, 1:141–165, 1992.
- [BS05] A. Bobenko and P. Schröder. Discrete Willmore flow. In *Eurographics symposium on geometry processing*, pages 101–110, 2005.
- [BW90] M. Bloor and M. Wilson. Using partial differential equations to generate free-form surfaces. *Computer-Aided Design*, 22(4):202–212, 1990.
- [CC78] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological surfaces. *Computer-Aided Design*, 10(6):350–355, 1978.
- [CCF91] M. Callahan, P. Concus, and R. Finn. Energy minimizing capillary surfaces for exotic containers. In *Computing optimal geometries*, pages 13–15, 1991.
- [CG91] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In *SIGGRAPH '91: Proceedings of the 18th annual conference on computer graphics and interactive techniques*, pages 257–266, New York, NY, USA, 1991. ACM Press.
- [CS93] D. Chopp and J. Sethian. Flow under curvature: Singularity formation, minimal surfaces, and geodesics. *Exp. Math.*, 2(4), 1993.
- [DFRS03] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855, 2003.
- [DKT98] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on computer*



*graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM.

- [DMSB99] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [EP97] J. Elliott and J. Peraire. Practical 3D aerodynamic design and optimization using unstructured meshes. *AIAA J.*, 35(9):1479–1485, 1997.
- [GGRZ06] E. Grinspun, Y. Gingold, J. Reisman, and D. Zorin. Computing discrete shape operators on general meshes. *Computer Graphics Forum*, 25(3), 2006.
- [GHDS03] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [Gra03] J. Gravesen. Third order invariants on surfaces. In *Proceedings of computational methods for algebraic spline surfaces (COMPASS)*, pages 193–211, 2003.
- [Gre94] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13:143–154, 1994.
- [Gri06] E. Grinspun. Discrete differential geometry: an applied introduction. In *ACM SIGGRAPH 2006 Courses*, (6), 2006.

- [GU01] J. Gravesen and M. Ungstrup. Constructing invariant fairness measures for surfaces. *Advances in Computational Mathematics*, 17(1):67–88, 2001.
- [HDD<sup>+</sup>94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH '94: Proceedings of the 21st annual conference on computer graphics and interactive techniques*, pages 295–302, New York, NY, USA, 1994. ACM.
- [HKS92] L. Hsu, R. Kusner, and J. Sullivan. Minimizing the squared mean curvature integral for surfaces in space forms. *Experimental Mathematics*, 1:191–207, 1992.
- [Jos07] P. Joshi. Energy minimizers for curvature-based surface functionals. Master’s thesis, EECS Department, University of California, Berkeley, May 2007.
- [JS07] P. Joshi and C. Séquin. Energy minimizers for curvature-based surface functionals. *Computer-Aided Design and Applications*, 4(5):607–617, 2007.
- [KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM.
- [Kje83] J. Kjellander. Smoothing of bicubic parametric surfaces. *Computer-Aided Design*, 15, 1983.
- [Lev06] Adi Levin. Modified subdivision surfaces with continuous curvature. In *SIGGRAPH 2006: ACM SIGGRAPH 2006 Papers*, pages 1035–1040, New York, NY, USA, 2006. ACM.

- [Mor93] H. Moreton. *Minimum Curvature Variation Curves, Networks, and Surfaces for Fair Free-Form Shape Design*. PhD thesis, EECS Department, University of California, Berkeley, March 1993.
- [MS92] H. Moreton and C. Séquin. Functional optimization for fair surface design. In *SIGGRAPH*, pages 167–176, 1992.
- [MT98] E. Mehlum and C. Tarrou. Invariant smoothness measures for surfaces. *Advances in Computational Mathematics*, 8(1):49–63, 1998.
- [MWP96] T. Maekawa, F. Wolter, and N. Patrikalakis. Umbilics and lines of curvature for shape interrogation. *Computer-Aided Geometric Design*, 13(2):133–161, 1996.
- [Oss02] R. Osserman. *A Survey of Minimal Surfaces*. Courier Dover Publications, 2002.
- [PFTV92] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 2nd Edition, 1992.
- [Por01] I. Porteous. *Geometric Differentiation*. Cambridge University Press, 2nd Edition, 2001.
- [Rad30] T. Rado. On Plateau’s problem. *The Annals of Mathematics, Second Series*, 31(3):457–469, July 1930.
- [Ren04] R. Renka. Constructing fair curves and surfaces with a sobolev gradient method. *Computer-Aided Geometric Design*, 21(2):137–149, 2004.
- [RN95] R. Renka and J. Neuberger. Minimal surfaces and Sobolev gradients. *SIAM J. Sci. Comput.*, 16(6):1412–1427, 1995.

- [RR91] T. Rando and J. Roulier. Designing faired parametric surfaces. *Computer-Aided Design*, 23:492–497, 1991.
- [Rus04] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D data processing, visualization, and transmission*, 2004.
- [SBL91] U. Seifert, K. Berndl, , and R. Lipowsky. Shape transformations of vesicles: Phase diagram for spontaneous-curvature and bilayer-coupling models. *Phys. Rev.*, A(44):1182–1202, 1991.
- [SCM95] C. Séquin, P. Chang, and H. Moreton. Scale-invariant functionals for smooth curves and surfaces. In *Geometric Modelling, Dagstuhl, Germany, 1993*, pages 303–321, London, UK, 1995. Springer-Verlag.
- [SK01] R. Schneider and L. Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer-Aided Geometric Design*, 18(4):359–379, 2001.
- [Sta98] J. Stam. Exact evaluation of Catmull–Clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH ’98: Proceedings of the 25th annual conference on computer graphics and interactive techniques*, pages 395–404, New York, NY, USA, 1998. ACM Press.
- [TO02] G. Turk and J. O’Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, October 2002.
- [WB01] K. Watanabe and A. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3), 2001.
- [Wei85] K. Weiler. Edge-based data structures for solid modeling in curved-surface environment. *IEEE Computer Graphics and Applications*, 5(1):21–40, 1985.

- [Wil71] T. Willmore. Mean curvature of Riemannian immersions. *J. London Math. Society*, 11:307–310, 1971.
- [WW92] W. Welch and A. Witkin. Variational surface modeling. In *SIGGRAPH '92: Proceedings of the 19th annual conference on computer graphics and interactive techniques*, pages 157–166, New York, NY, USA, 1992. ACM Press.
- [XPB06] G. Xu, Q. Pan, and C. Bajaj. Discrete surface modelling using partial differential equations. *Computer-Aided Geometric Design*, 23(2):125–145, 2006.
- [XZ07] G. Xu and Q. Zhang. G2 surface modeling using minimal mean-curvature-variation flow. *Computer-Aided Design*, 39(5):342–351, 2007.
- [YZ04] L. Ying and D. Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. In *SIGGRAPH 2004: ACM SIGGRAPH 2004 Papers*, pages 271–275, New York, NY, USA, 2004. ACM Press.
- [Zor06] Denis Zorin. Constructing curvature-continuous surfaces by blending. In *SGP 2006: Proceedings of the fourth Eurographics symposium on geometry processing*, pages 31–40, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.