Serious Putty: Topological Design for Variational Curves and Surfaces

William Welch

December 1, 1995

CMU-CS-95-217

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Andrew Witkin, Chair
Paul Heckbert
Fritz Prinz
Tony DeRose, University of Washington

This research was supported by a fellowship from the Engineering Design Research Center at Carnegie Mellon University, and by an equipment grant from Silicon Graphics, Inc. The views and conclusions contained in this document are those of the author and should not be interpreted as representing anyone's official policies, either expressed or implied.

Keywords: Curves and Surfaces, Shape Optimization, Topology, Computer Aided Design

Abstract

In this work we develop a new approach to designing curves and free-form surfaces on a computer. It is inspired by a style of pencil-and-paper design used for sculptured surfaces, in which the designer specifies the shapes of important curves (*character lines*) and indicates surfaces that pass through them smoothly, with no unnecessary bulges or wiggles (that is, the surfaces are *fair*). Unlike previous modeling approaches based on the notion of character lines, this approach allows surfaces to be cut apart and smoothly joined along arbitrary curves, so that the designer can build up complex shapes and topologies from simpler ones. Further, the surfaces are infinitely stretchy, so that the designer may add unlimited amounts of detail simply by indicating more control points and curves. Finally, portions of the surface may be made to copy externally controlled *shape tools*. This allows the designer to mix free-form and structured shapes within a single composite surface model of arbitrary topology.

This kind of conceptually simple shape description ("give me a fair surface bordered by *these curves* that passes through *those curves* while touching *that point*") may be precisely interpreted as a functional minimization problem in the calculus of variations ("give me the surface coordinate function that maximizes *this fairness integral* subject to *those geometric constraints*"). The modeler described here represents curves and surfaces implicitly, as the solutions of such variational minimization problems. As the designer interacts directly with a surface, the modeler interprets these actions as changing the variational shape specification. Triangulated point sets are used to approximate these smooth variational surfaces in real time, using a novel finite-difference scheme over arbitrary-topology surface meshes along with an adaptive, interactive mesh refinement and re-triangulation scheme.

Ultimately, all of these numerical details are hidden from the designer, who sees a pristine surface that may be grabbed at arbitrary points and along arbitrary curves, and whose shape changes in simple, predictable ways. The resulting ability to design variational shapes of arbitrary, mutable topology has never before been available in an interactive geometric modeler.

Acknowledgements

It has been more than a year since this work was defended at CMU. With the writing at last completed, I'd like to make a few acknowledgements and send out a round of thanks. It will be easy to resist the urge to write something clever, and to shrug off the guilt at not groping towards something profound. That would add another day's writing. And noone (well, noone I'm thanking here) would want that.

So let me begin by thanking my parents, who have always been there for me, and are as pleased as anyone to see me wrap this up and get a real job.

Thanks to Don Trotter and Dan Embree, of Mississippi State University's Electrical Engineering and English departments. Both have had an enormous effect on me through my undergraduate education.

Thanks to Carnegie Mellon University's School of Computer Science — the whole outfit: the deans, the assistant deans, the associate deans, the facilities folk, the janitors, the professors, the students, and the robots. CMU has been a dynamite place to be a grad student. Most importantly, the School gave me the freedom to try a few things, realize I wasn't made for honest work, and take up modeling and computer graphics.

Beaucoup thanks to Andy Witkin, my advisor, for inciting and supporting that decision, teaching me the math, showing me how to do research, and keeping me on-track through the whole process with a sense of humor. They say that choosing an advisor is the most important decision you make in graduate school. There's one I got right.

Thanks to the rest of my dissertation committee — Paul Heckbert, Tony DeRose, and Fritz Prinz — for their advice and assistance with this work.

A big thanks to my friends Dave and Gretchen Kosbie, Michael Gleicher, and Bryan Loyall, who have all joined me at one time or another in the behavioral extremes that accompany any Ph.D. program.

Thanks to Ajay Jain for providing ample time at Arris to finish writing this document.

And finally, thanks to Holly Schneider, for everything.

Contents

Chapter 1

Introduction

In this work we develop a new approach to designing curves and free-form surfaces on a computer. Traditionally, free-form surface design tools have handled only simple surface topologies, such as panels for auto bodies. But emerging manufacturing technologies, as well as increasing demand for richer models in computer graphics and animation, motivates us to consider more ambitious approaches to free-form surface design, including support for arbitrary topologies and higher-level, structured shape control.

The approach developed here addresses these issues by mimicking a style of pencil-and-paper design often used for sculptured surfaces, in which the designer sketches important curves (*character lines*) and indicates that a "nice" surface should pass through them. This work goes beyond pencil-and-paper sketching by bringing such descriptions to life in a 3D surface modeler: a designer will control surface shape by tracing out character lines on a computational surface and then re-shaping them, carrying the surface along. Unlike previous modeling approaches based on this notion of character lines, we allow surfaces to be cut apart and smoothly joined along arbitrary curves, so the designer can build complex shapes and topologies from simpler ones. Further, these surfaces are infinitely stretchy, so that the designer may add unlimited amounts of detail simply by indicating more control points and curves. Finally, portions of the surface may be made to copy externally controlled *shape tools*. This allows the designer to mix free-form and structured shapes within a single composite surface model.

Our approach is based on *variational shape design*, in which shapes are specified implicitly as the solutions to optimization problems. Though the necessary mathematics have been known since the 18th century, it is only within the past several years that computers have become fast enough to allow variational techniques to be used for interactive shape design. An important advantage of the approach is that shapes created by the designer can be viewed as templates for entire families of related shapes, parameterized by their character lines. The ability to create variational shapes of arbitrary and changeable topology, with user-specified sets of control points and curves, has never before been available in an interactive geometric modeler.

1.1 Definitions

Before beginning a description of the approach, a quick discussion of some terminology is in order. This work is concerned with the design of *free-form surfaces* — generally smooth, curved surfaces that may contain occasional creases or corners. At any given point on a surface, it may be curved in any and all local directions (that is, the surface is *doubly curved*). These are sometimes referred to as *sculptured* surfaces in the design literature.

Within this broad class of free-form surfaces, this work is most often concerned with what are known as *fair* shapes — surface (and curve) shapes that don't have unnecessary bulges or wiggles. Fairness is a necessarily fuzzy notion, based as it is on human perception of shape quality; but it is intimately related to the distribution of curvature over the shape [Mor93]. The surface of an inflated balloon would be considered fair, whereas its deflated counterpart would likely not be.

A curve or surface's *topology* is an abstracted version of its geometric structure: the relationships that don't change when its shape is smoothly deformed. For surfaces, this includes global notions such as open and closedness, handles and holes (a cylinder is open, and has two holes and no handles; a torus is closed, and has one handle and no holes). It also includes containment and connectedness relationships between subsets of the object. Informally, if one considers a shirt crumpled on the floor as an abstract surface, the topological information includes permanent features like openings for the arms and head, and the seams that join the sleeves to the body. The seams and holes carve it up into *regions*, giving containment relationships and boundary curves, as well as connectivity information between these *topological features*. None of this changes when you put the shirt on.

1.2 Motivation

Computer Aided Geometric Design (CAGD), a field whose principal concern is describing curves and surfaces using computers, has been a rich area of research and practical application ever since hardware for the automated machining of 3D shapes first became available in the 50's [Far90]. It is a place "where the math meets the metal." That said, it should be pointed out that since its beginnings, which are rooted in the development of design tools for the automotive industry, the majority of free-form surface design methods and tools have addressed simple, planar topologies — as might be used for designing stamped panels, or the separate pieces of injection molded surfaces. Computationally, dealing with arbitrary surface topologies is a *much* more difficult proposition than dealing with planar patches — enough so that issues have only recently begun to be addressed.

The attention to such issues is particularly timely, given recent advances in manufacturing technologies for rapid prototyping. It is now possible to directly fabricate free-form geometries of essentially arbitrary topology, *e.g.*, by metal deposition [MPR+94]. As of this writing, it is possible to render in metal a wide range of surface geometries for which there are no suitable design tools! Current design tools used with these processes are based on solid modeling techniques (discussed below), and do not address the problem of direct design for free-form surfaces. More general surface design tools will ultimately allow industrial designers (who still fall back to clay models on occasion) to exploit this new fabrication capability for designing and prototyping aesthetic shapes via computer. Aside from physical realization through manufacture, such geometries are also desirable as "virtual artifacts." As computer graphics and animation progresses, so does its appetite for richer geometric models, and it is desirable to consider more ambitious design tools.

1.3 Current approaches (and their limitations)

Direct-surface modelers typically represent free-form surfaces as piecewise smooth quilt-works of surface patches [Far90]. The designer is given a mesh of control points or curves whose number and arrangement is determined by fineness of the underlying surface representation. Thus, the designer exercises local control over shape, by re-shaping elements of the control net. The drawback is that making a conceptually simple change to shape using such a control net could require moving every single element in a coordinated way, if the geometry of the change does not fit well with the given mesh structure. As if this weren't bad enough (and in fairness, recent research has aimed at better, nonlocal control of such quilt-works [CG91, WW92, Kal93]), the mathematical form of many of these patches necessarily restricts the global surface topology. Piecewise smooth patch schemes based on rectangular control nets are obviously restricted to sheets, cylinders, and tori as their only natural model topologies. Many older approaches based on smooth triangular elements are similarly limited, though the reason is more subtle (Section 2.4). Topologically general piecewise smooth representation schemes are generally more complex constructions[MLL+92], and remain an active area of research[BGW88, LD90, HKD93, Loo94].

More ambitious topological modeling for manufacture has been the purview of Constructive Solid Geometry (CSG) approaches[RV82]. Here, complex composite shapes are built up by combining simple solid primitives (*e.g.*, prisms, spheres) using Boolean set-theoretic operations (*e.g.*, union and difference). Although unrestricted surface topologies can be designed this way, there is no direct, explicit control over surface shape; rather, it is the result of simple combinations of rigid solid primitives. Thus, even though CSG systems have been extended to handle primitives with sculptured faces[AMR83, CGP93], these solid-based methods do not address design situations where creating an overall graceful surface shape is a goal.

Another approach to topologically unrestricted design is direct volume sculpting, where local changes to the surface may be made by chipping away or adding "material" to a 3D neighborhood [GH91]. The drawback again is that global changes in shape must be formulated as sequences of local operations that touch potentially every point on the surface.

Finally, the character line approach to design described in opening paragraphs is an example of *variational* shape design, about which we will say a good deal more, below. As will be discussed in Chapter 2, there has been a fair amount of CAGD research (much of it in the last 5 years) into variational shape design. The work in surface design falls into two broad categories: schemes that generate high quality surfaces of perhaps arbitrary topology, but do not run at interactive speeds; and schemes that run at interactive speeds but cannot generally produce high-quality shapes, and generally do not accommodate arbitrary topologies.

1.4 Variational shapes

This work focuses on a class of free-form shapes that are optimal in a certain sense. In designing free-form curves and surfaces, one often wants them to take on fair shapes within a region; or one may want portions of a curve or surface to copy some other prototype shape as nearly as possible. Each of these notions can be expressed as an optimization problem: find a shape that maximizes fairness, or minimizes deviation from a given shape. Many shapes from nature can also be described in terms of optimization (soap films minimize their surface area; loaded elastic beams minimize their total curvature). A shape that optimizes some given quality measure is often referred to as a *variational* shape in the CAGD literature[HS90], because it is best described mathematically using the calculus of variations[CH37]. As will be shown, though the mathematics behind such shapes can be intricate, their high-level descriptions are remarkably simple ("give me a fair cylinder with end curves shaped like *this*, passing through *that* point"); and this yields a powerful, concise vocabulary for describing curves and surfaces. A nice side-effect of this kind of shape specification is that, instead of static geometry, one has actually specified an entire family of similar shapes, parameterized by the specified control curves.

In later chapters, the reader will manage to escape with only a very small dose of variational calculus (just enough for approximating solutions to the posed shape optimization problems). Nonetheless, this technical term "variational" occurs fairly often in the discussions below, for no better reason than that we need a convenient name for this class of shapes. One may safely ignore its technical meaning, and mentally substitute "optimal" or "high quality" if preferred.

1.5 Goals

In this work we will address the problem of specifying and representing free-form shapes variationally. Some specific goals are:

- Model variational surfaces of unrestricted topology. Surfaces may be open or closed, and have any arrangement of holes and handles.

- Allow the designer to precisely control portions of the surface shape by specifying control points or curves through which the surface must pass.

- Allow the designer to stretch, shrink, or deform surface areas without restriction, and add arbitrary amounts of detail.

- Allow the designer to interactively and incrementally build up smooth topologies through surface surgery: cutting, stitching, creasing, and skinning along embedded control curves.

- Since not all shapes are best described variationally, allow the designer to incorporate explicit shapes into a variational model and thus create structured composites.

1.6 Approach

The surface behavior outlined in these goals can be precisely characterized in terms of a simply formulated (if not simply solved) variational optimization problem. User-defined control points and curves act as geometric constraints on the possible shapes; the automatic fairing and shape-copying behaviors are then realized by optimizing the shapes of user-specified topology subject to these geometric constraints. In this work, a measure of the curve or surface's total curvature will be minimized as a way of making it seek a fair shape.

Unfortunately, it is not generally possible to solve such variational problems explicitly. We therefore develop a method for approximating solutions to these curvature-minimization problems using a triangulated surface mesh. This involves setting up a generalized finite-difference scheme and optimizing estimated curvatures over the mesh (a process that is complicated by the fact that there will be no global parameter plane over which to formulate the computation). A dynamic adaptive mesh scheme is used to maintain a good sampling and triangulation of the surface as its shape changes.

Ultimately, this approximation machinery will be treated as a "black box". This allows us to build an interactive modeler that operates on variational curves and surfaces as its basic shape representation, much like a conventional modeler might operate on B-splines or Bezier patches. User interactions will be interpreted as modifying the variational specification for the shape, and the modeler will be relied upon to recompute fresh approximations to the new shape at interactive speeds.

1.7 Contributions

The primary contributions of this work are:

- It is the first work to address interactive, incremental design of free-form surfaces of arbitrary topology where the user has explicit control over the topology at all times.
- It is the first interactive approach to free-form surfaces allowing the user to "slide" surface features around relative to each other.
- It includes an approximation scheme for geometric thin plate surfaces, based on triangulated surface meshes, that is speedy and robust. Some secondary results related to this scheme include:
  - an approach to computing neighborhood parameterizations for a finite-difference scheme over an arbitrary topology mesh
  - an adaptive mesh generation scheme suitable for use in interactive modeling for arbitrary-topology surfaces.

As part of this work, we built an interactive, direct manipulation surface modeler that demonstrates all of the functionality discussed above (though no one would mistake it for a full-fledged industrial design tool). The modeler itself represents a number of secondary contributions:

- It is the first modeler that uses variational curve and surface specifications as its basic shape representation (built on top of the approximation machinery above).

- It is the first *variational* modeler that allows a designer to build up surface topology in terms of "surgical operations" on the 3D surface. Traditional alternatives are to indicate edge correspondences on a single polygonal (2D) domain (*e.g.*, [FRC92]), or in 3D to connect-the-dots and fill-the-holes to build up a surface control net (*e.g.*, [CK83]). Neither of these protects the designer from specifying nonsensical surface topologies.

1.8 Road-map to the thesis

We begin with a discussion of a wide range of existing curve and surface modeling techniques (Chapter 2). The sections of this chapter are relatively self-contained, and the reader may prefer to refer back to them when directed from later chapters. An overview of our approach to shape design describes a sample design session (Chapter 3). A detailed discussion of the approach begins with methods of specifying and representing variational curves and surfaces (Chapter 4). This is followed by a method for approximating their shapes using triangulated meshes, based on a generalized finite-difference scheme (Chapter 5). In addition to serving as an approximate shape representation, the triangulated surface also serves as a computational mesh for these approximation calculations. In Chapter 6 we present a scheme for maintaining a quality mesh as curve and surface shapes evolve. Finally, our implementation of a modeler that uses these variational curves and surfaces as its basic representation for shape is discussed, along with additional implementation details for the machinery presented in the previous chapters (Chapter 7). The dissertation concludes with a summary of the contributions, and discussion of possible extensions to the work.

Chapter 2

Previous Work

In this chapter, we survey previous approaches to computer-aided design of free-form shapes, and review the mathematical topics and techniques that play a role in our modeling approach. There is a wide variety of free-form surface modeling approaches in the literature. None of them manages to deliver globally fair, arbitrary surface topologies at interactive speeds, but they fall short in a number of very different ways. Various mathematical/computational elements that go into a free-form surface modeler will be considered, leading up to a discussion of interactive modelers in Section 2.6. This is followed by a discussion of computational techniques for triangulated surface meshes used as approximations to smooth surfaces.

2.1 Overview

[**Topology and design**] The chapter begins with a review of some concepts and terminology from topology relevant to surface modeling (Section 2.2). A fundamental aspect of our approach to modeling curves and surfaces is the separate handling of topological and positional information for a given model. Some approaches to topological specification in computer-aided design are taken up in Section 2.3. A topological description of a surface captures the ways in which various regions of a surface are connected to or contained within each other — relationships that do not change when the surface is smoothly deformed. Because a given abstract surface (*e.g.*, the topological sphere S^2) will have many possible realizations as a surface in 3D (*e.g.*, a pear, a head, a dining-room table), most of these modeling approaches also specify shape along with topology.

[**Fair surfaces**] By far, the most common methods of representing topologically interesting free-form surfaces is as collections of simpler (typically rectangular or triangular) *patches*, joined along their edges to form a piecewise smooth *composite* surface. Methods for computing the shapes of such piecewise smooth elements can be usefully classified as *local* or *global*. Local methods construct a surface one patch at a time, using geometric constructions that take information such as edge position and tangency requirements along a patch boundary and conceptually work inward to compute the shape of the patch's interior. Global schemes, which operate simultaneously on collections of patches, most often optimize some global shape objective or attempt to satisfy global area or volume constraints. In this work, we are interested in generating globally *fair* surfaces — "nice" surfaces having no extraneous wiggles. This is a bit more restrictive than simply asking for a *smooth* surface; as an example, an egg-crate mattress's surface is smooth but not fair. As will be seen, purely local schemes aren't capable of generating high-quality faired surfaces, and this will lead us to ultimately consider only global schemes.

[**Shape optimization**] Precise mathematical definitions of "fairness" will be discussed. A number of different characterizations have been used previously, each having strengths and weaknesses. Given a particular characterization of fairness, it is possible to cast surface shape specification as an optimization problem. Using the calculus of variations, the desired shape can be described as the solution to a variational minimization problem with the fairness measure as the objective function. Discussion will cover some of the existing variational shape design approaches, their ways of assessing a shape's fairness, and (because such variational problems are almost never solvable in closed form) their solution approximation methods.

[**Interactivity**] Another important goal of our modeling approach is *interactivity.* Only a few of the global approaches discussed are appropriate for modeling at interactive speeds, recomputing curve or surface shapes quickly as the designer changes the specification. Section 2.6 discusses the current range of interactive variational modelers, and some of the issues involved in supporting interactivity. In order to keep their computations tractable, interactive systems generally restrict the kinds of constraints and objectives that can be used in describing a surface, and unfortunately this restricts the kinds of shapes and topologies that may be effectively modeled. Overcoming these limitations will motivate many of the representational and algorithmic choices in the following chapters.

[**Triangulated surfaces**] Finally, the approach to be developed uses triangulated surfaces. In support of the actual shape optimization computations of our approach, it will be important to compute over mesh vertices as if they were discrete samples of some underlying smooth surface. It will also be necessary to maintain good triangulations (serving as computational meshes) over surfaces. The last section of this chapter reviews some work in mesh generation for curved surfaces. Mesh generation for surfaces in 3D is a much more difficult problem than mesh generation in the plane, because of the difficulties in constructing surface triangulations over points in space.

2.2 Topology

In this work we are concerned with smooth curves and surfaces. We begin with a brief review of topological concepts, specialized to these objects. Viewing an abstract surface as a set of points, the set's *topology* defines connectivity and containment relationships among subsets of its points. Locally (that is, within a small neighborhood of a point on a curve or surface), object topology is always the same: curve neighborhoods have the 1-dimensional topology of an open interval (or half-interval, for boundary neighborhoods), and surface neighborhoods have the 2-dimensional topology of an open disc (or half-disc on the boundary). It is global topological information, rather than local, that interests us here: continuity and containment relationships among subsets, handles and holes in a surface, and the surface's *genus*. *Holes* are openings in a surface, bounded by closed

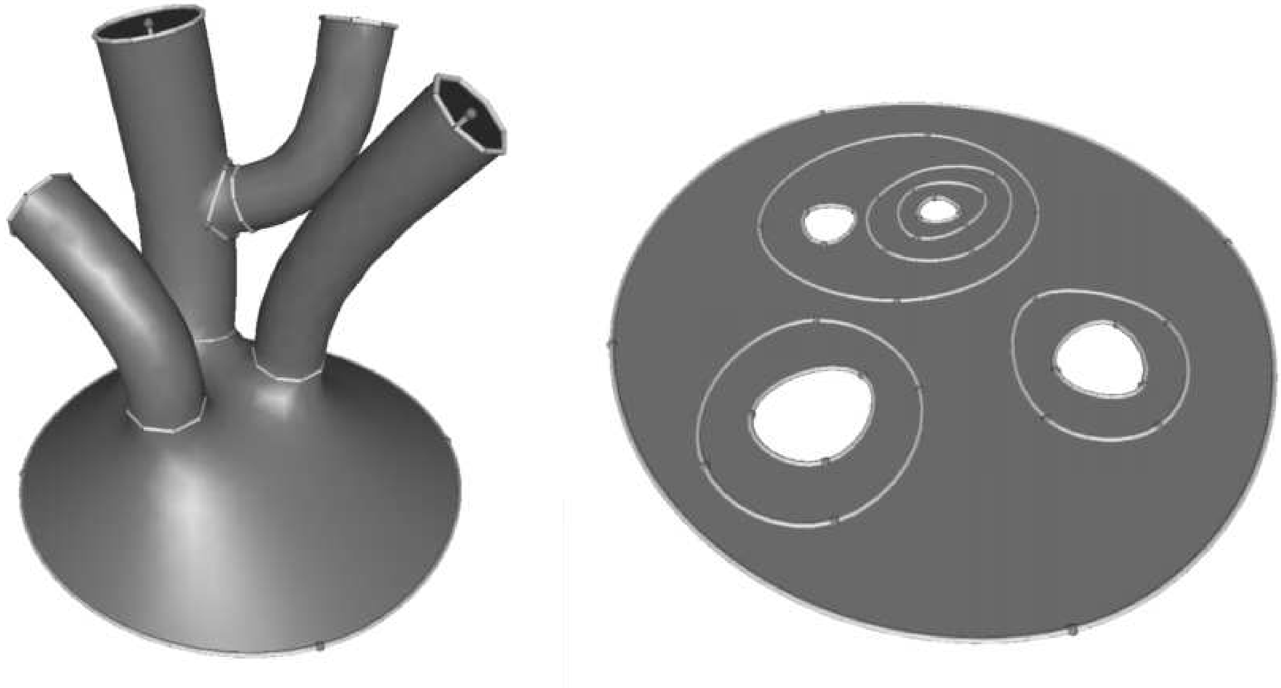

Figure 2.1: (L) Party Hat: a branching surface. (R) Party Flat: the topological regions associated with the surface on the left.

curves: topologically, a cylinder is a sphere with two holes cut in it. *Genus* is a count of the number of *handles* on a surface: a sphere is genus 0, a coffee-cup or donut is genus 1 (though it is probably too late to convince the world to speak of donut handles), and a pair of scissors has a genus 2 surface. Such topological information is largely independent of the particular shape a surface may have. Figure 2.1 shows a branching object and the topological information distilled from it.

The standard approach to representing general topological spaces builds them up as combinations of the simplest possible spaces. These simpler spaces (0D points, 1D intervals, 2D triangles, 3D tetrahedra, etc) are called *simplices*, and spaces built up from them according to a few simple rules are called *simplicial complexes*. General simplicial complexes can represent manifold topologies (curves and surfaces), and non-manifold topologies (Figure 2.2). A formal discussion of the rules that govern the construction of simplicial complexes would therefore be more precise and more general than is warranted here (see Jänich[Jän84] for a lively presentation of point-set topology leading to simplicial complexes and topological "gluing" operations, or Massey[Mas77] for a more comprehensive treatment). Informally, simplices may be glued together by having individual elements share boundary elements. Thus, 1D curve domains are built up by pairing endpoints of disjoint intervals, and 2D surface domains are built by pairing edges of disjoint triangular faces. Note that it is connectivity among simple topological elements that is being recorded here, not shape. Even though our illustrations render surface complexes as triangulated surfaces situated in 3D, topological reasoning over a complex is a combinatorial process and makes no use of such

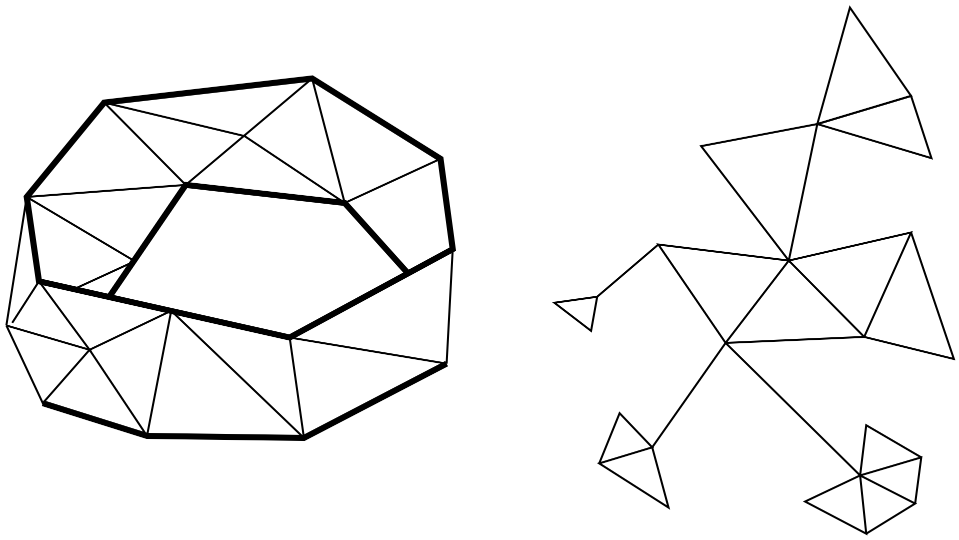

Figure 2.2: Two simplicial complexes. The one on the left represents a cylindrical surface strip. The one on the right, well.... A triangulated surface along with its nodes and edges constitutes a valid simplicial complex, but the converse is not necessarily true.

spatial coordinates. As an example, the Euler characteristic χ is a fundamental algebraic-topological quantity, and is given by F-E+V, the alternating sum of the number of faces, edges, and vertices. The Euler characteristic is related to the genus g of a surface by $\chi = 2(1-g)$. It is invariant for a given topological surface: the genus of the surface doesn't depend on the way it is triangulated.

To take an abstract surface and associate 3D coordinates with its points — as is done when rendering surface meshes as triangulations in 3D — is to create an *immersion* of the abstract surface. If that immersion is such that the 3D surface does not intersect itself, it qualifies as an *embedding* of the surface.

In modeling applications where a surface is represented as a piecewise smooth mesh of patches, it is trivial to interpret such a mesh as a simplicial complex, or more general *cell complex* when the patches are not triangles. Having an explicit topology representation leaves open the possibility of computing over and formally reasoning about model topologies [Män83, DC91, DE93, PBCF93].

2.3 Topological Modeling

This section considers approaches to surface modeling that admit variable topologies. Some of the approaches will represent topological information implicitly — it depends on and must be derived from other geometric information. Other approaches will build on explicit topological representations that may be directly modified. Of course, none of these approaches is concerned exclusively with topological specification, but also with specifying a particular shape along with the given topology. Some of these shape-related issues will be taken up in more depth in subsequent sections.

Figure 2.3: A molecular surface represented as a contour of a sum of Gaussian functions. A single bias parameter controls the "blobbiness" of the shape, and the topology may change as a side-effect of changing this parameter.

2.3.1 Algebraic curves and surfaces

One approach to representing free-form surfaces of arbitrary genus is as contours of algebraic functions defined over $\mathcal{R}^3$ that is, the set of points

$$\mathbf{x} \in \mathcal{R}^3 \;|F(\mathbf{x}) = 0. \tag{2.1}$$

for some scalar function F. As an example, a sphere of radius r centered at point $\mathbf{p}$ could be implicitly defined as the points satisfying $(\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p}) - r^2 = 0$. This is a broad class of functions, and has seen a variety of uses in modeling and animation[Bli82, WH94] (Figure 2.3). A benefit of working with such a representation is that it is computationally inexpensive to decide whether a given point in space $\mathbf{x}$ is inside, outside, or on the surface (just evaluate the function and look at the sign of the result). The chief difficulty with such a representation is that it is not easy to operate directly on the surface itself, *e.g.*, to render it, or measure its shape properties within some region, or determine its topology. Such operations need explicit representations of surfaces as mappings from some 2D domain into 3D (*parametric* surfaces, discussed next). Algorithms for recovering explicit surfaces from an implicit definition involve contour tracing (*e.g.*, the popular "marching cubes" algorithm[Blo94]), or point-sampling [dFGTV92, WH94] when a triangulation of the surface is not required.

Although it is trivial to modify functions and their parameters to generate a wide variety of surfaces, doing so in a controlled way, so that the resulting surface satisfies given geometric constraints, is more difficult. This kind of inverse-control for implicit curves and surfaces has been accomplished through algebraic function fitting[Pra87, Baj92], and direct construction of implicit blending surfaces [MS85, HH87, Roc89]. Many of these methods construct blend surfaces of simple, fixed topology using restricted sets of basis functions. Though this does not allow the range of topologies possible with general implicit functions, one does not generally want interesting topological behavior from a rounding or fillet operation.

More general inverse control with arbitrary constraint points is also possible with direct fitting techniques [Pra87] or iterative relaxation of the parameters of a given set of shape

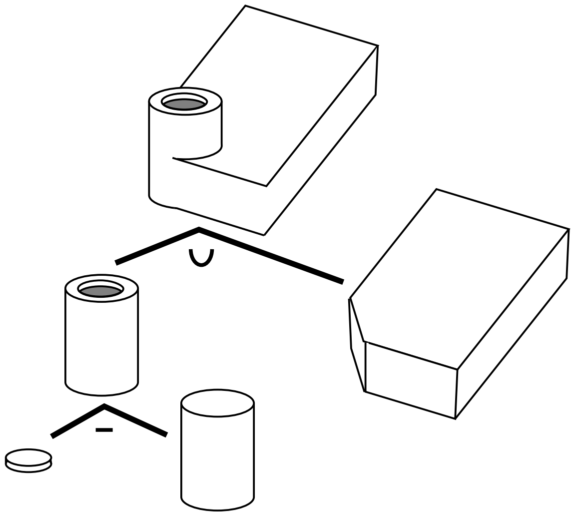

Figure 2.4: A solid model and its associated CSG tree of operations. The smaller cylinder is subtracted from the larger to make a socket in the top. The socketed cylinder is then unioned with the slab.

functions[WH94]. When a rich enough set of basis functions is used, the actual topology of the fitted algebraic surface may begin to depend on the particular values of parameters. For example, a surface generated by a sum of Gaussians [WH94] may vary between multiple disconnected regions and a single blob, or have holes and concavities depending on the settings of the Gaussians' origin, bias, and standard deviation parameters (Figure 2.3). Though it is possible to determine the topology of the fitted surface through contour tracing (though only within a given level of detail), there seems to be no general way to turn this around and constrain the topology of such surfaces to remain fixed, *e.g.*, by placing constraints on the allowable values of the parameters.

2.3.2 Particle systems

Szeliski, *et al.*[ST92, STT93] present a modeling approach that uses localized particle interactions to make the particles behave as if they were being attracted to some underlying surface. The result is evocative of a point-sampled implicit surface, but without an actual implicit function contour defining the surface. Because there is no fixed connectivity between particles, fluid "topological" changes occur when portions of the "surface" come into contact and their particles merge. The limitation for topological design is ultimately the same as with implicit surfaces: one wants to prescribe topology, and change it in controlled ways. This cannot be done using such a particle system because there is no underlying surface topology. It is not sufficient to merely impose and maintain a surface triangulation over the particles (this in itself is difficult, as the discussion in Section 2.8 will show), since the particles' interactions do not take such a triangulation into account. More will be said about particle systems and associated triangulations in the mesh generation discussions of Section 2.8.

2.3.3 Solid modeling and B-reps

Possibly the most widely-used approach to designing topologically interesting surface models is actually a method of specifying 3D solids: Constructive Solid Geometry (CSG)[RV82, Req80]. Briefly, a 3D *solid* is specified as the result of applying Boolean set operations (union, intersect, complement) to solid primitives. Figure 2.4 shows an example of such a construction. A solid model is represented implicitly as a tree of Boolean operations applied to a set of primitive solids. One popular way of implementing CSG schemes is in terms of an explicitly represented boundary surface, referred to as a *B-rep* solid[Req80]. These boundary representations comprise 2D faces joined by networks of boundary curves. Taking only the connectivity information in this graph of nodes, edges, and faces yields an abstract topological surface specified as a cell complex. In its purest form, a designer never directly interacts with these topological elements; rather, changes to the cell complexes are implied as the results of Boolean operations on bounded solids. After a Boolean operation has been performed (*e.g.*, the small cylinder is subtracted from the larger in Figure 2.4), holes are cut in the B-rep surface and necessary faces are added to arrive at the boundary of the resulting solid. An advantage of this approach to design is that, when an object is being designed for subsequent manufacture, very often one is interested in having a surface that does bound a 3D solid.

For the purposes of curved surface design, the approach has two main drawbacks: first, there is no direct control over the model's topology. One could imagine re-sizing or re-positioning the larger cylinder in Figure 2.4 and accidentally leaving the cylindrical socket behind, thus inadvertently changing the topology of the model as the result of a re-shaping operation. This is more of a problem in more complex models where one wishes to maintain constraints that may not readily fit into the decomposition hierarchy. The second drawback is that there is no direct control over the surface *as a surface*: a designer cannot directly reshape a face (or group of faces), since they are defined as boundaries of solids, not as free-standing surfaces.

More recent solid modeling approaches have allowed one to deal with B-rep surfaces as free-standing surfaces, so-called "non-manifold" topologies (really "non-solid"), and this allows open, bordered surfaces to be represented [Bau75, Wei85, RC86, CGP93]. Their motivation is that there are certain solid operations that seem to want to leave an open surface or unadorned edge as their result. One may represent intermediate stages in construction of a boundary surface by cutting and pasting feature boundaries [RIKM93], where the intermediate forms may not make sense as surfaces or solids. These approaches argue for being able to explicitly represent and operate on surfaces, even in a modeler that ostensibly is only concerned with generating solids.

2.3.4 Domain complexes

Ferguson, *et al.*[FRC92], present a finite-element approach to modeling arbitrary smooth topologies. In the tradition of 2D finite-element mesh generation, they address the problem of topological design for sculptured surfaces by conceptually "cutting" a desired closed surface and unfolding it into a single polygon (Figure 2.5). The originally joined edges are identified with one another and are forced to maintain complementary boundary conditions during subsequent computations (the numerical difficulties of actually doing this

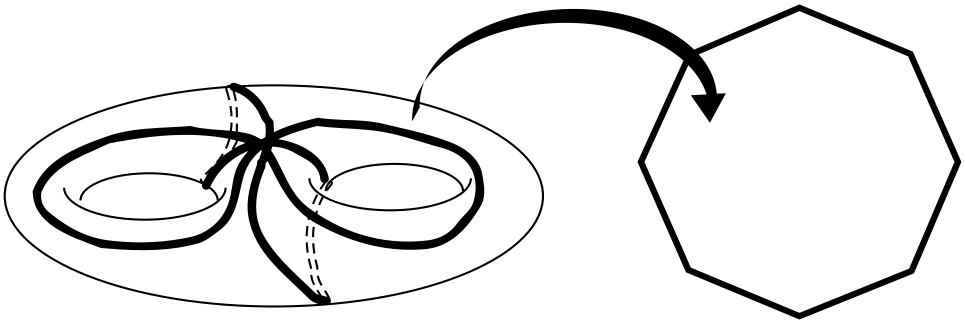

Figure 2.5: The double torus may be opened into a single octagonal domain by making a number of cuts (after *Ferguson,* et al.). In subsequent shape-related computations over this domain, these cut edges would be identified with one another using geometric continuity constraints.

will be taken up in later sections). Note that at the time the designer specifies the domain polygon, there is not an actual embedded surface to cut apart; a finite element calculation is subsequently used to compute an immersion for the domain. Incremental approaches to specifying complex topologies are more designer-friendly, allowing one to alternately reshape portions of a model and then perform surface surgery on the current immersion. These are discussed next.

2.3.5 Changing topology by directly editing cell complexes

Intermediate between the two extremes of having topology fall out as a side-effect of geometric operations (CSG) and having to completely specify topology in the absence of geometric specification (domain complexes) are schemes that allow the designer to directly operate on topological cell representation by interacting with a 3D realization of the cell complex — as with the polyhedral shells of [CK83, Nie88], or the "connect-the-dots" approach to specifying a cell complex in [MS92]. The designer works directly with the edges and vertices of a polyhedral (piecewise-linear) shell. Later, the shell edges and faces are skinned with smooth curves and surfaces.

Shells may be built up piece by piece, by connecting points with edges, then designating loops of edges as faces; but this leaves the problem of topological consistency up to the designer. Baumgart[Bau75] proposed an approach to constructing B-rep models that uses simple operations guaranteed to preserve the topological consistency of the model. One starts with a base solid and whittles away at it with "topology-safe" local operations until the desired polyhedral boundary has been achieved. A collection of topology-safe operations for surface meshes will be developed in Chapter 4.

2.3.6 Higher-level geometric operations implying topological change

Preventing the designer from directly interacting with (and botching) a cell complex is an important idea. Along these lines, others have considered automating lower-level changes to the topological representation through the use of higher-level geometric operations. Toriya, *et al.*[TTSC91] propose CSG-flavored Boolean operations on the skinned polyhedral shells above. They successfully avoid the expense of computing accurate surface intersections by making a coarse polyhedral approximation to the smooth intersection curve, deducing topological changes using this coarse curve, and ultimately incorporating the polyhedral curve into the boundary of the primary shape (which is then re-skinned with a smooth surface). Bonner, *et al.*[BJWK93] use higher-level operations to interactively build up surface topology for tubular structures (such as in auto body frames). Tubular structures are represented as quiltworks of patches offset from an underlying network of backbone curves. When the designer attaches one tube backbone to another, this triggers appropriate surgery on surface patches around the join to adjust the surface topology. The resulting models combine independent explicit faces (the flat sides of frame tubes) connected by dependent faired blending faces.

2.4 Parametric surface fitting

From the discussion above, it can be seen that free-form modeling approaches allowing direct control of arbitrary smooth surface topologies do so by embedding 2D cell complexes in 3D space. This is often accomplished by constructing a piecewise smooth surface, with a separate patch for each cell of the topological domain complex. In order to treat arbitrary topologies, one must not attempt to treat the surface as a single map from a triangulated subset of $\mathcal{R}^2$ to $\mathcal{R}^3$. It can be shown that in imposing a (u, v) coordinate system over an arbitrary genus surface, one cannot avoid introducing a number of singularities, this number being related to the genus of the surface. This is known colloquially as the "hairy ball" theorem, which says that you cannot comb the hair on a hairy ball without leaving a crown (singularity) somewhere (see [GP74] for a user-friendly discussion of the Poincaré-Hopf theorem, from which this derives). This disqualifies most techniques from the scattered-data fitting literature, as they depend on having a set of interpolation points all referred to a common parameter plane, and construct a surface function of the form $\mathbf{s}(u, v) = (u, v, f(u, v))$. Franke and Nielson[FN90] survey this type of surface construction.

Methods that map patches of $\mathcal{R}^2$ to $\mathcal{R}^3$ are known in the Computer Aided Geometric Design (CAGD) literature as *parametric* schemes. The general functional form required is vector-valued, $\mathbf{s}(u, v) = (x(u, v), y(u, v), z(u, v))$. Many particular functional forms and constructions have been developed as parametric schemes, and an excellent survey is Mann, *et al.*[MLL$^+$92]. Some additional parametric approaches not covered in the survey include subdivision surfaces[Doo78, CC78, HDD$^+$94], generalized B-splines[LD90, Loo94], and, though not exactly a parametric approach, piecewise smooth implicit surfaces that interpolate triangulated data[Baj92]. These and the surveyed approaches are all *local*; that is, each patch is constructed independently, using only data that is nearby. An important observation of Mann, *et al.*is that *none* of the surveyed local schemes produces very fair composite surfaces. That it should be so difficult to find a local construction that leads to a globally fair surface is not so surprising [1]: though there is no precise definition of fairness, and such a notion is ultimately subjective, fairness has much to do with the distribution

[1]The surprising result was that wildly differing construction schemes all exhibited similar shape defects: curvature tended to be concentrated near patch boundaries.

of curvature over the entire surface[MLL$^+$92, Mor93]. Locally constructed surfaces can be arbitrarily bad with respect to this global criterion. In these constructions, curvature information is not being communicated between adjacent patches, much less globally.

2.5 Optimal shapes: variational curves and surfaces

The inability of local patch schemes to produce high-quality composite surfaces leads us to consider global approaches to surface patch construction. Information about surface position and shape will be shared among neighboring patches, requiring that all patch shapes be solved for simultaneously. This is much more computationally involved than a local scheme; but if one cares about such global shape properties, then clearly a global scheme is indicated. A survey by Lounsbery, *et al.*[LMD92] compares some of the local schemes surveyed in [MLL$^+$92] with Moreton's high-quality global scheme[Mor93] (discussed below), and makes similar observations.

Global schemes involve geometric quality measures evaluated over the surface as a whole, rather than considered patch-by-patch. They lead to constrained optimization problems, in which some global shape measure (*e.g.*, fairness) is optimized subject to local constraints such as point or curve interpolation or global constraints on, *e.g.*, area or volume. The mathematics involved is the calculus of variations, which in its most general form allows one to pose a problem whose solution is a *function* $y = f(x)$ such than an integral objective function $\int g(x, y, y', y''...)dx$ is at a maximum or minimum subject to side conditions (constraints) on f.

The calculus of variations has its roots in several problems related to shape, including the catenary arch (an optimal load-bearing shape), the brachistochrone (an optimal shape for a roller-coaster track), and the shapes assumed by thin rods (*elastica*) under various bending moments[Boy91]. This latter is particularly relevant here, because physical splines — thin flexible strips guided through rotating sleeves called "ducks" — have long been used in drafting and design to construct fair curves that pass through specified interpolation points. The first mathematical model for such physical behavior was proposed by James Bernoulli in 1694, that the bending moment of an elastica was proportional to the radius of curvature produced. Some years later Daniel Bernoulli cast this one of the earliest variational minimum principles, suggesting that all varieties of elastica could be described as taking on shapes that minimize total bending energy $\int \kappa^2 ds$, the integral of squared curvature over their lengths. (see [Tru83] for a brief history of the elastica problem).

Elastica have long served as a basis for numerical curve fitting in CAGD, yielding fair curve functions that emulate the geometric behavior of physical splines[Far90]. More general use of minimum principles to define variational shapes[2] for design has been taken up in earnest within the past several years (see the survey in [HS90]). Variational modeling approaches are best distinguished from one another by the constraints and objective functions they use to specify shape, and the particular explicit curve and surface representations they use in constructing solutions or approximate solutions, and we will have a closer look

[2]The term "variational shape" is also used in the industrial design literature [LGL81, BS91] in a different sense, to refer to a continuum of shapes that satisfy mechanical tolerances specified as inequality constraints. This should not be confused with our usage here as the solution of a variational optimization.

at each of these below. Not all of the work reviewed below is cast explicitly in terms of variational optimization; but in the cases where the shape optimization has been formulated directly in terms of some explicit piecewise surface representation (as in, *e.g.*, [DWS93]), it is often true that a variational problem has actually been discretized, and a variational description serves to unify these disparate approaches.

2.5.1 Solving for variational shapes

Though a variety of minimum principles and variational shape problems will be discussed below, there will be no real need to discuss direct solution techniques. Though there are explicit solutions for some of the curve forms developed below, the same is not true for surfaces, for fairly deep reasons. There is a direct connection between variational surface problems and 2D systems of partial differential equations [3]. Except for the very simplest of topological domains and restricted classes of objective functions, explicit solutions cannot be found for these variational shape problems, for the same reasons that it is difficult to write down direct solutions for any but the simplest systems of partial differential equations. For the kinds of curves and surfaces considered here, the best we will be able to do is construct approximate solutions.

Approximations to variational curve and surface shapes are typically computed using either a finite-difference approach over a set of discrete sample points (*e.g.*, [KWT87, BCGH92, Ter86]), or a finite-element approach over a piecewise smooth polynomial curve (*e.g.*,[Nie74, CG91, Mor93]). At a high level, we will not make much of an issue over the differences between the two approximation approaches, because they can be viewed as the same basic approximation method (minimization of a residual error term) applied to different curve/surface representations. Because of the way derivatives are computed in a finite-difference scheme, it can be seen as numerically equivalent to a finite-element method using a C^0 (continuous, but not smooth) collection of polynomials, one for each data point neighborhood[ZM83]. Both approaches end up computing a "best-fit" of their piecewise curves to the given variational shape by re-expressing the variational objective as a set of algebraic equations written in terms of explicit curve and surface elements, a procedure known as *discretization*, then solving for the parameter values that minimize the discretized integral. The actual numerical details are taken up in Chapter 5. The reader may also consult Becker, *et al.*[BCO81], an excellent general finite element text.

The amount of computational effort needed to compute such approximations will be of concern to us, since our modeler must operate at interactive speeds. There are several contributing factors, all related to the complexity of the algebraic equations that arise when the variational problem is expressed in terms of a specific approximating representation. These include the numerical form of the approximating representation, the complexity of the objective itself, the numerical form of constraints placed on the solution, continuity constraints imposed between neighboring elements of the representation, and the topological complexity of the domain. In reviewing some of the modeling approaches below, a fundamental tradeoff will be seen: schemes that are capable of generating high quality surfaces

[3] They are related via the Euler-Lagrange equation, a basic tool in the calculus of variations. Strang[Str86] includes introductory chapters on the Euler-Lagrange equation, and on solution techniques, both symbolic and numerical.

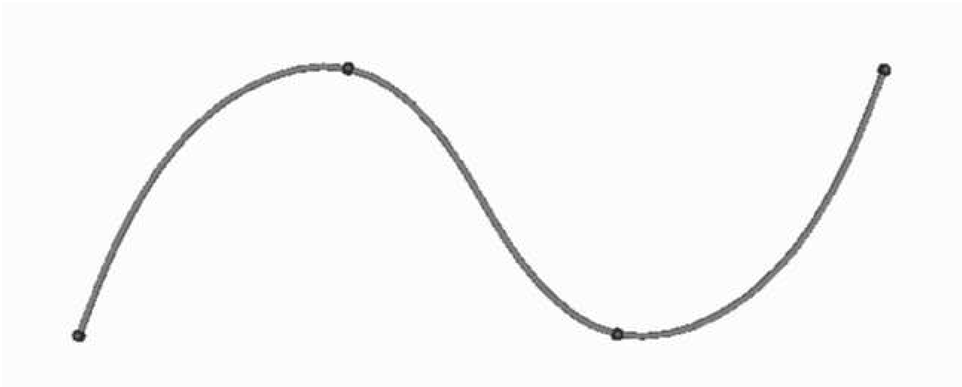

Figure 2.6: A curve that minimizes the geometric elastica functional subject to the constraint that it interpolate the given control points. This functional approximates the bending energy of a physical spline.

of arbitrary topology do not compute their surface approximations at interactive speeds; and schemes that run at interactive speeds cannot produce generally acceptable surfaces, or else are topologically limited. We will return to these issues after first considering current approaches to variational shape description in more detail.

2.5.2 Fairness functions

Mathematical measures of curve and surface fairness, like Bernoulli's measure of bending energy, are formulated as local measures that are integrated over an entire piece of geometry. Thus, a fairness function takes the shape of a curve or surface as input and returns a single number measuring its fairness. In our work, the particular value returned for a single curve or surface needn't be — and often isn't — meaningful in and of itself; what is wanted from the objective function is a relative ranking of different curve/surface shapes by fairness. The fairness functions discussed below all evaluate to 0 for the fairest possible shape, and greater than 0 otherwise. Thus, one finds the fairest shape possible by minimizing such a measure over the space of possible shapes.

Elastic curve fairness functions

As mentioned, perhaps the most venerated measure the fairness of a curve is its elastic bending energy, the integral of the squared curvature with respect to arc-length,

$$E = \int \kappa(s)^2 ds = \int (\mathbf{c}_{ss})^2 ds, \tag{2.2}$$

where the subscript indicates differentiation with respect to the arc-length parameter s, and the squared vector in the integrand is shorthand for the dot-product of the vector with itself (two conventions we use henceforth). A curve $\mathbf{c}$ is sought that minimizes E for given interpolation conditions (Figure 2.6). The usual approach is to compute an approximation by minimizing E over a given algebraic curve. It is unrealistic to expect such a curve to be parameterized by arc-length, and for the more general case, E becomes more complicated:

$$E = \int \kappa(s)^2 ds \tag{2.3}$$

$$= \int \kappa(t)^2 ||\mathbf{c}_t|| dt \tag{2.4}$$

$$= \int \frac{||\mathbf{c}_t \times \mathbf{c}_{tt}||}{\mathbf{c}_t \cdot \mathbf{c}_{tt}} dt \tag{2.5}$$

with respect to an arbitrary parameterization t. The value of E doesn't depend on the particular way the curve has been parameterized, since it is normalized with respect to an arc-length parameterization — that is, the measure is *geometric* [4]. Simpler *parametric* measures are discussed below, measures that do depend on parameterization. But a basic assumption of this dissertation is that geometric measures are worth their added complexity. Parameterizations are an artifice in geometric modeling — something needed at a low level in order to do calculus on a curve or surface (to measure shape properties), but not something a user should ever worry about. More importantly, when considering smooth surfaces of arbitrary topology, geometric measures are absolutely necessary for the simple reason that global parameterizations do not always exist.

E is a nonlinear function, and computing an approximate minimizer will require iterative numerical techniques, much like those used to solve systems of nonlinear differential equations, and accompanied by similar difficulties[PTVF94]. Because of the complexity of working with Equation 2.2, a variety of simplified approximations have been proposed. The simplest approximation minimizes $\int (\mathbf{c}_{tt})^2 dt$, the squared magnitude of the second parametric derivative. When a linear curve representation is used (Section 2.5.4), minimizing this version of E becomes a linear problem. This linearization will generate reasonable shapes as long as the curve parameterization resembles a scaling of arc-length parameterization (that is, as long as $||\mathbf{c}_t||$ doesn't vary much over the length of the curve).

Unfortunately, this is not a good assumption to make about the parameterization of a fitted curve. Schweikert[Sch66] considered this approximation, and noted unwanted inflection points in segments with "excessive" arc-length relative to the parameterization. He recommended reigning in the arc-length by adding a *membrane* term to the objective:

$$E = \int (\mathbf{c}_{tt}^2 + \alpha \mathbf{c}_t^2) dt, \tag{2.6}$$

which opposes large values of $||\mathbf{c}_t||$ by causing the curve to contract. The differential equations describing curves of this form can be solved directly, yielding an explicit curve known as the *exponential spline*, although it is much more common in modeling systems to approximate these shapes with composite curves[NLL90, Kal93, Cel90, WW92]. Of course, for a

[4]More confusion in terminology: in some of the elasticity literature cited here, such a parameterization-independent measure would be referred to as *intrinsic*. In differential geometry literature, on the other hand, *extrinsic* and *intrinsic* describe measures that do or do not depend on a manifold's embedding in ambient space. A curve's curvature at a point would be considered intrinsic by the former, extrinsic by the latter. In CAGD literature, where parameterization-independence and embedding-independence are both useful notions, *intrinsic* retains its differential-geometric meaning, and parameterization-independent extrinsic measures are called *geometric* (well, usually...).

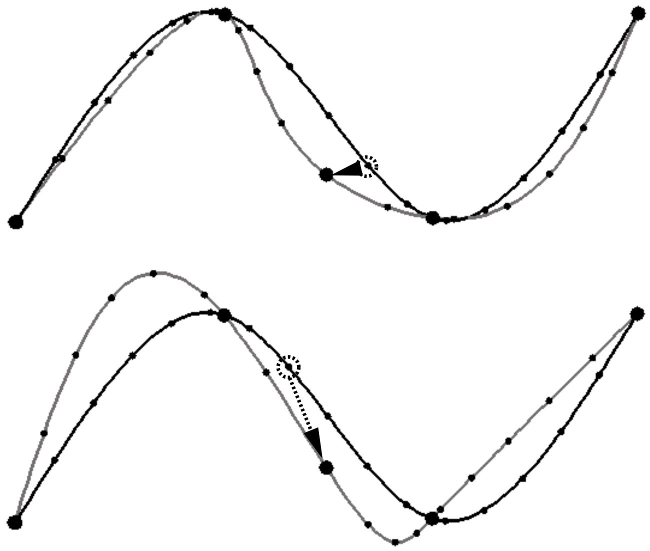

Figure 2.7: Composite B-spline curves that minimize the Schweikert functional subject to interpolation constraints (large dots), and their associated uniform parameterizations (small dots). Identical original curves (in black) were grabbed at different parametric points, and these new constraint points were moved to identical 2D positions (resulting curve shapes shown in grey). The before/after pairs demonstrate how global curve shape depends on the constraint parameterization, and how a poorly parameterized constraint point can degrade the resulting global curve shape.

true arc-length parameterization the tangent magnitude $\|\mathbf{c}_t\| = 1$ by definition, so having a membrane term drive it to 0 is not really what one wants to do. We have experimented with penalizing the deviation of $\|\mathbf{c}_t\|$ from 1, but this does not work well for low-degree polynomials (which simply do not seem to want to take on arc-length parameterizations).

Any linearization of E will have the drawback over the original geometric form of sensitivity to the parameterization of the fitted curve. A popular way of addressing this limitation, when fitting composite curves, is to construct the composite domain to approximate an arc-length parameterization using a *chordal approximation*[Far90, Epp76, PS83, RF89, SM91, HB91b]. As an example, consider a sequence of discrete sample points to be used in a finite-difference scheme (Figure 2.8). Rather than spacing these points at equal parametric intervals as is often done in smooth interpolation methods, a chordal parameterization approximation takes the straight-line Euclidean distance between the points as their parametric separation. A chordal parameterization converges on an arc-length parameterization in the limit as smaller and smaller parametric intervals are considered.

This is an approximation and the actual fitted curve will of course have greater arc-length than the chordal parameter intervals (unless the fitted curve is a straight line). If the parameterization remains fixed but the curve is subsequently reshaped, this approximation may become arbitrarily bad, and the 2nd derivatives cease to be good approximations to curvature. The overall quality of the shape then decreases, as illustrated in Figure 2.7. One

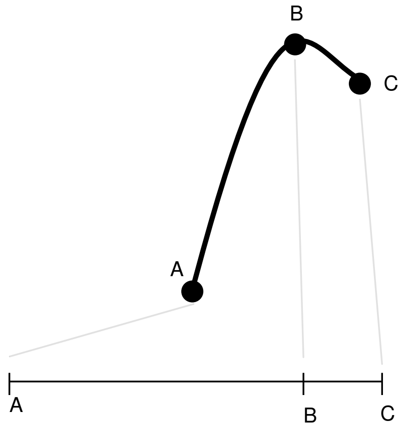

Figure 2.8: A curve, and beneath it the resulting chord-length parameterization of the indicated points. The parametric distance between successive points is taken to be their Euclidean distance.

way of minimizing this effect is to allow constraint points to "slide" on the curve as it is re-shaped, something we consider further in the discussion on constraints in Section 2.5.3. A different way of addressing this is to actually re-build the composite parameterization from the ground up when the curve changes shape, an approach developed in Chapter 5.

Higher-order curve fairness functions

Finally, it should be noted that 2nd-order elastica are not the last word in fair curve shapes. Other authors have made use of higher-order derivatives, including 3rd-order ([Meh74, HB91b, Mor93]) and 4th-order (mentioned in [HS90]) terms. Moreton[Mor93] pointed out that minimizing bending energy alone tends to concentrate curvature near the endpoints of a faired region. He uses a curve fairness functional that measures the *variation* of curvature over the interval, $\int (\frac{d\kappa}{ds})^2 ds$. Minimizing this functional yields curves that seek constant curvature, and it finds circles and lines whenever they are possible as solutions to the given interpolation problem, with results that look much more "draftsman-like" than those arising from strain energy. This is quite a bit more complicated an objective function than the elastica, and although computing approximations is algorithmically similar, the calculations are correspondingly more delicate.

Thin plate surface objectives

Much of the calculus of curve shape carries straight over to surfaces through the use of cross sections. One can measure the curvature of a surface at a particular point in a particular tangent direction by slicing the surface with a perpendicular cutting plane and measuring the curve of intersection. The result is called the *normal section curvature* in the given direction[O'N66]. One of the first results in the classical differential geometry of surfaces, due to Gauss[Spi79a], is that the sectional curvature in the neighborhood of a surface point is a smooth function of tangent direction and takes on its maximum and minimum values (the *principal curvatures* κ_1 and κ_2 — the subscripts do not indicate a derivative) in orthogonal tangent directions. The principal curvatures completely characterize the shape of the surface about a point, and in turn give rise to important geometric quantities such as Gaussian curvature ($\kappa_1\kappa_2$) and mean curvature ($\frac{\kappa_1+\kappa_2}{2}$).

The elastic curve functional of Equation 2.2 can be generalized to surfaces using principal curvatures:

$$\int (\kappa_1^2 + \kappa_2^2) du\, dv. \tag{2.7}$$

The result is commonly called the *thin plate* functional, because it approximates the strain energy of a thin elastic plate[TL25] (Figure 2.9). When expressed with respect to an arbitrary parameterized surface, this simple geometric form becomes much more complex, and we do not write it out here (see, *e.g.*, Hagen and Schulze[HS90]). This is a standard criterion for surface fairness[NR83, LP88], and has been used on a small scale as the basis for variational "twist-elimination" schemes in parametric patch constructions[HS90, KR90]. Nonetheless, Equation 2.7 does not seem to be widely used (if at all) to fair composite surfaces having many free parameters, perhaps because of the complexity of its parametric form and the computational difficulties of optimizing such a highly nonlinear objective function over a large number of degrees of freedom.

Much more popular are linearized forms of Equation 2.7 that use second-order parametric derivatives in place of principal curvatures:

$$E = \int (\mathbf{s}_{uu} \cdot \mathbf{s}_{uu} + \mathbf{s}_{vv} \cdot \mathbf{s}_{vv}) du\, dv, \tag{2.8}$$

or including a "twist term":

$$E = \int (\mathbf{s}_{uu} \cdot \mathbf{s}_{uu} + 2\mathbf{s}_{uv} \cdot \mathbf{s}_{uv} + \mathbf{s}_{vv} \cdot \mathbf{s}_{vv}) du\, dv. \tag{2.9}$$

As with curves, 2nd-order terms alone are not quite enough to control the surface, this time because there is no area-related feedback. Lott and Pullin [LP88] augment Equation 2.8 with 1st order springs to keep the surface from straying far from its original shape. Much more prevalent is the use of 1st-order membrane terms, analogous to Schweikert's curve functional[Ter86, Pot91, Nie74, CG91, WW92, HKD93]:

$$E = \int \left(\begin{array}{c} \alpha(\mathbf{s}_{uu} \cdot \mathbf{s}_{uu} + 2\mathbf{s}_{uv} \cdot \mathbf{s}_{uv} + \mathbf{s}_{vv} \cdot \mathbf{s}_{vv}) \\ + \\ \beta(\mathbf{s}_u \cdot \mathbf{s}_u + 2\mathbf{s}_u \cdot \mathbf{s}_v + \mathbf{s}_v \cdot \mathbf{s}_v) \end{array} \right) du\, dv. \tag{2.10}$$

This approximation has drawbacks similar to the those of the corresponding curve functional, this time when surface areas don't match a uniform scaling of their parametric areas. This can happen if interpolation points are poorly parameterized, or if the surface begins with a reasonable parameterization but is unevenly stretched in the course of re-shaping by the user, as illustrated in Figure 2.9.

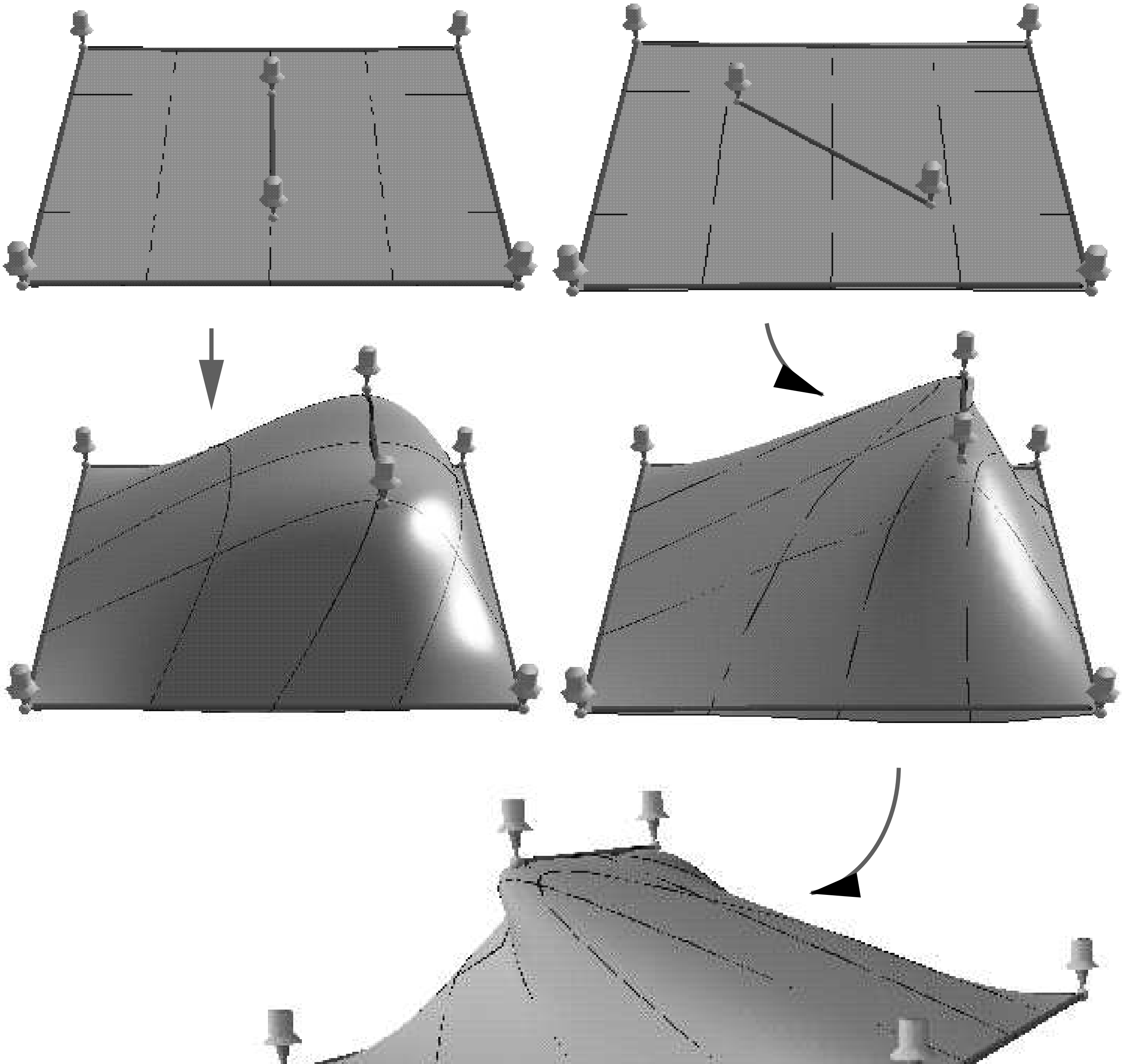

Figure 2.9: Surfaces that minimize a linearized strain energy functional can still have unpleasant shapes, because of mismatches between constraint parameterizations and the surfaces' actual metric. For the surfaces on the left and the right, a constraint curve has been added and then moved, each to a similar final position. What differs is the set of (u, v) surface coordinates associated with these constraint curves. The surface on the right suffers a serious shape defect as a result of parametric "twisting" (the bottom image is of this surface rotated to show its other side).

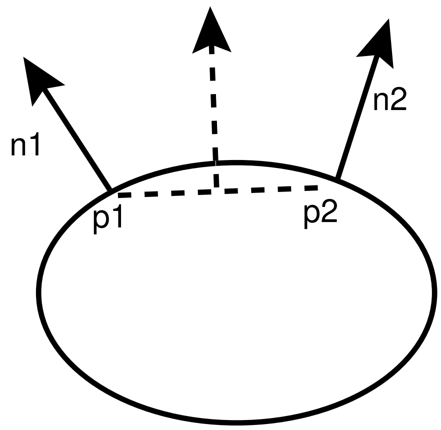

Figure 2.10: Surface points and normals used in the co-circularity potential.

Higher-order surface fairness functions

Thin plate energies, which use 1st and 2nd order information, are not the only possible fairing functions, or even the best. As with curves, these functions concentrate curvature near patch boundaries. Moreton[Mor93] proposed a fairing function that measures the variation of principal curvatures over the surface:

$$E = \int (\frac{d\kappa_1}{d\hat{e}_1})^2 + (\frac{d\kappa_2}{d\hat{e}_2})^2 dA, \tag{2.11}$$

where the $\hat{e}_i$ are the principal directions associated with the principal curvatures κ_i. Because of the explicit appearance of principal directions in this objective, it is a *much* more complicated function than its curve counterpart, and a delicate numerical optimization is needed to compute finite-element approximations of its minimizers. The method yields surfaces with an even distribution of curvature throughout, whose shapes seek circular or straight-line cross-sections. These are visually superior to surfaces produced by other published schemes, on a wide variety of interpolation problems.

Related but simpler is the *co-circularity* potential used in Szelizki, *et al.*'s oriented particle systems[ST92]. It takes position and normal vectors from two nearby points on a surface (Figure 2.10), and measures the deviation of their average direction from the normal "between" them:

$$V(\mathbf{p}_1, \mathbf{n}_1, \mathbf{p}_2, \mathbf{n}_2) = ((\mathbf{n}_1 + \mathbf{n}_2) \cdot (\mathbf{p}_1 - \mathbf{p}_2))^2 w(\|\mathbf{p}_1 - \mathbf{p}_2\|), \tag{2.12}$$

where the $\mathbf{p}_i$ are positions, $\mathbf{n}_i$ are normals, and $w(r)$ weights the potential by the distance between the points. This potential is 0 for points on the surface of a sphere, and at umbilic points on any surface (points where the sectional curvature is the same in all directions). Unlike Equation 2.11, it is not 0 for cones, cylinders, or tori, because it demands that curvature be the same in all directions rather than teasing apart the surface curvature into its principal components and treating them separately. Szeliski includes a finite-element analysis of its behavior in [ST91], in which he shows that, for small deflections of a surface element represented as a graph of a function f $(s(u, v) = (u, v, f(u, v))$, the co-circularity energy over the element is approximated by

$$E \approx \int\int f_{uuu}^2 + 3f_{uuv}^2 + 3f_{uvv}^2 + f_{vvv}^2 du\, dv. \tag{2.13}$$

This is simply a 3rd-order analog of Equation 2.10, and the same caveats hold in applying this parametric "curvature variation" measure as an objective function.

Minimal surfaces, mean curvature, and the Laplacian operator

A different class of geometric fairing function uses the notion of a *minimal surface*, a surface interpolating a given boundary curve with the minimum possible surface area. The physical analogy here is to a soap film stretched between wire loops. Such films take on graceful free-form shapes in interpolating their given boundaries. The fundamental result in the theory of minimal surfaces is that their mean curvature, $H = \frac{\kappa_1+\kappa_2}{2}$, is everywhere 0, a straightforward exercise in the calculus of variations (see Spivak[Spi79c] for a discussion of the history of minimal surfaces in analysis).

As it happens, the published fairing approaches based on minimal surfaces are both point-based unstructured mesh schemes — a departure from the discussions so far, which have focused on fairing functions formulated with respect to smooth surface patch parameterizations. Delingette, *et al.*[DWS93] make a geometric approximation to mean curvature over a mesh with hexagonal topology. Fairing a mesh with this geometric objective requires an iterative nonlinear optimization to drive an initial mesh to a final minimum shape.

As with elastica there is also a straightforward parametric (linearized) approximation of mean curvature. It can be shown[O'N66] that mean curvature is given by the average of the curvatures measured in any two orthogonal tangent directions at a point on the surface, not just the principal directions. Thus, given an orthonormal u, v coordinate system at a point on a surface, the condition of 0 mean curvature is described by the well-known *Laplace equation*:

$$\mathbf{s}_{uu} + \mathbf{s}_{vv} = 0 \tag{2.14}$$

By integrating the square of the Laplacian $\mathbf{s}_{uu} + \mathbf{s}_{vv}$ over the surface one obtains a quadratic objective function. This is perhaps as good a rationalization as any for Mallet's proposed mesh "roughness" criterion[Mal89]:

$$E = (v^c - \sum_{i=1}^{n} v^i/n)^2, \tag{2.15}$$

where v^c is the center vertex of a mesh neighborhood and the v^i are the neighbor vertices. When this is applied to a rectangular mesh, it reduces to the finite-difference form for the squared Laplacian.

What seems to have been missed in using minimal surfaces for fair surface design is that they only make sense in situations where interpolation constraints occur along fixed boundary curves rather than at interior points. Your physical intuition about soap films should tell you that if interpolation points or curves are present in the interior, the membrane will happily crease or corner at these places. Likewise, it should be clear that one cannot directly control the orientation of the tangent plane at a point or along a curve in a membrane, as is possible with elastica (see the "tangent ribbon" discussion below). In fact, such membrane interpolation problems are standard examples of variational problems for which no smooth solutions exist[CH37]. It is not entirely clear why approaches based on minimal surfaces have actually demonstrated smooth interpolation of interior constraints. It seems safe to say that such "smoothness" must ultimately be due to approximation error.

2.5.3 Constraints

Constraints are used in variational shape specification to impose explicit control over some aspect of the shape — that it interpolate a particular point or curve, maintain tangency to some other object at a point, *etc.*. Nowacki[NLL90] surveys a variety of geometric constraints for design in conjunction with piecewise polynomial curves and surfaces. He considers local interpolation constraints involving interior points, endpoints, and tangent ribbons at surface patch boundaries; and global *integral* constraints such as surface area or volume. Similar local constraints, and also a general curve interpolation constraint are considered by Celniker and Welch in [CW92]. All this work uses parametric curves and surfaces, and the constraints themselves are *parametric*, formulated with respect to the natural parameterization of the curve or surface:

Point constraint on surface:

$$(\mathbf{s}(u_0, v_0) - \mathbf{X}_0)^2 = 0, \tag{2.16}$$

where $\mathbf{X}_0$ is a fixed point in space, and $\mathbf{s}(u_0, v_0)$ is a point on the surface with fixed u, v coordinates u_0, v_0.

Curve interpolation constraint:

$$\int (\mathbf{s}(u(t), v(t)) - \mathbf{c}(t))^2 dt = 0, \tag{2.17}$$

where the parameter t and the surface curve $(u(t), v(t))$ associate points on s with points on the constraint curve $c(t)$. It is common to refer to the parametric coordinates $(u(t), v(t)$ and (u_0, v_0) as *material* coordinates, as if (u, v) grid lines were painted on the "material" of a deformable curve or surface. Figure 2.9 illustrates these constraints applied to a surface patch.

This need for material coordinates is an artificial constraint if one wants to specify merely that *some* point on a curve or surface coincide with a fixed point in space, or that a surface should contain a given space curve along some cross-section within a given region. As was discussed earlier, having "good" material coordinates for constraints is especially important when computing shape approximations. Chord-length constraint parameterizations are often used to address this issue for static data fitting; but clearly the approximation can become arbitrarily bad if these interpolation points are used as shape handles and manipulated by the user after being assigned fixed material coordinates.

Ideally, the material coordinates of the interpolation points would remain free, much like physical splines are left free to slide within their positioning ducks to a minimum energy shape. Something like this was considered for scattered data fitting using curves (interpolation constraints but no shape optimization) by Hoschek[Hos88, HSW89] and Sarkar and Menq[SM91], and for surface fitting by Plass and Stone[PS83] and Rogers and Fog[RF89].

The technique has been referred to as "parameter optimization," because one ends up computing an optimal set of material coordinates for the interpolation points such that overall fitting error is minimized. We refer to such parameterless constraints as *nonparametric interpolation constraints.*

Curve and surface tangent vectors may be constrained with formulae similar to Equation 2.17, using parametric derivatives of the curve and surface functions. These allow *tangent ribbons* to be controlled along the edge of a surface, and may be used to "stitch" curves or surfaces together smoothly by matching derivatives along a shared boundary. They are known, not surprisingly, as *parametric continuity* constraints[Far90], with nth-order parametric continuity typically designated C^n in CAGD literature. A more general continuity condition uses geometric rather than parametric quantities, and is therefore known as geometric continuity[DeR90]. As an example, first-order (G^1) geometric continuity equates normal vectors along a shared boundary:

G^1 **surface continuity constraint:**

$$\int (\mathbf{N}(\mathbf{s_1}(u_1(t), v_1(t))) - \mathbf{N}(\mathbf{s_2}(u_2(r(t)), v_2(r(t)))))^2 dt = 0, \tag{2.18}$$

where

$$\mathbf{N}(\mathbf{s}(u,v)) = \frac{\mathbf{s}_u \times \mathbf{s}_v}{||\mathbf{s}_u \times \mathbf{s}_v||}, \tag{2.19}$$

$\mathbf{s}_1$ and $\mathbf{s}_2$ are surfaces intersecting along their respective parametric curves (u_1, v_1) and (u_2, v_2), and $r(t)$ is a reparameterization of (u_2, v_2) such that t identifies coincident points on both surface curves.

Lastly, there are the so-called *integral* constraints, such as constraints on the area contained in a closed planar curve, area of a bounded surface, or volume contained within a closed surface:

Surface area constraint:

$$\int_s ||\mathbf{s}_u \times \mathbf{s}_v|| du dv = A_0 \tag{2.20}$$

These are geometric constraints, as the area of a surface, or contained volumes, do not depend on the particular parameterization used in performing the integration.

Applying constraints

In applying any of the constraints that have been discussed to a particular surface representation, as when computing an approximation to a constrained variational surface, it may happen that the given representation cannot exactly satisfy the constraint. For example, if a polynomial surface patch is constrained to interpolate a circular arc, the interpolation cannot be exact. Generally, one must be satisfied with minimizing constraint error, *e.g.*in the least-squares sense, rather than satisfying constraints exactly.

For parametric constraints — involving fixed material coordinates for point or curve positions and parametric derivatives in fixed directions — a least-squares formulation of the constraint leads to a quadratic minimization problem. As with the linearized objective functions of the previous section, this yields a set of linear constraint equations to be solved for a zero error gradient. The problem of minimizing a quadratic objective subject to linear equality constraints is a straightforward numerical optimization problem, requiring the solution of a single linear system, assuming a linear curve or surface representation is being used.

There are any number of approaches to setting up and solving this system[PTVF94, GVL89, LH74, MS78, GMW81]. Perhaps the simplest directly combines the constraint error with the shape objective as a penalty term to be minimized[WW92, MS92]. The drawback with penalty approaches is that the constraints must compete with the objective, and thus will not generally be satisfied exactly. One must heavily weight the constraint error to increase its dominance over the objective terms, and this can lead to badly conditioned equations[GVL89]. There are several approaches that enforce constraints exactly. In some curve and surface representations, constraints are "built in" so that they cost nothing to enforce, as with parametric continuity constraints for B-splines (below), or control-point interpolation constraints for Catmull-Rom splines (below). Linear constraints can also enforced by *constraint reduction*[GMW81], in which a matrix is computed that transforms the original constrained representation parameters into a smaller set of unconstrained parameters for which the constraints are built-in (see [Cel90, CW92] for applications to curve and surface fairing). Finally, the technique of *Lagrange multipliers*[Str86], which augments the representation parameters with additional degrees of freedom, one per degree of constraint, has also been used to enforce parametric interpolation and tangent constraints[WW92]. Constraint reduction and Lagrange multipliers are used in the approximation scheme of Chapter 5.

For geometric constraints, things are a good deal more complicated numerically because the constraints are typically nonlinear. For nonparametric interpolation constraints, there *will* be material coordinates for the interpolation points, and these must be left free to vary. That means gradients of the constraint error with respect to these material positions inherit the nonlinearity of the curve or surface position function. To make this distinction between the linearity/nonlinearity of parametric/geometric interpolation constraints clearer, consider a general linear parametric curve $C(t)$, which by definition may be written as a linear combination of parametric basis functions $B(t)$:

$$C(t) = \sum_i \alpha_i B_i(t), \tag{2.21}$$

where the subscripts refer to the ith element of the set. This form, to be discussed in more detail in the next section, should not be confused with a p.l. curve, which is one of the simplest instances of this more general class of curves. The position of the curve at a fixed parameter value t_0 is a *linear* function of the α_i. The position of the curve as t_0 varies is typically a *nonlinear* function, since the basis functions B_i are typically nonlinear in their parameter t. If error or objective gradients are formulated with respect to free t parameters rather than the α_i, they pick up this nonlinearity. Such constraints cannot generally be solved directly, but must be satisfied by iteratively minimizing constraint error.

Geometric continuity conditions suffer a different kind of nonlinearity, arising from their use of vector cross-products and division by vector magnitudes to produce unit vectors (Equation 2.18). Additionally, there are *corner compatibility conditions* that must be satisfied where multiple patches meet at a single vertex[Pet91]. Essentially, all surfaces must meet at a 2nd-order smooth neighborhood at such a node if their 1st-order cross-boundary derivatives are to be compatible. It is most common to directly construct surfaces that satisfy these constraints ([SS87, Pet91, DeR90, LD90]). But combining such a construction with a global surface fairing scheme is daunting because the constructions typically over-constrain the surface in the name of definiteness, leaving few if any degrees of freedom available to minimize the objective (subdivision surfaces being a notable exception[HKD93], discussed below). It will generally be more practical and productive to satisfy such constraints through iterative relaxation[MS92] of an under-constrained surface.

2.5.4 Linear curve and surface representations for approximation

Several times the previous discussion made reference to *linear* curve and surface representations as being desirable for approximating variational shapes. Such a curve or surface can be written as a linear combination of a set of parametric *basis functions*:

$$\mathbf{c}(t) = \sum_i \alpha_i b_i(t), \tag{2.22}$$

or

$$\mathbf{s}(u, v) = \sum_i \alpha_i b_i(u, v), \tag{2.23}$$

where the subscripts indicate the ith element of the set. There are a variety of representations in use in CAGD that satisfy this definition. We will not go into any real detail, beyond discussing a few broad classes of basis functions — see Farin[Far90] for a survey of the mathematical particulars. Here we are interested in how linear curve and surface elements can be pieced together into piecewise smooth composites, and ultimately be used to approximate variational shapes. It is worth mentioning at this point that the work in later chapters will not be using *any* of the piecewise smooth schemes discussed here; but the background is needed in explaining our choices later.

Polynomial curves

Perhaps the most commonly used basis functions are univariate polynomials — B-spline, Bezier, Catmull-Rom, or the simple monomial functions $(1, t, t^2, t^3, ...)$. Though all generate polynomial curves, their α_i have different geometric meaning as points distributed on or about the curves, and are referred to as *control points* (except for the monomial's α_i, which are simply polynomial coefficients). The number of control points in a curve segment is always one more than the polynomial degree of the curve, just as a degree-n polynomial has $n-1$ coefficients. As Figure 2.11 illustrates, B-spline and Catmull-Rom curve segments can be chained end-to-end to make piecewise smooth composites by sharing control points between neighboring curve segments. These are "built-in" parametric continuity conditions. Geometrically continuous composites may be built up by construction, or by setting up and solving continuity-constraint equations between (no longer shared) control points of neighboring segments. Piecewise smooth Bezier curves or blends of monomial curves, with either parametric or geometric continuity, must be built by construction or though explicit constraints. There is no analog to control-point sharing here, although, assuming one is more interested in geometric continuity constraints, the Bezier and polynomial forms are simpler to work with in setting up the constraint equations.

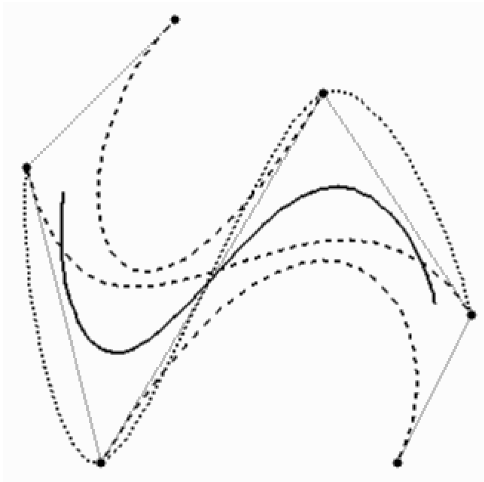

Figure 2.11: Polynomial curves: cubic Bezier (dashed), Bspline (solid), and Catmull-Rom (dotted) spline segments are computed for the same sequences of control points. Each cubic segment uses 4 successive control points, so the 6 control points yield sequences of 3 curve segments. The three Bezier curve segments are discontinuous. The three Catmull-Rom and B-spline segments are C^1 and C^2 continuous, respectively, because of control point sharing.

Polynomial surfaces

Linear surfaces may be built up from such univariate basis functions through the *tensor product* operation:

$$\mathbf{s}(u, v) = \sum_i \sum_j \alpha_{ij} b_i(u) b_j(v), \tag{2.24}$$

where α_{ij} is now a 2D rectangular array of control points. As with the 1D forms, these patches can be stitched together into parametrically continuous sheets by control point sharing (B-spline, Catmull-Rom), or by constraints or direct construction (Bezier, monomial). Unfortunately, because patches are rectangular and can only be joined into rectangular meshes (4 patches must meet at each interior vertex), the only topologies possible with simple parametric continuity conditions are sheets, cylinders, and tori. More general topologies require interior vertices where more or fewer than 4 patches meet smoothly, or the use of other than 4-sided patches (*i.e.*, by collapsing an edge of a patch control mesh to create a "triangular" patch). Both approaches introduce degeneracies into the relationships between partial derivatives at the shared vertices, and cause the parametric continuity condition to break down. The smoothness of such neighborhoods must be ascertained using geometric continuity conditions.

Other linear surface patches do not rely on the tensor-product construction, but are instead use true 2D basis functions. The most widely used are Bezier triangles[Far90], a triangular generalization of the Bezier curve construction. Triangular patches are more convenient than rectangular ones for constructing nontrivial surface topologies, and the complexity of maintaining geometric continuity constraints or evaluating a geometric objective is essentially the same for both.

Subdivision Surfaces

A subdivision surface is defined, not as an explicit parametric function, but rather as the limit of repeated refinement of an initial control mesh[Doo78, CC78, Loo94, HDD+94]. The resulting surfaces are smooth, can be of arbitrary topological type, and are linear in the initial mesh vertices (though, like their B-spline cousins, they do not directly interpolate these points). It is possible to describe these iterated subdivision steps as repeated applications of a particular linear transformation to the patch control points, and to analyze the limit behavior of the iterated transformation to determine the ultimate locations of surface points and normals corresponding to the original mesh vertices. (see [HKD93]).

Work on global shape optimization with subdivision surfaces is very recent. Hoppe, *et al.*[HDD+94] attract such a surface to an unorganized set of points, minimizing the difference between limit positions of surface points and nearby data points. Halstead, *et al.*[HKD93] compute fair Catmull-Clark subdivision surfaces by minimizing the parametric thin-plate functional of Equation 2.10. The difficulty here is in evaluating the integral (recall that there is no explicit parametric equation for the surface). They show how to compute the integral and necessary derivatives analytically, by analyzing the eigenstructure of the subdivision matrix. It is not quite clear what it means to minimize such a parametric function over a composite, geometrically continuous (but not parametrically continuous) surface. It would appear that the objective function measures the interior of each patch independently, and any fairness across patch boundaries is a side-effect of coupling between vertices shared by neighboring patches.

2.6 Interactive variational shape design

Having discussed the variety of approaches to the individual components of a variational shape specification and its subsequent approximation, we will now look at the several interactive variational shape design approaches[Cel90, Kal93, WW92, BB89, Fow92]. Here we are interested in approaches that compute their surface approximations in real time, as the user interacts with the variational specification by modifying constraints. All the approaches considered here make similar choices from the constraint, objective, and representation menus above. All constraint and objective functions are parametric, rather than geometric.

Bartels'[BB89] and Fowler's[Fow92] approaches allow point interpolation constraints on B-splines, but minimize no explicit fairing function. However, it is easy to show that their constraint solution technique minimizes the RMS displacement of points on the curve from their initial positions to their final positions. Celniker[CG91], Kallay[Kal93], and Welch[WW92] use the fairing function of Equation 2.10, and all use linear C^1 and C^2 constraints to stitch patches together (with the exception of Celniker[CG91], whose triangular patches allow almost-everywhere G^1 continuity). All offer point interpolation constraints with fixed material coordinates. Additionally, interactive curve position and tangent ribbon constraints on surfaces are developed in [CG91, CW92, WW92].

In every case, the choices of parametric functions over more versatile, higher quality geometric ones were motivated by two needs: the need to have a numerical optimization problem that could be solved at interactive speeds, and the need for robust numerics in the face of user interaction. Both of these needs are satisfied because the parametric formulations yield linearly constrained quadratic minimizations, whose solutions can be found by solving a single large linear system in a single step, rather than through iterative minimization as is required for nonlinear geometric schemes. And the solutions can be found relatively quickly: as long as no new constraints are added or old ones deleted, and as long as the representation's refinement is not changed, the matrix associated with the linear system remains constant; thus, it can be factored once, and thereafter used to solve for new surface shapes as the designer changes constraint values (*e.g.*, by moving a control point)[CG91, Fow92, CW92, WW92]. Because the solution can be found directly, it is not necessary to construct a good initial guess at the solution prior to iterative minimization in order for the solution to converge. This in turn means that it is safe for the user to interactively reposition constraints *while the solver is running* without fear that the minimization process will be destabilized. Having the user's interactions with the shape specification out-pace an iterative solver's ability to keep up can be a real problem for interactive systems incorporating nonlinear optimization. In such approaches, there is an implicit dependence on the user to change parameters smoothly and at an appropriate rate, relying on visual feedback as the solution evolves to judge how quickly changes can reasonably be made[WGW90, GW91, Gle94].

The shortcomings of parametric constraint and objective functions were outlined previously. These problems are even more pronounced in an interactive modeler, where the user may severely distort a surface region while tugging on control points or curves. The designer is forced to think of the surface in terms of fixed material coordinates, and be careful not stretch the surface nonuniformly. This is exacerbated by the requirement that linear constraints also be expressed in material coordinates, preventing the surface from "sliding" across control points and curves to achieve fairer shapes. Finally, recall that those interactive modelers that rely on tensor-product surfaces[Kal93, WW92, BB89, Fow92] are incapable of representing topologies other than sheets, cylinders, and tori. In summary, current interactive schemes, though fast and stable, use simplified formulations that suffer a variety of difficulties due to their dependence on an underlying surface parameterization.

2.7 Triangulated surfaces

The surface modeling approach developed in the next few chapters does not use smooth parametric patches as its basic representation, but rather, discrete points connected in a surface mesh. The previous section mentioned some earlier mesh-based surface fairing schemes ([Mal89, DWS93]). This section reviews some lower-level issues involved in computing with meshes: how to estimate smooth quantities such as curvature at node vertices.

If mesh is to be treated as an approximation to smooth surface, we need to be able to estimate smooth quantities such as curvature and normals over the mesh. Simple, standard approaches to estimating gradients (tangent planes) for scattered data[Law77, Aki84, Ste84, Fra82, She68] are not adequate for our purposes, as they assume all points are referred to a single global u, v parameterization. They take as input a collection of data point triplets (u_i, v_i, f_i) and at each point compute gradients of a surface $s(u, v) = (u, v, f(u, v))$ by fitting some functional form for f to the point and its neighbors. The functions used (*e.g.*, planes, quadratics[Fra82], Shepard's surfaces[She68], and others) and the method of deciding which

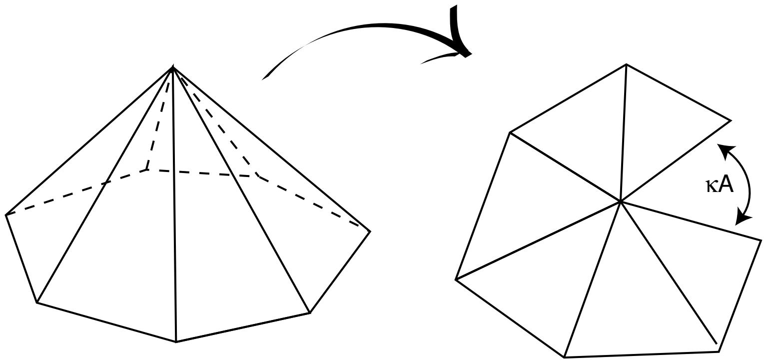

Figure 2.12: The Gaussian curvature κ at a polyhedral vertex is the "excess angle" when the neighborhood is flattened out, scaled by the neighborhood surface area.

points are in a given point's neighborhood distinguish these methods from each other, good surveys being Nielson and Franke[NF83, FN90].

Approaches that can be applied to true 3D data are somewhat rarer. Any of the parametric surface fitting approaches discussed in Section 2.4 might be used within a mesh face, but these methods are either inappropriate or are overkill for estimating smooth quantities at mesh points only. We consider here a number of geometric and parametric fitting schemes for approximating gradient (normal field) and curvature information at 3D surface mesh vertices.

2.7.1 Mesh surface normals

Perhaps the most widely used approach to computing a surface normal at a mesh vertex takes an area-weighted average of the normals of the faces incident at the point (*e.g.*,[Ham93, MS92, Tur92, SZL93]). Variations on this idea use different weightings, *e.g.*, Akima's inverse area weighting [Aki84].

One may instead solve for the normal to a plane that best fits the points in a nearby a neighborhood, as in Hoppe, *et al.*[HDD+92]. This is a standard linear regression technique, in which a set of neighborhood points Nhd and a given "center" point $\mathbf{c}$ one forms the 3 x 3 covariance matrix

$$\mathbf{S} = \sum_{\mathbf{y} \in Nhd} (\mathbf{y} - \mathbf{c})(\mathbf{y} - \mathbf{c})^T, \tag{2.25}$$

The smallest eigenvalue of $\mathbf{S}$ is the normal to the plane passing through $\mathbf{c}$ and best-fitting the points of Nhd.

2.7.2 Mesh curvature

Perhaps surprisingly, it is possible to compute the *Gaussian* curvature at a surface mesh node exactly. Maxwell[Max54] originally observed that the Gaussian definition of intrinsic curvature by means of spherical projection[Huf75] readily applies to polyhedral surfaces. Because the faces are flat and edges straight, all the intrinsic curvature must be concentrated in the vertices themselves, and it can be shown (by spherical projection) that the Gaussian curvature K at a vertex is

$$K = \frac{\beta}{A}, \tag{2.26}$$

where β is the solid angle of the vertex and A the area associated with the vertex (1/3 the area of each of its triangles). It can also be shown that β is 2π minus the sum of the vertex angles of its triangles (Figure 2.12).

In dealing with surface shape, extrinsic (geometric) curvature measures will be more useful than the intrinsic Gaussian curvature. Turk[Tur92] estimates the radius of curvature at a node by first estimating a normal at the node (face averaging) and then for each neighbor computing the radius of the sphere tangent to the line connecting the point and neighbor, and having its center on the normal ray from the point. The minimum of these radii is taken as the minimum radius of curvature at the node. Delingette, *et al.*[DWS93] fit a sphere to a tetrahedral neighborhood (all nodes are degree 3 in their meshing scheme), and claim to estimate mean curvature as a function of various relationships between the neighborhood tetrahedron and this sphere. Koenderink[Koe90] points out that the mean curvature for polyhedral surfaces can be determined exactly: just as Gaussian curvature is concentrated at the vertices, mean curvature is concentrated in the dihedral angles of the facet edges.

Estimates of smooth surface behavior at a mesh point are possible through the use of parametric fitting. Sander and Zucker[SZ90] and later Hamann[Ham93] present similar parametric fitting schemes for approximating principal curvatures at mesh nodes (and from them the gamut of intrinsic and geometric curvature measures). First, the mesh neighborhood is projected against an estimated surface normal at the neighborhood center (Sander and Zucker obtain this normal as the gradient of 3D volume data, Hamann by averaging triangle face normals). This yields a local parameterization for the neighborhood points, and after expressing a mesh neighborhood as a height-field with respect to this tangent plane a bivariate quadratic function is fit (essentially as in Lawson[Law77]). Analysis of the derivatives of this fitted function yields the desired gradient and curvature information. Stokely and Wu survey a variety of related local parameterization approaches in [SW92]. A similar fitting technique will be used in Chapter 5, and further numerical details are deferred until then.

The construction of a separate parameterization in order to fit a smooth function to a mesh neighborhood seems an extra step of mathematical indirection. We experimented with fitting algebraic functions (3D quadrics) to mesh neighborhoods as a way of estimating derivatives without parameterizations, a straightforward generalization of the implicit fitting techniques described by Bookstein[Boo79] and Pratt[Pra87], which are in turn generalizations of the linear regression technique discussed above. The drawback with an algebraic fitting approach is that there seems to be no inexpensive way to build *topological* constraints into the fit that would prevent algebraic projection onto the quadric surface from scrambling the neighbors' radial order (Figure 2.13). We noticed that in the case of nearly flat neighborhoods, the fitted quadric was often a hyperboloid of two sheets, with some samples on one sheet and some on the other. The only cure for this two-sheet problem is to put a nonlinear constraint on the discriminant of the quadric, to force the fitted quadric to be closed. This leads to an expensive iterative solution procedure rather than the direct eigen-

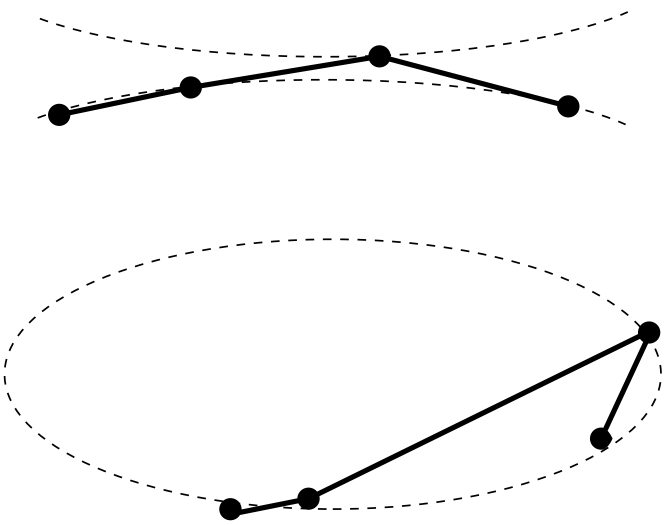

Figure 2.13: Topological pitfalls with algebraic neighborhood fitting. The dotted lines indicate the desired topology of the curve to be fitted to the given points. When fitting a general quadric, there is no reasonable way to build the desired topology into the fit, and it is quite possible to end up with topology mismatches. (Only curve fitting is demonstrated here, but the failures generalize to surfaces.)

value computation above, and makes the technique impractical to use as the neighborhood shape estimator for our interactive modeler. Nor does it address the topological ordering problem.

2.8 Computational mesh generation

An important part of our surface approximation scheme is mesh generation and maintenance — keeping mesh nodes distributed and triangulated so as to yield well-conditioned computations. We review here traditional grid generation approaches, and more recent work in surface sampling and triangulation.

2.8.1 Continuum grid generators and Laplace's equation

.

Numerical grid generation techniques were originally developed within the scientific computing community for the solution of partial differential equations over physical fields (see Thompson, *et al.*[TWM85], or the survey paper [Tho85]). Classical grid generation techniques address the problem of mapping regular (typically Cartesian) computational grids onto irregularly shaped physical domains, for use with finite-difference approximation schemes (Figure 2.14). *Grid generation* in this context really means *node placement*, since the local mesh topology is specified at the outset. The bias here is towards continuum generation schemes, involving the solution of elliptic systems of partial differential equations with respect to node positions in order to smoothly map the computational mesh onto the

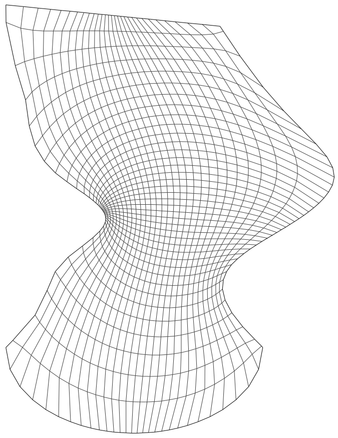

Figure 2.14: Laplace's equation has long been used to generate smooth mappings of regular computational domains to irregular physical domains (grid courtesy of Edward Luke, National Grid Project

physical domain (as opposed to the discrete sample-based approaches discussed below). Elliptic systems are preferred because of an extremum property they possess: solution extrema cannot occur in the infield, but instead occur on the boundary[MT78]. This is sufficient to guarantee a one-to-one mapping between computational domain points and physical domain points; that is, the grid will never fold back on itself.

One of the oldest and most widely used elliptic grid generators (and the one on which the mesh controller in Chapter 6 is based) is Laplace's equation, which produces extremely smooth coordinate maps (Figure 2.14). Its effect is easiest to explain for a 2D Cartesian grid, where one may consider two 2D coordinate systems: the fitted curvilinear coordinate system u, v representing the regular computational domain, and the orthogonal x, y coordinate system in which the irregularly shaped physical domain is situated. Taking for the moment the (u, v) coordinate lines as functions of (x, y), the Laplacian grid generation system is

$$u_{xx} + u_{yy} = 0, v_{xx} + v_{yy} = 0. \tag{2.27}$$

Grids satisfying these equations tend to have uniform spacing away from irregularly shaped boundaries. In fact, an easy way to derive these equations is as the solution to a variational problem asking for minimum variation in coordinate spacing over the grid, an approach introduced by Brackbill and Salzman[BS82].

The Laplace system and associated variational equations for an unstructured triangle mesh in the plane are more involved, as one no longer has a single pair of (u, v) coordinate functions covering the physical domain. This will be considered in more detail in conjunction with surface meshes in Chapter 6. We will be led to generalize a time-worn approximation to the Laplacian grid system known as *Laplacian smoothing*[Fie84], in which nodes are iteratively moved towards the centers of their mesh neighborhoods until the distribution reaches equilibrium.

2.8.2 Surface meshes: node placement

Most of the attention in continuum grid generation literature is given to grids for 2D planar domains and 3D solid domains. Much less is said about gridding surfaces embedded in 3D, and this is certainly not helped by the unpleasant differential geometry that comes into play (see, *e.g.*, Warsi[War86, WT90] for a true debauch of indices). More recently, mesh generation for unstructured surface meshes has become active topic in computational geometry, computer aided design, and computer graphics communities. The interest here is in representing free-form surface geometries in terms of polygonal meshes — either for the pure shape representation aspect, or as a computational mesh for subsequent finite-element computations. The general approach has been to first place sample points on a surface and subsequently connect them into a surface mesh.

There are a number of schemes for establishing a fixed sampling of a surface, including random placement[Cav74], spatial subdivision methods[SZL93], or incremental "Steiner" point placement as part of a mesh improvement scheme[Che93]. Such static sampling schemes are not appropriate for our interactive application, which will demand a sampling approach where point positions can vary continuously as the underlying surface changes smoothly. The near-universal time-varying approach to distributing sample points over surfaces of arbitrary topology uses the notion of *point repulsion*[RA82, Tur91, SG92b, dFGTV92, SBG93, WH94, Tur92, ST92]. Points are distributed over a physical domain by iteratively relaxing a pairwise repulsive "force" they exert on one another, inversely related to their spatial separation. At equilibrium, the points are evenly spaced over the domain. An attractive feature of this approach is the ease with which underlying features of the surface (such as curvature) can be used to control the local density of sample points and thus achieve a more efficient sampling[Tur92, SG92b].

A difficulty with using point repulsion to control a dynamic surface mesh (as opposed to just sampling a surface) is that, unlike the continuum schemes, there is currently no robust way to maintain a surface triangulation over the point set as it evolves. If the triangulation is fixed, nothing prevents the mesh from folding back on itself as points move about on the surface, and there is no penalty serving to undo such folds where they occur. Although the incremental re-triangulation techniques discussed below can go a long way towards preventing such folds from occurring, it will be seen that they are stymied when a fold is accidentally introduced. Instead, current point repulsion schemes impose a triangulation over their points *a-postiori* in a variety of ways once they reach equilibrium positions. We will consider how such surface meshes might be erected over point sets after a brief detour to discuss the special case of planar triangulations.

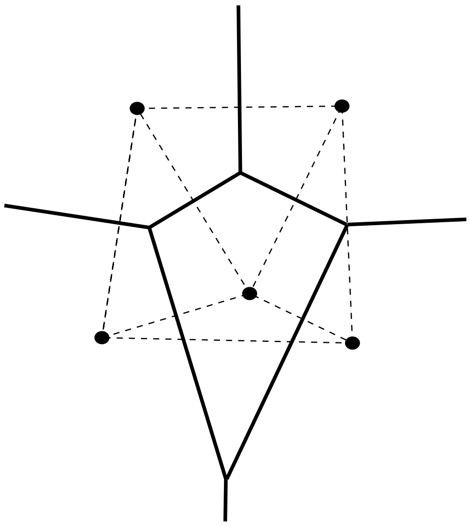

Figure 2.15: The planar Delaunay triangulation (dashed lines) is the dual of the Voronoi diagram (solid lines) of a set of points. The triangulation can be built by putting an edge between any two points whose Voronoi cells touch.

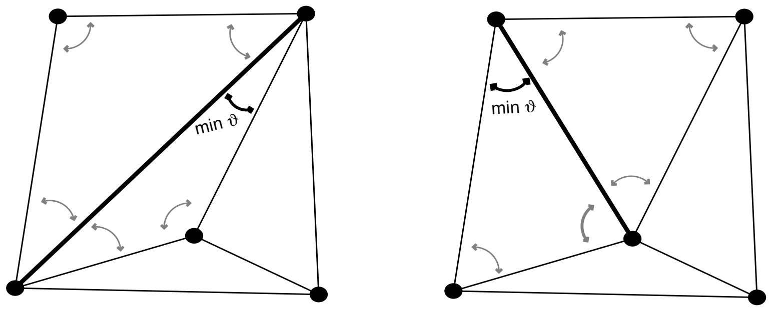

Figure 2.16: Construction of the planar Delaunay triangulation through iterative edge-flipping. The highlighted diagonal on the left is "reversed" within its quadrilateral. Flipping the edge reduces the maximum included angle, and restores the DT (right).

2.8.3 Quality meshes in the plane: the Delaunay triangulation

The problem of constructing a triangulation over a set of points in the plane has been well-studied, and algorithms exist that yield triangulations optimizing a variety of quality measures. Bern and Eppstein's excellent survey [BE92] covers much of the work from within the computational geometry community for the planar triangulation problem. Perhaps the most celebrated results involve the *Delaunay triangulation* (DT), a triangulation having qualities that make it particularly desirable as a computational mesh over a set of planar vertices. The DT maximizes the minimum included angle over the triangulation, and thus eliminates skinny triangles whenever possible, which improves the conditioning of computations over the mesh. Because of its dual relationship with the Voronoi diagram[For92] (Figure 2.15), the neighbor relations assigned by the DT yield an even partitioning of the plane in terms of nearest neighbor distances and relative triangle areas. There are a number of ways of constructing the (unique) DT over a set of points. One that will play a role in Chapter 6 takes advantage of the DT's max/min angle property to incrementally restore a DT from some sub-optimal triangulation through a series of edge-flips[Law77] (Figure 2.16). We will consider a generalization of this algorithm to polyhedral surfaces.

Though our ultimate interest is in a general surface triangulation, there is some related mesh generation work for the special case where a surface is defined as a height-field with respect to a parameter plane, or more generally, as a vector function that refers to a global u, v parameterization. It is possible to apply the planar DT algorithm to the surface points' u, v coordinates (that is, triangulate the parameterization rather than the surface itself). As an example, in the surface meshing scheme of Shimada and Gossard[SG92b, SBG93] (see also Fang and Gossard[FG92]) a point repulsion relaxation is followed by construction of a DT within the parameter plane. The price of simplicity here is that such an approach is not applicable to surfaces of arbitrary topological type, and the quality of the triangulation as measured in 3D will not necessarily be good just because the u, v triangulation is good.

2.8.4 Unstructured surface meshes: triangulating 3D point sets

As thoroughly solved as the planar triangulation problem may be, the situation is not nearly so rosy for general surface triangulations. The problem of taking a set of 3D points representing a sampling of some surface, and returning a triangulation of that surface is under-specified: unlike the planar case, a unique topology for the surface is not determined by the points, nor can a specific topology generally be imposed on a triangulation process (beyond the spherical topology recovered by a convex hull construction). This aspect of surface triangulation is pointed up by Edelsbrunner's *alpha shapes*[EM94], a generalization of the convex hull construction that enforces a maximum allowed edge-length (the so-called α parameter). For $\alpha = \infty$, the alpha shape is just the convex hull. For $\alpha = 0$, the alpha shape is the point set itself. Values in between allow the hull to "shrink-wrap" the point set ever tighter as the parameter decreases, resolving concavities and associated fine detail, even allowing enclosed volumes to split into disconnected pieces. A wide variety of surface topologies may systematically recovered from the *same* point set by varying this parameter. It should be noted that alpha shapes are tetrahedralizations of the space in and around a 3D point set, rather than a direct surface construction; there is no way to force all the given points to lie on the surface of the constructed alpha shape. Because of these topological shortcomings, alpha shapes are not a very good tool for surface reconstruction.

Szeliski, *et al.*[STT93] impose an *a posteriori* triangulation on their oriented particle systems by generalizing the empty circumcircle definition of the planar Delaunay triangulation[BE92] They look at triangle circumspheres — the smallest sphere containing the three vertices of a given triangle, and only include the triangle in the mesh if the circumsphere is empty. Circumcircles on curved surfaces will be taken up in Chapter 6. For now, it is enough to say that this "smallest sphere" test yields inconsistent results in highly curved, irregularly sampled neighborhoods, and thus cannot be relied upon to associate a unique triangulation (or surface topology) with a given point set. Chew[Che93] has developed a provably consistent generalization of the Delaunay triangulation to surfaces[Che93] that also relies on a local flatness assumption, also to be discussed further. It is not appropriate here because it requires a valid initial surface triangulation, which it iteratively improves until the surface DT definition has been satisfied. Finally, Szeliski, *et al.*'s construction includes a maximum edge length parameter, analogous to that of alpha shapes. It allows tears and holes to appear in the mesh as points move farther apart, which is a feature in their particle-based approach to recovering surface topology from unstructured CT or range data, but would be inappropriate in a modeling system that must maintain fidelity to a particular surface topology.

Hoppe, *et al.*'s surface reconstruction scheme[HDD+92] may be the most successful to date at moving from an unorganized 3D surface point set to a triangulated surface. But it doesn't actually triangulate the given point set; rather, it uses the points to fit a collection of tangent planes, constructs a signed surface distance function from their union, and then triangulates a contour of this distance function.

What one may take away from all of this is that, if it is important to maintain a particular surface topology as recorded in a mesh, it is not safe to discard the triangulation and then re-triangulate starting from the point set alone. The next section considers "topology-safe" schemes for re-triangulating a point set given an initial triangulation.

2.8.5 Unstructured surface meshes: transformation and optimization

A number of schemes for re-triangulating a given surface mesh in a topology-safe manner have been put forward. Turk's polygonal surface re-sampling scheme begins with a polygonized surface, scatters points over the surface, and re-triangulates each of the polygons to include the new scattered points in the polygonization. Nodes are iteratively deleted from the mesh, and each time the affected neighborhood is re-triangulated by projecting it onto a plane. This is not a topologically safe thing to do, especially in areas of high curvature; for this reason, topological consistency checks are performed on the resulting triangulation and the deletion is undone if it leads to inconsistent results. Similar projections and consistency checks are involved in Schroeder, *et al.*'s mesh decimation scheme[SZL93].

Much more satisfying are mesh transformation operations that do not rely on such "project-check-and-reject" tests, but instead are guaranteed to preserve the global surface topology represented by the mesh. This idea goes back to the topological work of Alexander[Ale30], who defined the mesh transforms illustrated in Figure 2.17. He shows that the order-1 moves (edge splitting and its inverse) are sufficient to transform between any two triangulations of a surface. More recently, Hoppe, *et al.*[HDD+93] made one of the first principled applications of mesh transformations to the problem of surface mesh re-sampling and optimization, in the course of reconstructing a surface from scattered 3D points. They choose sequences of moves that improve the efficiency with which a mesh represents a given target shape by splitting triangles in areas of high curvature and merging triangles in flatter areas. In Chapters 4 and 6 we will need topology-safe ways of performing various bits of mesh surgery — adding and deleting nodes, and inserting p.l. curves into a mesh. The Alexander moves provide basic transformation primitives, and mesh operations will be built in terms of these.