Geometric Modeling Using High-order Derivatives

by

Elif Tosun

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
September, 2008

Denis Zorin

To my parents, Nilgün and Yılmaz Tosun

Acknowledgments

It has been a strenuous journey surviving these years of graduate school and putting this dissertation together. It would have been absolutely impossible to manage, if not for the help and support of many people.

I am most grateful to my advisor Denis Zorin, for the invaluable support and guidance he has given me. It was his sense of perfection and scientific ardor that drove this research to the point it is at now. He has been an indispensable source of advice not only in the technical sense but also in my professional development, for which I am greatly indebted.

I am also thankful to my readers Ken Perlin and Olga Sorkine, for reading this rather long dissertation and giving timely, constructive feedback. I am grateful for their patience and understanding, especially in the last few weeks before my defense. I would like to thank Chee Yap and Rob Fergus for being valuable members of my defense committee and encouraging and supporting me along the way. Eitan Grinspun and Demetri Terzopoulos helped me greatly in the starting phase of my dissertation, for which I am also very grateful.

I am in great debt of my undergraduate advisor Ileana Streinu for her encouragement, support and guidance in every single step of the way. I am thankful for all that she has done for me while I was her student and for her continuing support and guidance after my graduation.

I would like to thank Marsha Berger, Sana‘ Odeh, Margaret Wright and all members of WinC for being such avid supporters of women in computer science, and for supporting me at various times when I needed it. I would also like to thank Rosemary Amico and Anina Karmen and the rest of the administrative staff for being such valuable resources for all students.

I could not have survived even one year of graduate school without my friends, old and new. My thanks go to Robb Bifano, Rebecca Davidson, Ayse Naz Erkan, Aslı Ertekin, Yotam Gingold, Matthew Grimes, Raia Hadsell, Jeff Han, Meghan Hartley, Koray Kavukçuoğlu, Başak Kocaman, Denis Kovacz, Harper Langston, Audrey Lee, Alyssa Lees, Evgueni Parilov, Chris Poultney, Marc'Aurelio Ranzato, Jason Reisman, Ilya Rosenberg, Adrian Secord, Ben Wellington, Chris Wu, Lexing Ying, and Anna Zaks. I cannot thank them enough for keeping me sane.

I am greatly indebted to Kush Sharma, for being with me and bearing with me throughout this journey. His care and encouragement helped me get through the toughest times. He has been a much appreciated source of diversion, when I needed it the most.

Finally, I would like to thank my parents Nilgün and Yılmaz Tosun, and my brother Umut Tosun, whose loving support and endless encouragement got me through this endeavor. Despite the distance between us, I have felt them take each and every step with me, by my side, which gave me the will and the power to go on. Without them, this would not have been possible.

Abstract

Modeling of high quality surfaces is the core of geometric modeling. Such models are used in many computer-aided design and computer graphics applications. Irregular behavior of higher-order differential parameters of the surface (e.g. curvature variation) may lead to aesthetic or physical imperfections. In this work, we consider methods for constructing surfaces with high degree of smoothness.

One direction is based on a manifold-based surface definition which ensures well-defined high-order derivatives that can be explicitly computed at any point. We extend previously proposed manifold-based construction to surfaces with piecewise-smooth boundary. We show that growth of derivative magnitudes with order is a general property of constructions with locally supported basis functions, derive a lower-bound for derivative growth and numerically study flexibility of resulting surfaces at arbitrary points.

An alternative direction to using high-order surfaces is to define an approximation to high-order quantities for meshes, with high-order surface implicit. These approximations do not necessarily converge point-wise, but can nevertheless be successfully used to solve surface optimization problems. Even though fourth-order problems are commonly solved to obtain high-quality surfaces, in many cases, these formulations may lead to reflection line and curvature discontinuities. We consider two approaches to further increasing control over surface properties.

The first approach is to consider data-dependent functionals leading to fourth-order problems but with explicit control over desired surface properties. Our fourth-order functional is based on reflection line behavior. Reflection lines are commonly used for surface interrogation and high-quality reflection line patterns are well-correlated with high-quality surface appearance. We demonstrate how these can be discretized and optimized accurately and efficiently on general meshes.

A more direct approach is to consider a polyharmonic function on a mesh, such as the fourth-order biharmonic or the sixth-order triharmonic. These equations can be thought of as linearizations of curvature and curvature variation Euler-Lagrange equations respectively. We present a novel discretization for both problems based on the mixed finite element framework and a regularization technique for solving the resulting, highly ill-conditioned systems. We show that this method, compared to more ad-hoc discretizations, has higher degree of mesh independence and yields surfaces of better quality.

Table of Contents

List of Figures

List of Tables

Introduction

High-order PDEs (fourth- and sixth-order in particular) arise in many geometric modeling operations requiring surface optimization: blending, hole-filling, curve network interpolation and interactive surface editing.

Blending algorithms construct a smooth surface connecting given surfaces with prescribed boundary conditions such that the connecting surface blends smoothly with existing ones. Hole-filling is similar, where given a hole as described by its boundary, the problem can be formulated as constructing a surface that satisfies the boundary conditions (See for example, [11, 35, 189, 190]). Curve network interpolation computes a surface that interpolates a curve network defining the surface at a higher level, while satisfying the boundary conditions on the curves. Some solutions to this problem can be found in [29, 124, 131, 154]. Interactive surface editing provides tools to control a model by applying or modifying constraints such as those tools constructed in [18, 108, 157]. In all above cases the order of the PDE determines the number of derivatives specified on the boundary. This in turn affects the quality of the joints and the resulting surfaces. For example, a second-order PDE would lead to G^0 boundary conditions, whereas a sixth-order PDE would result in G^2 boundary conditions. This is the main motivation for using high-order PDEs for high quality surfaces. G^1 and G^2 boundary conditions are especially important in design of surfaces of cars, ships and aircraft, since the aesthetic and physical properties play an important role in these applications.

In many cases, these PDEs are derived from functionals involving second- and third-order quantities (for example, curvature and curvature variation) in a variational setting. However, in many other cases geometric PDEs not derived from functional optimization can yield equally good results but with less complex formulation. [18, 157] are two such examples. A more complete picture can be found in Section 1.2.

Several formulations may be used in the discretization of high-order functionals. In one direction, sufficiently high-order (C^2 or C^3) basis functions are used to represent the surface. In this case, all higher-order derivatives can be computed point-wise. An advantage of this is the need for fewer discretization points to describe a smooth surface, similar to the limited number of control points needed to specify a patch. This way, one can avoid the need for highly refined meshes for the smooth surface approximation. Another advantage is the straightforward computation of the derivatives of functionals, as there exists a closed form describing the surface. Furthermore, convergence guarantees are easier to provide and faster convergence rates can be achieved.

We discuss a general manifold-based construction of this type of basis functions in Chapter 2 where the few discretization points are taken from the coarse input mesh. In general, however, these type of representations are relatively expensive to evaluate and difficult to construct for arbitrary topology and from piecewise-linear mesh data. Furthermore, locally supported basis functions necessarily lead to rapid growth of derivative magnitude with order. We derive the estimates for this derivative growth in Section 2.7.

The second, closely related technique is to use conforming finite elements. In this case, the quantities of interest are not defined point-wise everywhere but almost everywhere (for example, C^1 finite elements are sufficient for functionals involving curvatures, and related fourth-order PDEs). In all commonly used constructions of this type, additional degrees of freedom are introduced, typically related to derivatives. C^1 continuity requirement leads to the need for higher-order basis functions (polynomials of order 5) or splitting elements into smaller pieces.

The third alternative is to use non-conforming elements. In this case, no derivative (or even C^0) continuity is enforced on element boundaries, which allows one to use simpler basis functions, although higher than linear-order is required. In this case additional conditions dependent on the PDE being solved (”patch test” or inf-sup conditions) may require verification for the basis to work correctly. Section 1.3.3 provides an overview of finite element methods.

One can also use discrete-geometric constructions (closely related to linear finite elements) and local polynomial fits to obtain discretizations of high-order problems without introducing additional degrees of freedom. This type of constructions lacks theoretical guarantees but experimentally is observed to converge for certain types of problems. We develop this type of method for a fourth-order reflection-based functional in Chapter 3.

Finally, one can split the problem into a sequence of second-order problems by introducing additional variables and therefore can use linear conforming finite elements for discretization. This type of formulation is widely used for biharmonic and other fourth-order problems and it is referred to as mixed methods in finite elements literature. It is particularly appealing for mesh-based modeling, as it is relatively easy to implement even for complex PDEs. Moreover, it uses small-support piecewise linear basis functions only, which are particularly natural in an arbitrary mesh setting. In some approximate versions of this technique, additional variables can be eliminated to yield a system with the minimal number of degrees of freedom, closely related to the discrete-geometric approximations. On the down-side, especially in the case of sixth-order system, this formulation leads to badly conditioned linear systems which require additional effort to solve. We consider this type of a discretization in Chapter 4 and compare it to discrete-geometric constructions.

Organization and Contributions

This dissertation is organized as follows: Chapter 1 provides a detailed background on several aspects of high quality free-form surface design, such as surface quality assessment methods (Section 1.1) and variational surface design (Section 1.2). This chapter provides an detailed overview of concepts in each section, as opposed to the directly relevant previous work covered briefly in the chapters following as described below. We refer the reader to the beginning of Chapter 1 for putting the topics covered in context.

The following three chapters contain the core contributions of this work. In each chapter there exists an introduction specific to the problem and a section on previous work that reviews the most related previous work in the context of the problem solved. Moreover, each chapter independently presents the results.

Chapter 2 covers the manifold-based construction of surfaces described by sufficiently high-order basis functions. We extend previously proposed manifold-based construction to surfaces with piecewise-smooth boundary and explore trade-offs in some elements of the construction. We show that growth of derivative magnitudes with order is a general property of constructions with locally supported basis functions and derive a lower bound for derivative growth and numerically study flexibility of resulting surfaces at arbitrary points.

In Chapter 3, we present a discrete-geometric construction of a fourth-order functional based on reflection line behavior. Reflection lines are commonly used for surface interrogation and high quality reflection line patterns are well-correlated with high quality surface appearance. We demonstrate how these reflection line based functionals can be discretized and optimized accurately and efficiently on general meshes.

The mixed finite element formulation for the sixth-order triharmonic PDE is presented in Chapter 4. This PDE can be thought of as the linearized Euler-Lagrange equation for minimizing curvature variation. We present the discretization based on mixed finite elements and a regularization technique for solving resulting highly ill-conditioned systems of equations. We show that this technique, compared to more ad-hoc discretizations, has higher degree of mesh independence and yields surfaces of better quality. Finally we conclude with a summary of the problem we attempted to solve, our results and directions for future work.

1 Background

High-quality free-form surface design requires formalizing the notion of surface quality, choosing equations for surface evolution resulting in the improvement of numerical quality criteria, and efficient techniques for solving these equations. In this section we review related work on each of these aspects of free-form surface design.

Surface quality is typically evaluated using surface interrogation methods (Section 1.1). Such methods visualize and quantitatively characterize the quality of different types of characteristic lines on surfaces: isophotes, principal curvature lines and reflection lines.

These methods usually are not directly applied to *construct* or *improve* surfaces. Instead, a separate fairness functional or a flow equation is used to obtain an improved surface. Then interrogation methods are used to evaluate its quality, and, if necessary, adjust the way the surface is obtained. We review a variety of functionals in Section 1.2.1 and flow equations in Section 1.2.3 with the focus on high-order functionals and equations which are most closely related to our work.

A less commonly used technique (due to its relative complexity) is to construct an optimization functional from an interrogation method directly. We develop one such formulation in Chapter 3 based on reflection lines, one of the most commonly used surface interrogation techniques among those discussed in Section 1.1.

A variety of discretizations are used to solve for high-quality surfaces numerically. We discuss some of these in the variational setting in Section 1.2 along with corresponding functionals. High-order surface representations of the type we develop in Chapter 2 allow for most straightforward discretization of high-order

PDEs and functionals, as all necessary derivatives can be computed explicitly point-wise. However, in many applications it is essential to be able to deal with high-resolution meshes directly, both in the context of interrogation and surface construction and optimization. Related work on different techniques for discretizing high-order quantities on meshes is discussed in Section 1.3.2, with finite element techniques for related PDEs surveyed separately in Section 1.3.3. The techniques for biharmonic and triharmonic equations for surfaces we consider in Chapter 4 build on finite element methods and related discrete geometry techniques we discuss in these sections.

1.1 Surface Interrogation

Free-form curves and surfaces are used in many fields including computer graphics, scientific computing, medical imaging and industrial design. In some applications quality of the curve or surface is of extreme importance. Design of automobiles, ship hulls and aeronautical vehicles are such examples. An imperfection in curvature properties can cause dents on a car body, problems in flow behavior around a plane, or issues may arise with the NC-milling machine.

To avoid defects in the production of smooth objects, surfaces generated for these models should be analyzed for anomalies before being sent for manufacturing. The process of analyzing a surface for imperfections is called *surface interrogation.* In this section we review a number of surface interrogation methods used today. There have been many survey papers written regarding these methods in early nineteen-nineties; see [51, 69, 74, 77–79, 86, 123]. We explain the methods given in these surveys and also report on the state of the art.

We break down the many methods of surface interrogation into two main classes based on the representation of the properties. We call *dual representations* (see section 1.1.1) those that describe the surface (curve) quality in terms of another surface (curve). The other class includes those methods where the properties are displayed on the given surface which we call *on-surface representations* (see section 1.1.2). Other than these two classes there are other application-specific surface interrogation tools which are outside the scope of this work. For example, in car design there are numerous papers describing physical and computer systems that help in surface interrogation mostly on the already manufactured parts before varnishing. See [112] and [97] for some more recent directions. Some surface interrogation methods are modified versions of curve interrogation methods. In these cases we mention the basic case on the curve first and then explain the extension to surfaces.

1.1.1 Dual Representations

Plots

The aesthetic quality of a curve depends mostly on its curvature properties. One natural way of displaying the behavior of the curvature of a curve is a plot of the curvature. Such plots have been used for the definition of fair curves as in [48]: "A curve is fair if its curvature plot is continuous and consist of only a few monotone pieces".

The curvature can be plotted against the parameter of the curve or the arc length [48, 123]. As depicted in [123], the accuracy of curvature versus parameter plot depends on the uniformity of the parameterization and may result in false positives. The curvature versus arc length plot shows better results although typically it is hard to find a closed form expression for an arc-length parameterized curve. In [123], Moreton goes on to explore other plots such as radius of curvature versus arc length, derivative magnitudes versus curve parameter and so on. Also in [48] plots of logarithm of curvature are mentioned that highlight the flat areas. Curvature plots are useful in detecting flat spots (by near-zero curvature values) and points of inflection (by crossing the zero line). In [50] the authors use curvature plots to detect and fix problems related to the fairness of curves. Despite the many uses of curvature plots in evaluation of curves (and surfaces) one should pay attention to the scaling properties of the plot. As in [153], non-uniform scaling of the plot may also cause sharp corners to appear in these graphs although the surface is fair. One should follow the guidelines stated in [153] to correctly read the plots.

To extend this to surfaces, one has to find curves on the surface for plotting. This is mentioned briefly in [77] and it is noted that this method could be very efficient if the critical directions of a surface are known a priori so that the section curvature can be taken in those directions.

Polarity

Polarity is a linear transformation that maps a point onto a line or vice versa. Given a planar curve $C(t) = (x(t), y(t))$, for any t_i the point on the curve $C(t_i)$ can be mapped to a straight line by the polarity at the unit circle. For the whole range of t, the envelope of these lines forms a curve which is called the polar curve. In parametric form, the polar curve $P(t)$ is given by the following, where $\dot{x}$ stands for the first derivative:

$$P(t) = (\frac{-\dot{y}}{x\dot{y} - \dot{x}y}, \frac{\dot{x}}{x\dot{y} - \dot{x}y})$$

As given in [77, 78, 90]: "If the planar curve $C(t)$ has an inflection point at a point t_i then the polar curve $P(t)$ has a singularity at $t = t_i$."

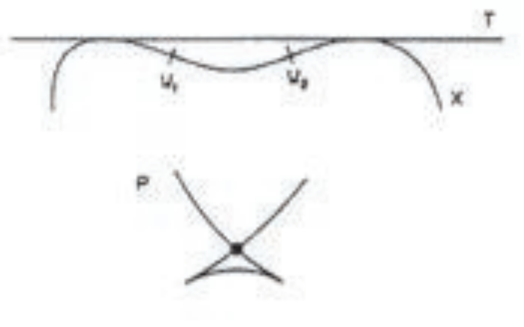

Figure 1.1: The polarity analysis of a curve T [77].

For a surface $S(u,v) = (x(u,x), y(u,v), z(u,v))$, any point (u_i, v_i) can be mapped to a plane by the polarity at the unit sphere. For all points on the surface, the envelope of these planes forms a surface, called the polar surface. In parametric form, the polar surface $P(u,v)$ is given by

$$P(u,v) = \frac{(S_v \times S_u)}{det(S, S_u, S_v)}.$$

As stated in [77, 78, 90] : "If the surface $S(u,v)$ has a root or change of sign in the Gaussian curvature at (u_i, v_i) then the polar surface has a singularity there". Note that the singularities may appear in the form of cusps, edges or dovetails [69]. Polarity for surface interrogation was introduced by Hoschek in [90]. As well as deriving these results he also describes a way of correcting such singularities detected by polarity by means of moving the control points of the curves (and surfaces).

Orthotomics

Orthotomics were introduced by Hoschek in [91] and are mostly used as a convexity test as described in [69, 74, 77, 78].

Given a planar curve $C(t)$, pick a point P such that P is not on $C(t)$ or any tangent of it. For all points of $C(t)$, the curve formed by reflecting P over the tangent at t_i is called a 2-orthotomic. In the general case, the k-orthotomic of a curve is given by

$$o(t) = P + k((C(t) - P) \cdot n(t)) \cdot n(t)$$

where n is the normal at t. Given this, $o(t)$ has a singularity at point t_i if and only if $C(t)$ has an inflection point at t_i.

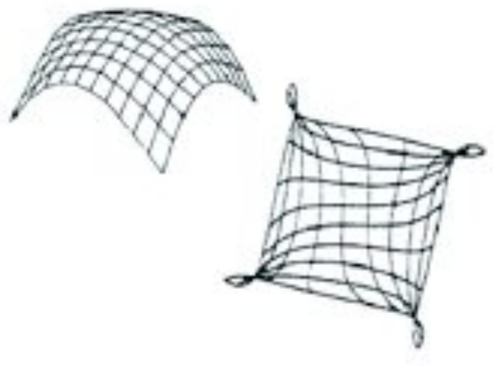

Figure 1.2: The orthotomic analysis of a bi-cubic patch [77].

Similarly, for a surface $S(u, v)$ pick a point P such that P is not on the surface or any tangent plane of it. Then the k-orthotomic of the surface with respect to P is given by:

$$o(u, v) = P + k((S(u, v) - P) \cdot n(u, v)) \cdot n(u, v).$$

Similarly, $o(u, v)$ has a singularity at point (u_i, v_i) if and only if the Gaussian curvature vanishes or changes sign at this point.

An increase in the factor k only increases the effect of the orthotomic mapping such that the singularities appear more clearly [91].

Focal Surfaces/Hedgehog Diagrams

A hedgehog diagram for a planar curve shows the normals of the curve proportional to the curvature value at some points on the curve. Although very useful for planar curves hedgehog diagrams for surfaces and even space curves are difficult to interpret [78].

Focal curves are generalizations of hedgehog diagrams. Given a curve $C(t)$, the generalized focal curve is given by

$$F(t) = C(t) + a \cdot f(\kappa(t)) \cdot n(t)$$

where $\kappa(t)$ is the curvature function, $n(t)$ is the normal and a is a scalar to aid in clearer visualization [73]. (Note that if $a = 1$ and f is the identity this is equivalent to the hedgehog diagram where instead of normals, only the loci of the points are drawn). These curves can visualize properties of the original curve such as inflection points, discontinuities, and curvature behavior [71].

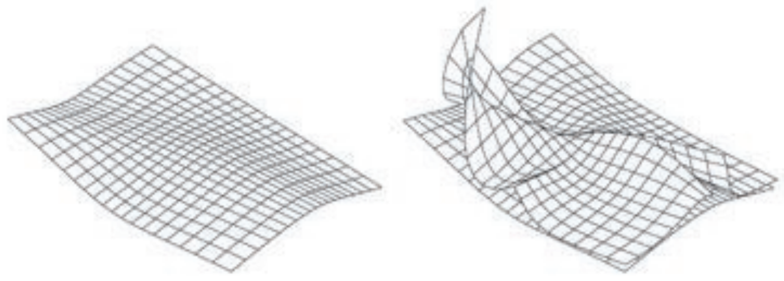

Figure 1.3: Focal analysis of four bi-cubic patches with curvature discontinuity at patch boundaries [78].

Focal surfaces were first introduced by Hagen and Hahmann [71]. Given a surface $S(u, v)$, the generalized focal surface is given by

$$F(u, v) = S(u, v) + a \cdot f(\kappa_1, \kappa_2) \cdot N(u, v).$$

For different applications, different functions f can be chosen. For example, for convexity test one can use $f = \kappa_1\kappa_2$, for flat point detection $f = \kappa_1^2+\kappa_2^2$, and so on. For examples of different functions and uses see [71] and [78]. For the continuity test one has to note that if a surface is C^r its generalized focal surface will be C^{r-2}, therefore the order of differentiation goes down by 2 [71]. Note also that the sphere is the only surface for which the focal surface may degenerate into a point and the Dupin cyclides are the only surfaces whose focal surfaces may degenerate into curves [74]. For an application of focal surfaces in automobile body design see [31].

In [180] focal surfaces are used in detection of curvature features on polygonal meshes, as well as in constructing discrete approximations to curvature based quantities. Similarly in [9] they are used in the detection of skeleton bifurcations in evolving curves and surfaces.

Discrete focal surfaces, also known as focal meshes, were recently explored. As well as the description to algorithmically construct such surfaces, a discrete shape operator based on them is provided in [198]. It was extended to be used as a geometric modeling tool for smooth surfaces in [199].

1.1.2 On-Surface Representations

Color and Texture Maps

Color maps are used to visualize functions over a surface. For fairness purposes the function visualized is a function of curvature, which is usually the Gaussian curvature, although the mean and the principal curvatures may also be used [51]. As in [69], sometimes it is best to inspect all such function plots to get a better idea of the surface. The usefulness of the map depends highly on the choice of the color scale, for if the range is too large it may not be sensitive enough to small changes in the function and if the range is too small it may be hard to interpret. In [161] a method for determining a suitable scale for a given surface is described. If one is interested in convexity test, a 3-color scale should be used [78]. This method was introduced in [43] by Dill.

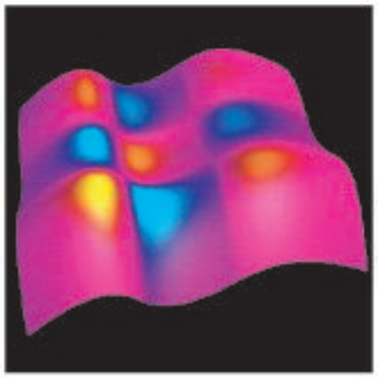

Figure 1.4: Color coding of Gaussian curvature on a surface [78].

In [105] another way of using the color map is introduced. Instead of using the regular Gaussian curvature or others mentioned above, they introduce two new measures of curvature and fairness called *shape index* and *curvedness*. Shape index specifies shape independent of size and curvedness specifies the size. Curvedness is a positive number and shape index varies in $[-1, +1]$. Due to the predefined range of this index, it is easier to assign a color scale. In [53], Forrest states that it is best to use color maps with line drawings such as contours to make the best out of this method.

Another way of visualizing shape properties of a surface is artificial texturing [160]. Instead of coloring the surface, one adds some kind of texture to the surface. In [160] the author uses polka dots which act like *flattened Hedgehog diagrams* since one can interpret the direction of the normal on a surface if there is a circle on it (See Figure 1.5 left). An overlaid Hedgehog diagram may help in ambiguities one may be faced with.

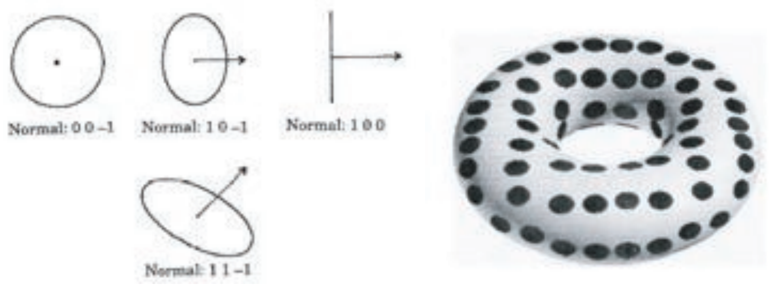

Figure 1.5: Circle projections of different normals and a textured torus [160].

Isophotes

Isophotes are lines of equal light intensity. Given a parameterized surface $S(u, v)$ and the direction L of parallel lighting then the isophote condition is given by

$$(N(u, v) \cdot L) = c = const.$$

Note that when $c = 0$ the isophotes are the same as silhouettes with respect to the light source [74]. One needs to check numerous values of c to get a good understanding of the surface. However, it is impossible to check all such values. Therefore, one may miss the irregularity on the surface if the correct constant c or the right light direction L are not used. This method was introduced by Poeschl in [142]. This initial method required that there be no flat points on the surface and it was not very reliable due to the choice of c and L. Pottmann in [143] introduced a modified isophote method which does not depend on the light direction but only visualizes curvature discontinuities on the patch boundaries. It is based on drawing at several points of the patch boundaries the maximum tangent discontinuity of an isophote instead of the entire set of isophotes. Using isophotes for checking continuity is useful since if a surface is C^r continuous then the isophotes are C^{r-1} continuous curves [74, 78, 142].

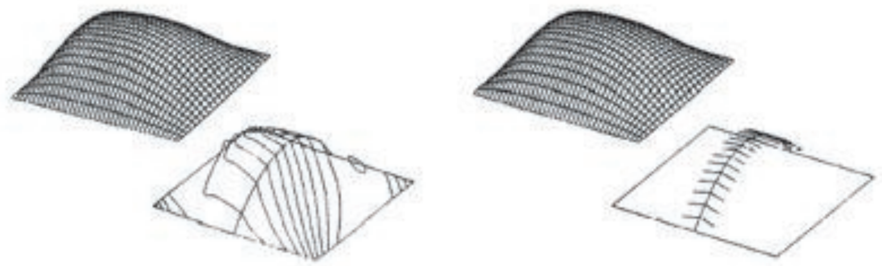

Figure 1.6: Isophote and the refined isophote method on a G^1 surface [77].

A generalization of isophotes called *lv-curves* are described in [69]. l is the direction vector to a light source and v is the direction to the viewpoint. If v is equivalent to the normal and l constant then the lv-curve becomes an isophote. For more information on lv-curves and their use in generating ir-nets we refer the reader to [69]. Another generalization of isophotes form a class that also includes reflection lines called reflection circles. This is explained in detail in [172].

Reflection and Highlight Lines

Reflection lines are a standard tool for surface interrogation in car manufacturing. It was first considered in academic literature by Klass in 1980 [100]. It visualizes dents and other imperfections as irregularities in the reflection line pattern of parallel light lines on a shiny surface. Klass also describes a way of correcting such irregularities.

A reflection line is the projection of a line L on a surface $S(u, v)$ which can be seen from viewpoint A if L is being reflected on S (See Figure 1.7). For a fixed eye point A, to find a reflection point $P = (u, v)$ the following system of non-linear equations are solved:

$$\vec{b} + \lambda\vec{a} = 2(N(u, v) \cdot \vec{b}) \cdot N(u, v)$$

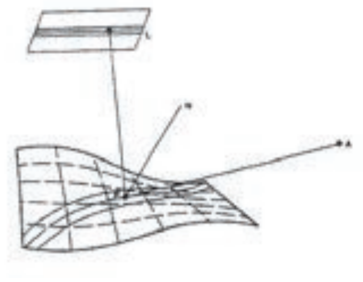

Figure 1.7: Reflection line method [77].

where $\overrightarrow{a} = P - A$, $\overrightarrow{b} = L - P$ and $\lambda = \frac{\|\overrightarrow{b}\|}{\|\overrightarrow{a}\|}$ [74, 77, 78]. The existence and un-ambiguity of a solution to this system depends on the correct choice of the eye point A [78]. Furthermore, solving such a set of nonlinear equations can get computationally expensive. For an implementation of reflection lines for car body design see [31].

A generalized version of reflection lines, where instead of parallel lines, concentric circles are reflected over a surface is called reflection circles [172]. A similar idea based on circular lines was covered in [116], where it is claimed that this formulation is superior to straight line methods since it can detect discontinuities in more than one direction. As a different point of view, in [176, 177] reflection lines are described as contour curves of a certain function in the parameter domain of the surface.

A different version of reflection lines are called *highlight lines*. A highlight line is defined as the loci of all points on the surface, where the distance between the surface normal and light line is zero. The main difference between the two is that highlight lines are independent of the viewpoint. A linear light line is given by $L = L_0 + Bt$, where B is the direction of light and L_0 is the source. An extended surface normal is defined as: $E(s) = S(u, v) + s \cdot N(u, v)$ for some scalar s.

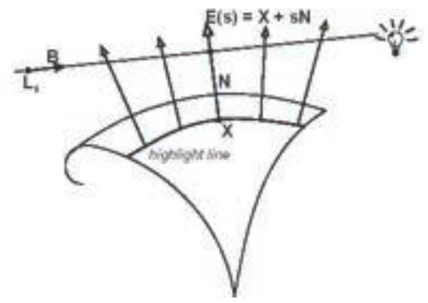

Figure 1.8: Highlight line method [78].

Given these two lines, a surface point $S(u, v)$ belongs to a highlight line if the perpendicular distance between them is zero [78]:

$$d = \frac{\|[B \times N] \cdot [L_0 - S]\|}{\|[B \times N]\|} = 0$$

Highlight lines can be extended to highlight bands if $d \leq r$ for some fixed r [78] or equivalently, if a light cylinder with radius r is used instead of a line [69].

Isolines and Characteristic Curves

Contour lines are planar lines on the surface parallel to a fixed reference plane. Equivalently, they are the intersection lines with a set of equidistant parallel lines [86]. Closed contour lines indicate maxima and minima and they only cross at saddle points [78]. [53] states that the contour lines relate directly to the surface geometry and are very useful especially if used in conjunction with color shading. Similarly in [86] it is stated that orthogonals to the contour lines are very useful when used with contour lines resulting in an orthogonal net. The main drawback of contour lines is that they are expensive to compute. In [77] a few methods are mentioned to cut down this computational effort. In [127] one can find an example of a case study which includes the level contours of the z-direction as a surface interrogation tool. Also in [111], a process used in Saab Aircraft is explained that depends on contour lines used together with intersection with planes.

Curvature contours are curves that have equal values of Gaussian, mean or principal curvatures. An irregularity in these contours points out an irregularity on the surface by means of change in the contour spacings and the directions of the contours. They are especially useful if one needs to catch small shape defects although they are sensitive to high amplitude defects as well. However, using only one type of curvature would not be sufficient in detecting all defects [127]. Parabolic lines are lines of zero Gaussian curvature, therefore are a subset of Gaussian curvature contours [78]. They divide the surface into parabolic and hyperbolic regions which may help in the convexity test, although in [127] it is noted that parabolic lines may return false positives.

Lines of curvature form an orthogonal net on the surface. They are the curves on the surface whose tangent directions are the principal directions. They indicate a directional flow for the maximum curvature across a surface [51]. Although they are useful in detection of surface defects, their computation is cumbersome and requires a numerical integration method. Also, the net of lines become singular near an umbilic ($\kappa_1 = \kappa_2$), since the principal directions are indeterminate and the integration becomes unstable [78, 86]. A few methods have been proposed to overcome this problem as described in [78] and references within. Also one should be careful while integrating lines of curvature over patch boundaries for the function of surface changes from one patch to the next [51]. A line of curvature has a continuity C^{r-1} if the surface is C^r [69].

Other characteristic lines such as geodesic paths ([51, 78]), equi-gradient lines ([86]) and fields of principal and asymptotic directions ([127]) have also been used in surface analysis.

1.1.3 Summary of Interrogation Methods

To summarize, we state what each method is designed to detect on the surface. For detection of *flat points and inflection points* plots, polarity and orthotomics may be used. Also focal surfaces where the function is defined to be $f = {\kappa_1}^2 + {\kappa_2}^2$ or $f = |\kappa_1| + |\kappa_2|$ can be used for detection of flat points. For *convexity test*, color maps, parabolic lines and focal surfaces with function $f = \kappa_1 \kappa_2$ may be used. For *discontinuities*, isophotes (which can also be used for curvature discontinuities), lines of curvature, reflection and highlight lines may be used. Focal surfaces with the function $f = {\kappa_1}^2 + {\kappa_2}^2$ may also be used for this test. Reflection lines and highlight lines are also useful in detecting global shape imperfections whereas curvature contours and plots are better in catching small shape defects. Focal surfaces may also be used for visualization of technical smoothness of a surface which is important in the milling process. In this case the function to be used is $f = \frac{{\kappa_1}^2 + {\kappa_2}^2}{\kappa_1 + \kappa_2}$. Or one may use $f = \frac{1}{R_{cutter}} - \kappa_{max}$ if the radius of the cutter (R_{cutter}) is known.

1.2 Functionals and PDEs for Surface Design

Computation of surfaces that optimize a quality measure while maintaining the constraints introduced by the designer is called variational design. In this section, we describe previous work in the field of variational surfaces, with an emphasis on the functionals. In the first section (Section 1.2.1) we describe the formulation, reviewing functionals that have been used in literature. We continue with a section on PDE surfaces (Section 1.2.2), as well as a section on surface fairing with flows (Section 1.2.3).

1.2.1 Functional Formulation

There are two popular directions to formulating functionals for variational design:

1) functionals as geometric invariants,

2) functionals as approximations to geometric invariants.

Functionals derived from geometric invariant measures result in parameterization independent fairness metrics. The parameterization dependent measures based on various derivatives of the parametric surface are presented separately as approximations to geometric invariants.

Geometric Invariants

Low-order Functionals. A set of smoothing measures are presented in [147] by Rando and Roulier. The three metrics are called flattening, rounding and rolling metrics respectively:

Flattening Metric	$K \cdot n$
Rounding Metric	$S + [H/K] \cdot n$
Rolling Metric	$(K + H^2) \cdot n$

where n is the normal and S is the surface. Fairing with respect to the flattening metric tends towards developable surfaces whereas, the rounding metric tends the surface towards a spherical shape and rolling metric leads to a cylindrical or conical shape.

There are two variations to the the *rolling metric* introduced later by the same authors [150]:

$$H \cdot n$$

$$(1/(2(2H^2 - K)) \cdot n$$

The first one has a tendency towards planar, cylindrical and conical shapes while the second one is similar but will not flatten the surface.

Minimization of bending energy leads to minimum energy curves and surfaces. Minimum energy curves (MEC) are given by (1.1) and minimum energy surfaces (MES) are given by (1.2), where the integral in (1.1) is taken over arc length and in (1.2) is taken over area. Bending energy is also referred to as the strain energy of thin plate, or for short thin plate energy. This was introduced by Moreton and Séquin in [124].

$$\int \kappa^2 dL \tag{1.1}$$

$$\int \kappa_1^2 + \kappa_2^2 dA \tag{1.2}$$

The authors extend on MEC in order to make it scale invariant in [126]. Note that MES surfaces are already scale invariant. The scale invariant MEC, referred to as SI-MEC is given by:

$$(\int dL)(\int \kappa^2 dL)$$

Due to Gauss-Bonnet Theorem for closed surfaces [92] equation (1.2) is equivalent to minimizing the mean curvature:

$$\begin{aligned}
&\int\int \kappa_1^2 + \kappa_2^2 dA \\
=&\int\int (\kappa_1 + \kappa_2)^2 - \kappa_1\kappa_2 dA \\
=&\int\int (2H)^2 - K dA \\
=&\int\int 4H^2 + C dA
\end{aligned}$$

where C is some constant, and is not used in the minimization process. Using H^2 as an energy functional is known as Willmore energy [92, 162]. Note that Willmore energy can also be computed as the area of the image of surface under the conformal Gauss map [92].

In [185] and [184] numerous fairness criteria have been studied by Westgaard and co-authors. The functionals are classified by order. For example, following second-order functionals are used.

Gaussian Curvature	$K = \kappa_1\kappa_2$
Mean Curvature	$H = (\kappa_1 + \kappa_2)/2$
Absolute Curvature	$A = \lvert\kappa_1\rvert + \lvert\kappa_2\rvert$
Total Curvature	$T = \kappa_1^2 + \kappa_2^2$

In [120], a second-order and a third-order smoothness measures are introduced by Mehlum and Tarrou. The second-order measure is the squared normal curvature integrated over all directions. These measures can also be used in assessing the quality of a given surface. This smoothness measure is given as the following, where $\kappa_n(\phi) = \kappa_1 \cos^2\phi + \kappa_2 \sin^2\phi$:

$$\lambda(u, v) = (\frac{1}{\pi}\int_0^\pi \kappa_n(\phi)^2 d\phi)^{1/2}$$

Evaluation of this integral is straightforward thanks to Euler's formula for $\kappa_n(\phi)$. This results in the following:

$$\lambda(u,v)^2 = \frac{3}{8}\kappa_1^2 + \frac{2}{8}\kappa_1\kappa_2 + \frac{3}{8}\kappa_2^2 = \frac{3}{2}H^2 - \frac{1}{2}K,$$

which is easy to compute given the coefficients of the first fundamental form.

In [130] the following quadratic form of the maximal and minimal curvatures is mentioned:

$$\int_U (C(\kappa_1^2 + \kappa_2^2) + 2D\kappa_1\kappa_2)dA,$$

which is shown to be equivalent to the functional of the clamped plate (Equation (1.15)).

High-order Functionals. Higher-order functionals are constructed when variation of curvature is minimized instead of just the curvature which lead to smoother curves and surfaces. Minimum variation curves (MVC, Equation 1.3) and surfaces (MVS, Equation 1.4) are improvements over MEC (Equation 1.1) and MES (Equation 1.2) and are described by Equations (1.3) and (1.4) respectively where e_i are principal directions.

$$\int (\frac{d\kappa}{de})^2 dL \tag{1.3}$$

$$\int (\frac{d\kappa_1}{de_1})^2 + (\frac{d\kappa_2}{de_2})^2 dA \tag{1.4}$$

Functionals described in equations (1.3) and (1.1) are described in [124] and are used in order to create minimum energy networks that are G^2. Note that sometimes, instead of κ_1 and κ_2 in equation (1.4), κ_n, the normal curvature, is used in the formulation.

In [126] the scale invariant counterparts of MVC and MVS are introduced, called

SI-MVC and SI-MVS respectively. The invariant functionals look like:

$$(\int dL)^3(\int (\frac{d\kappa}{de})^2 dL)$$
$$(\int dA)(\int (\frac{d\kappa_1}{de_1})^2 + (\frac{d\kappa_2}{de_2})^2 dA)$$

[185] and [184] studied high-order functionals as well. For third-order measures the norm of the gradient of the measures above are given.

$$K_3 = ||\nabla K||$$
$$H_3 = ||\nabla H||$$
$$A_3 = ||\nabla A||$$
$$T_3 = ||\nabla T||$$

And as the fourth-order measures variation of the variation, the Laplacians of the same measures are given.

$$K_4 = \triangle K$$
$$H_4 = \triangle H$$
$$A_4 = \triangle A$$
$$T_4 = \triangle T$$

In [120], a third-order smoothness measure is introduced. The measure is the squared variation in normal curvature integrated over all directions. Similar to the second-order version mentioned before, this measure can also be used in assessing the quality of a given surface.

Given that $\kappa_n'(\phi_d)$ is the variation of normal curvature described by an Euler-like equation (See [120], Proposition 2 for the exact formulation), the following expression is the third-order functional used:

$$(\frac{1}{\pi}\int_0^{\pi} \kappa_n'(\phi_d)^2 d\phi_d)^{1/2}$$

Once again this functional can be computed directly from the coefficients of the first and second fundamental forms. Using the description of variation of normal curvature described in this paper, one can compute the extremal values of the variation of normal curvature by solving a cubic equation. Also by introducing weight functions in the measures, one can design surfaces that are smoother in some directions than others.

The surface fairness measures in [57] are derived based on the fairness of specific curves that are used to judge the fairness of a surface such as reflection lines and planar intersection curves. The authors introduce six third-order geometric invariants for this purpose that are rational functions in the components of the first and second fundamental forms. Using some combination of H and K and this set of six invariants one can generate a fairness functional that targets the given set of curves. Actually, they show that they can re-create some functionals that were used before using their invariants. Once the fairness measure is computed it is used in the variational design of the whole surface. They show the derivation of fairness measures based on plane intersection curves, and present the results of this applied to ship hull fairing.

Approximating Geometric Invariants

The method of approximating geometric invariants with higher-order derivatives in order to cut down computation time is a method employed by many, even though the resulting surfaces are not as fair [156]. It is explained in [13] that the higher the surface derivatives used in the functional the smoother the surface is. The main problem with these formulations is the need for a local parameterization, since the second and higher-order functionals work best if a surface has an isometric parameterization, for then the approximations are exact. In the case of close to isometric parameterizations these approximations are close. We will present these functionals in increasing order.

Low-order functionals. The *first-order functionals* are based on area and minimizing this energy leads to minimal area surfaces. Note that the surface that minimizes this energy without constraints will be a single point. This functional can be written as:

$$E_{area} = \int\int_U 1dA = \int\int_U ||S_u \times S_v||dudv \tag{1.5}$$

An upper bound to this functional (equation (1.6)) is given by [61], which can lead to a linear system of equations when minimized.

$$\int\int_U \nabla S \cdot \nabla S dudv = \int\int_U (\nabla S)^2 dudv \tag{1.6}$$

This approximation can also be formulated as in [185] and [184].

$$\int\int_U S_u^2 + S_v^2 dudv$$

Greiner et al. also introduced the idea of data-dependent functionals in [62]. In this case, there is a reference surface G that is close to the desired surface where the functional formulations depend on this surface, thus the name "data-dependent". A concise summary of this method is given in [61]. This area-based energy, given as a data dependent approximation can be written as:

$$\int_U (\nabla_G S \cdot \nabla_G S)_S dA_S$$

which is equivalent to the following as given in [183]

$$\int_U trace(I_G^{-1} I_S) dA_S$$

where I_* is the first fundamental form at surface $*$.

The *second-order functionals* are those where the maximum order of derivatives is two. These functionals are used when an interactive design scheme is sought, for minimizing them can be reduced to a linear system of equations that can be solved very efficiently. Such interactive systems include [62, 95, 182].

In [184] and [185] the following second-order measures have been presented

$$\int\int_U S_{uu}^2 dudv \tag{1.7}$$

$$\int\int_U S_{vv}^2 dudv \tag{1.8}$$

$$\int\int_U (S_{uu}^2 + S_{vv}^2) dudv \tag{1.9}$$

$$\int\int_U S_{uu}^2 + 2S_{uv}^2 + S_{vv}^2 dudv \tag{1.10}$$

Note that any fairing functional that has derivatives in one direction only may be used for anisotropic fairing which is most useful in elongated shapes such as a ship hull. For example, the first two functionals above have been used in [94] for this purpose. However, note that fairing in the wrong direction or sometimes using an isotropic functional may sometimes alter the original shape of the surface.

Equation (1.10) is the most popular second-order functional, which is the approximation of the thin plate energy given in Equation (1.2). Minimization of this functional results in minimum curvature surfaces. In [59] another way of describing this measure was given by Greiner which uses the Laplace-Beltrami operator:

$$\int\int_U \triangle S \cdot \triangle S dudv \tag{1.11}$$

Greiner introduced a data-dependent approximation of this functional as well, that involves the Hessian of the surface S computed at the reference surface G:

$$\int_U \sum_{i=1}^{3} trace(Hess_G(S_i))^2 dudv$$

where the components S_i of surface are defined as in equation (1.12).

$$S(u,v) = (S_1(u,v), S_2(u,v), S_3(u,v)) \tag{1.12}$$

A slightly different version of thin plate energy is derived from the squared Frobenius norm of the matrix describing the second fundamental form of a surface, since that is equivalent to $\kappa_1^2 + \kappa_2^2$. In this derivation, the energy functional involves projections of derivatives onto the normals [181]:

$$\int_U ((S_{uu} \cdot n)^2 + 2(S_{uv} \cdot n)^2 + (S_{vv} \cdot n)^2) dudv \tag{1.13}$$

In [13], there is a mention of a series of second-order functionals of the form:

$$\int \int_U (S_{uu})^n + (S_{vv})^n dudv \tag{1.14}$$

Note that when $n = 2$, this functional is equal to (1.9).

In [75] the so called "functional of the clamped plate" is explored. This functional is given by:

$$\int_U A(S_{uu} + S_{vv})^2 - B(S_{uu}S_{vv} - S_{uv}^2) dudv \tag{1.15}$$

Since this is a physical energy, it can be used for physically based modification of surfaces. The same was also mentioned in [134] in the derivation of the approximation of thin plate energy (Equation (1.10)). The two energies are equivalent when $A = 1$ and $B = 2$. This simplification of the clamped plate to thin plate was also used later by Fasshauser and Schumaker in [52].

In [130] the constants are given as $A = \frac{E}{1-\nu^2}$ and $B = \frac{2E}{1+\nu}$. The constant E is the modulus of elasticity and ν is Poisson's ratio and they can be tuned to reflect different physical materials, such as lead, glass, and steel.

In [182] the objective function is based on first and second fundamental forms (G and B respectively):

$$\int_U (||G||_\alpha^2 + ||B||_\beta^2) dudv$$

which when simplified based on derivatives becomes:

$$\int_U \alpha({S_u}^2 + 2S_uS_v + {S_v}^2) + \beta({S_{uu}}^2 + 2{S_{uv}}^2 + S_{vv}^2) dudv \tag{1.16}$$

One can also generate functionals based on the deviation from a given surface. These we call *deviation based* functionals. Some examples of such functionals can be found in [184]. Suppose one starts with an initial surface $\bar{S}$ and would like to get to the improved surface S. We will provide a second-order example, however they can be generated to be of any order.

$$\int\int_U ((\bar{S}_{uu} - S_{uu})^2 + (\bar{S}_{vv} - S_{vv})^2) dudv$$

Following this example one can generate a "deviation" version of any of the functionals listed in this section. The same can be done by using Greiner's equations (equations (1.6), (1.11), (1.18)) to generate parameterization-independent deviation measures as such. For example, the second-order deviation measure can be written as [184]:

$$\int\int_U \triangle_{\bar{S}}^2 (\bar{S} - S) dudv$$

Another deviation based objective function is the one used in [136] based on the distance of a point from the center of curvature. Given the center of curvature

$C(u,v) = S(u,v) + \frac{1}{\sqrt{K(u,v)}} N(u,v)$, the objective function to be maximized (in order to prevent numeric errors, the functional has been inverted) is:

$$\int_U \min\{\frac{1}{D}, d\} \times \sqrt{G} dudv$$

where D is the distance function from the center of curvature, d is just a variable in order to avoid divergence issues and G is the determinant of the first fundamental matrix of the surface.

One can also use *combinations* of all the functionals listed here. For example, the functional in [95] has two terms. One of them is for smoothing and the other for deviation.

$$\int_U {S_{uu}}^2 + 2{S_{uv}}^2 + S_{vv}^2 + (S - S_0)^2 dudv$$

In [183] the energy to be minimized is the sum of internal and external energies. The internal energy can be written as:

$$E_{int} = \alpha E_{area} + (1-\alpha) E_{bend}, \tag{1.17}$$

where α is some value in $[0,1]$ and area and bending energies are as covered before. The exterior energy is based on forces that have an intuitive deformation effect on the surface. They can be attractors (attract the surface to a point or plane) or repellers that are usually based in minimizing some norm of differences, i.e. some deviation measure.

In [29], the deformable model energy (Equation 1.17) is minimized where the coefficients need not add to one. In this case both coefficients are defined as second-order tensors such that anisotropic behavior can be generated by varying the values of these matrices.

In [61], a table that summarizes the most commonly used geometric functionals and their parametric and data-dependent approximations has been provided, which

Name	Geometric	Parametric	Data-dependent
Area	$\int_U 1dA$	$\frac{1}{2}\int_U [\nabla S]^2$	$\frac{1}{2}\int_U (\nabla_G S \cdot \nabla_G S)_S dA_S$
H	$\frac{1}{4}\int_U (\kappa_1 + \kappa_2)^2 dA$	$\frac{1}{4}\int_U (\triangle S)^2$	$\frac{1}{4}\int_U (\triangle_G S)^2 dA_S$
K	$\int_U \kappa_1 \kappa_2 dA$	$\frac{1}{4}\int_U (S_{uu} \cdot S_{vv}) - S_{uv}^2$	$\int_U det(Hess_G(S))dA_S$
T	$\int_U \kappa_1^2 + \kappa_2^2 dA$	$\frac{1}{4}S_{uu}^2 + 2S_{uv}^2 + S_{vv}^2$	$\int_U trace[(Hess_G(S))^2]dA_S$
∇ H	$\int_U [\nabla_S(\frac{\kappa_1+\kappa_2}{2})]^2$	$\int_U [\nabla(\triangle S)]^2$	$\int_U (\nabla_G(\triangle_G S) \cdot \nabla_G(\triangle_G S))dA_S$

Table 1.1: Commonly used functionals and their approximations [61].

we will present here as well to provide a big picture.

High-order Functionals. There are several ways of constructing high-order functionals. In [13] one construction is provided:

$$\int\int_U ||\frac{\partial^n S(u,v)}{\partial u^n}||^2 + ||\frac{\partial^n S(u,v)}{\partial v^n}||^2$$

where the behavior of the functional for different values of n is studied. In [70] and [72] $n = 3$ is used in order to approximate curvature variation (Equation (1.4)). Greiner has another way of describing this functional based on the gradient of the Laplacian:

$$\int\int_U \nabla(\triangle S) \cdot \nabla(\triangle S) dudv \tag{1.18}$$

In [61] three methods for generating higher-order functionals are provided. We will list them here:

1. Functionals based on Sobolev norm of order m:

$$\sum_{k=0}^{m} ||\frac{\partial^m S}{\partial u^k \partial v^{(m-k)}}||^2$$

These are not invariant under rotation.

2. Functionals based on Frobenius norm of the m^{th} derivative:

$$\sum_{k=0}^{m} \begin{pmatrix} m \\ k \end{pmatrix} \|\frac{\partial^m S}{\partial u^k \partial v^{(m-k)}}\|^2 \tag{1.19}$$

3. Iterating gradient and divergence

Applying the gradient and divergence in alternating order to a function allows generation of derivatives of arbitrary order. For example, a third-order grad-div functional may be:

$$(S_{uuu} + S_{uvv})^2 + (S_{uuv} + S_{vvv})^2$$

In [185] and [184], the following third-order,

$$\int\int_U S_{uuu}^2 dudv$$
$$\int\int_U S_{vvv}^2 dudv$$
$$\int\int_U S_{uuu}^2 + S_{vvv}^2 dudv$$
$$\int\int_U S_{uuu}^2 + 3S_{uuv}^2 + 3S_{uvv}^2 + S_{vvv}^2 dudv$$

and fourth-order measures are provided:

$$\int\int_U S_{uuuu}^2 dudv$$
$$\int\int_U S_{vvvv}^2 dudv$$
$$\int\int_U S_{uuuu}^2 + S_{vvvv}^2 dudv$$
$$\int\int_U S_{uuuu}^2 + 4S_{uuuv}^2 + 6S_{uuvv}^2 + 4S_{uvvv}^2 + S_{vvvv}^2 dudv$$

Note that again, the first two functionals provided in each order can be used for anisotropic (directional) fairing. Also note that the last functional in each set can be derived from the second method given above (1.19).

Another high-order measure used in [179] is the following:

$$\int\int_U (c_1 S_{uv}^2 + c_2 S_{uuv}^2 + c_3 S_{uvv}^2 + c_4 S_{uuvv}) du dv \tag{1.20}$$

1.2.2 PDE Surfaces

In this section we review work in a direction that is closely related to variational surfaces, but does not fit into the functional classification. Note that the surfaces resulting from flows used in fairing are effectively PDE surfaces but those will be reviewed separately in the next section.

Given a functional in the form of

$$\int\int_U f(u, v, S, S_u, S_v, S_{uu}, ...) du dv,$$

the solution to the minimization problem can be characterized as an Euler-Lagrange equation which is a partial differential equation. Solving this partial differential equation is equivalent to solving the minimization problem. A PDE is characterized by its prescribed boundary conditions. There are numerous methods to solve PDEs based on finite differences or finite elements (See Section 1.3.3 for more details on the finite element methods). The appealing property of this formulation is the fact that one does not have to give up geometric invariance in order to reduce computing time.

Some examples of Euler-Lagrange form of well-known functionals are given in [61] and replicated in the following table.

A C^k-Dirichlet condition means that the position and derivatives up to order k need to be specified. The fact that all second and all third-order functionals are equivalent to the same Euler-Lagrange equation can be explained as follows: These quantities differ by an integral on the boundary. Then, if the boundaries

Order	Functional	Constraints	Euler-Lagrange Equation
1	$\int_U (\nabla S)^2 dudv$	C^0 Dirichlet	$\triangle S = 0$
2	$\int_U (\triangle S)^2 dudv$	C^1 Dirichlet	$\triangle(\triangle S) = 0$
2	$\int_U S_{uu}^2 + 2S_{uv}^2 + S_{vv}^2 dudv$	C^1 Dirichlet	$\triangle(\triangle S) = 0$
3	$\int_U \nabla(div(\nabla S))^2 dudv$	C^2 Dirichlet	$\triangle(\triangle(\triangle S)) = 0$
3	$\int_U S_{uuu}^2 + 3S_{uuv}^2 + 3S_{uvv}^2 + S_{vvv}^2 dudv$	C^2 Dirichlet	$\triangle(\triangle(\triangle S)) = 0$

Table 1.2: Euler-Lagrange form of some well-known functionals [61].

are fixed, all k^{th}-order functionals return the same surface, and if the boundary conditions are different then so are the results.

Another PDE-based technique is described in [156]. In this paper the authors use

$$\kappa'' = 0$$

for curves. A curve that consists of parts that satisfy this equation is called a clothoid spline. They extend this idea to space curves and closed surfaces. In the case of closed surfaces, the functional used is the mean curvature minimization. Another extension of this to surfaces with regularity constraints, namely for surfaces with subdivision connectivity, is given in [155]. In this case the fourth-order PDE solved is given by:

$$\triangle H = 0 \tag{1.21}$$

The authors overcome the regularity constraint in [157] and apply the PDE to general meshes. The trick here is to use the discretized Laplace-Beltrami operator as opposed to applying a quadratic local fitting near a vertex in order to compute differential quantities. They also avoid the noise-free initial mesh constraint by applying the method described in [158] to the initial mesh to clear noise.

In these works computation of high-order derivatives is avoided by factorizing this PDE into a set of two nested second-order problems. Although this procedure may seem similar to the mixed method we utilize, it is in fact quite different. The second-order problems resulting from the factorization here will need to be solved sequentially: one to compute mean curvature values at vertices, the other to compute vertex positions given these mean curvature values. The mixed method leads to a system solving both second-order problems in one system. results in both second-order problems being solved in one system. Furthermore, the fourth-order problem we are solving is linear whereas the one considered here is non-linear.

In [45, 46] a finite difference approximation to fourth-order PDEs is applied in an interactive tool set. Unlike other PDE based methods, this involves changing constraints not only on the boundary, but in the interior as well in terms of point, region, normal or curvature modifications.

It is worth pointing out that there is a large amount of literature regarding PDE Surfaces that is not based on approximations, but rather on derivations of analytical solutions for the driving PDEs. We will briefly point out few examples here without going into details. In [11, 12, 60, 174, 175, 203], fourth-order problems are considered where closed form solutions are computed. One can observe the shift to interactive applications. Some work involved sixth-order closed form PDEs [197, 204, 205]. In [108], the triharmonic equation is presented in an interactive setting where the boundary conditions are prescribed through a set of modifiable curves near each boundary.

1.2.3 Surface Fairing and High-order Flows

Algorithms for smoothing a given initial surface smoother are studied under surface fairing. The idea is to smooth out noise or other artifacts on the given surface while maintaining geometric features. There are several trends in approaching this, two of which are variational fairing and diffusion flow that are relevant for our purposes.

The variational formulation is very similar to variational design methods given in the previous section. The most common functionals used in this setting are the total curvature or the approximations of the area functional or the thin plate energy([102], [202]). Since these have been covered in detail before, in this section we will primarily consider flows.

The goal of using diffusion flow is to remove high frequency noise in meshes [157]. Flows force the surface to evolve in the direction of the gradient of a function not necessarily minimizing it all the way to the limit. This is especially useful if coming close to the minima instead of computing it exactly takes less effort. Once again flows of order up to two are considered low-order and flows of order three and higher are considered high-order.

Low-order Flows

Taubin [170] introduced the signal processing technique in 1995 which takes a combination of area and thin plate energies to provide Gaussian filtering. This method is linear in the number of vertices in time and space. It involves a discrete approximation to the Laplacian such that the eigen vectors become the frequencies of a mesh. This reduces the problem of fairing to low-pass filtering in signal processing terms. The problem with this is the uniform approximation of the Laplacian (see equation (1.24)) which results in artifacts in the case of irregular meshes. Also note that Laplacian smoothing causes shrinkage. In [41], Laplacian smoothing is reformulated as a time integration of the heat equation which allows the use of implicit integration schemes that allow large time steps. The authors also introduce discretizations of the Laplacian that are parameterization-dependent.

Furthermore a curvature flow is used instead of the umbrella operator (see equation (1.24)).

The heat diffusion equation is given by the following, where $\triangle X$ is the Laplacian of the mesh X:

$$\frac{\partial X}{\partial t} = \lambda \triangle X. \tag{1.22}$$

Note that λ in this equation is a weight factor. This factor needs to have the property that $0 < \lambda < 1$ for stability purposes. This causes small time steps (as opposed to large ones in [41]). As in [170] and [168] repeated application of such a diffusion step causes shrinkage. To avoid that Taubin proposed to use two steps with different weight factors: one positive, one negative. The reader is referred to the references for details on the choice of these weight factors.

The general approximation to the Laplacian is given by equation (1.23) where the weights w_{ij} have the property : $\sum_j w_{ij} = 1$.

$$\triangle(x_i) = \sum_j w_{ij}(x_j - x_i), \quad j \in N(i) \tag{1.23}$$

The simplest weight scheme for the general Laplacian in the case of a triangular mesh is the umbrella operator given by the following, where m is the number of neighbors.

$$\triangle(x_i) = \frac{1}{m}\sum_j (x_j - x_i), \quad j \in N(i) \tag{1.24}$$

Another direction is to use the inverse of edge lengths as weights to compensate for irregular edge lengths, as introduced by Fujiwara [171]. However, this does not solve the problem of unequal face angles.

$$\triangle(x_i) = \frac{2}{E}\sum_j \frac{x_j - x_i}{|e_{ij}|}, \quad E = \sum_j |e_{ij}|, \quad j \in N(i),$$

The discretization of the Laplacian used in [41] that works with both irregular edge lengths and unequal face angles is given by weights of the form $cot\alpha + cot\beta$ where α and β are angles opposite the edge. The equation follows:

$$\triangle(x_i) = \frac{1}{4A}\sum_j(\cot\alpha_j + \cot\beta_j)(x_j - x_i) \tag{1.25}$$

In [113], the umbrella operator is used in conjunction with extra constraints in order to avoid shrinkage. The constraints force the barycenters of triangles to stay in place at each iteration.

In [93] a fairing process that uses vertices with increased support is provided. This allows the fairing to be extended to non-manifolds. The new position of a new vertex not only depends on its own Laplacian but also the Laplacians of its neighbors with weights based on properties of the non-manifold mesh. It is claimed that most discrete Laplacians may be used for this purpose with no modifications to the framework.

Mean curvature flow, equivalent to heat diffusion as described in [41], is given by the following.

$$\frac{\partial X}{\partial t} = -Hn$$

Right hand side is discretized using the discrete Laplacian with cotangent weights given in equation (1.25), and n is the surface normal. Mean curvature flow smooths the surface by moving the vertices along the normal with speed equal to the mean curvature H. This is also called mean curvature motion [36] and is the local surface area decreasing flow.

In [137], several flow equations are derived in the following form:

$$\frac{\partial X}{\partial t} = Fn + Gt$$

where F is the speed of flow in normal direction and G is the flow in tangent direction. The authors suggest different functions for F and G mostly based on principal curvatures to produce better fairing. One can also define some threshold value in these functions such that only vertices with function values that exceed these thresholds gets faired, avoiding over-smoothing.

In [36] an anisotropic diffusion is introduced given by the following equation where D is the diffusion tensor based on the shape operator that acts on the gradient of X.

$$\frac{\partial X}{\partial t} = div(D\nabla X)$$

The use of the shape operator allows an increase or decrease in diffusion in certain directions, therefore helping in enhancing sharp features. For example, an edge corresponds to a large eigen value, where the largeness required depends on a threshold set by the user. Given this, one can reduce diffusion in the direction of the corresponding principal direction. In [37], this flow is used to fair point-based surfaces instead of meshes. Another type of anisotropic diffusion is introduced in [30] where the thresholds for feature detection is computed using Bayesian analysis.

Instead of an anisotropic diffusion tensor, one can introduce an *adaptor* into the flow equation to achieve adaptive fairing such that dense parts of the mesh gets faired less. Such adaptive diffusion was constructed in [7] where the PDE is of the form:

$$\frac{\partial X}{\partial t} = A(X)div(\nabla X) = A(X) \triangle X$$

The adaptive diffusion function $A(X)$ is a smooth positive function that characterizes the density of the surface mesh. In this paper, the choice for $A(X)$ is a quartic box spline function. Using such a function that is based on density (smaller function values at denser parts of mesh) allows the parts with high density, therefore larger curvature, to be faired less.

A homogenization function that is also adaptive to the density helps in avoiding tiny and collapsed triangles on a mesh. This function $P(X)$ causes tangential displacement near smaller triangles in order to make them bigger. Using such an homogenizer in the equation, PDE becomes:

$$\frac{\partial X}{\partial t} = A(X)div(P(X)\nabla X)$$

Also one can force approximation or even interpolation of some of the vertices of the initial mesh throughout the fairing process. All this was done in [187] where the flow equation becomes:

$$\frac{\partial X}{\partial t} = A(X)div(P(X)\nabla X) + R(X)$$

where $A(X)$ is the adaptor, $P(X)$ is the homogenization term that homogenizes the mesh while smoothing and $R(X)$ represents the approximation (or interpolation) function.

In [27] a different kind of diffusion, namely spherical diffusion is introduced. Spherical diffusion is fast and avoids shrinkage and stability problems. However, this is only applicable to star-shaped meshes where the surface can be described as a height field on a sphere. The spherical diffusion equation is given by equation (1.26) where $\triangle_{S^2}$ is the spherical Laplace operator (equation (1.27)).

$$\frac{\partial X}{\partial t} = k \triangle_{S^2} X \tag{1.26}$$

$$\triangle_{S^2} = \frac{1}{sin(\vartheta)}\frac{\partial}{\partial \vartheta}(\sin(\vartheta)\frac{\partial}{\partial \vartheta}) + \frac{1}{\sin^2(\vartheta)}\frac{\partial^2}{\partial \vartheta^2} \tag{1.27}$$

In the recent paper by Bobenko and Schröder [14], flow of Willmore energy is explored. Willmore energy of surface X is given by:

$$E_W(X) = \int_X (H^2 - K)dA = 1/4(\kappa_1 - \kappa_2)^2 dA$$

The discretization of Willmore energy at a vertex i is given by:

$$W_i = \sum_{e_{ij}} \beta^i_j - 2\pi$$

where β^i_j is the angle between circumcircles of the two adjacent triangles to edge e_{ij}. The global energy is the sum of such terms for all vertices of the mesh. The geometric flow in the direction of steepest descent of this energy (equation (1.28)) is useful in surface fairing due to the non-shrinking nature of the flow. Since the energy is scale-invariant the irregularity of the mesh does not effect the quality of fairing. This is better than the method in [137], since $H^2 - K$ is guaranteed to be greater than or equal to zero in this case. This is important since when computing principal directions from Gaussian and mean curvatures this is the term that appears under the square root.

$$\frac{\partial X}{\partial t} = -\nabla E_W(X) \tag{1.28}$$

In [202], the integral mean curvature (equation (1.40)) is approximated by a sum of squared distance measures to a set of faces of the mesh and the vertex flow is in the direction that minimizes this energy. It is referred to as the "MIN-DIST" flow. The correct choice of faces for distance computation leads to a feature-enhancing flow.

Another use of diffusion for surface fairing is to apply it on the surface normal field instead. Then the surface should evolve to match the smoothed normals. This technique was used by Tasdizen, et al. in [167] where an anisotropic diffusion flow on level set surfaces is used in the first step and the surface is re-fitted to the normals via a second-order PDE in the second step. A similar two step procedure is used in [192] for mesh smoothing. The first step is based on mean or median filtering on the normals in this case. Mean filtering consists of taking the normalized area weighted average of normals of neighboring faces of each triangle of the mesh. There are two median filters introduced in the paper: One is based on angles, the other on directional curvatures.

High-order Flows

In [157] the bi-Laplacian for diffusion is used. Equilibrium state of diffusion flow allows only C^0 boundary conditions and therefore is of limited use in surface design. It follows that using the square of the Laplacian would lead to C^1 boundary conditions and is actually equivalent to minimizing the thin plate energy given by equation (1.10).

Another type of curvature flow is given in [196] called the gradient descent flow for the elastica (or the thin-plate energy functional). The same flow is also known as the bi-Laplacian flow [137] since the right hand side of the equation is equivalent to the Laplacian squared. It is also the Euler-Lagrange equation for the Willmore energy. It is different from mean curvature flow in the speed of the flow:

$$\frac{\partial X}{\partial t} = -(\triangle H - 2H(H^2 - K))n$$

The authors also propose to add a mesh triangle equalization term to the right hand side of the equation for numerical stability. This term is basically a tangent speed component made of the tangent component of an area weighted bi-umbrella operator. The discretization of the Laplace-Beltrami operator used in this paper is the following which differs from Desbrun's approximation by the different area factor:

$$\triangle P = \frac{3}{A}\sum_{i=1}^{n}(\cot \alpha_i + \cot \beta_i)(Q_i - P_i) \tag{1.29}$$

A is the total area of neighboring triangles of vertex P, vertices Q_i are the neighbors and α and β are the angles opposing the edge between P and Q_i. The discretizations for the Gaussian and the mean curvatures are covered in the following section on Curvature Estimation (X.1.3.2).

Another type of flow is the surface diffusion flow given by the following where $\triangle$ is the Laplace-Beltrami operator, H is the mean curvature and n is the normal vector:

$$\frac{\partial X}{\partial t} = \triangle H n$$

This flow converges fast to a sphere especially if the initial surface is close to one. It is area-shrinking and volume-preserving. In [206] this flow is used for smoothing of quadrilateral and hexahedral meshes accompanied by a discretization of the Laplace-Beltrami operator for such meshes.

In the technical report by Xu, et al. [188], a higher-order flow has been explored. The equation for the flow is the following:

$$\frac{\partial X}{\partial t} = (-1)^{k+1} \triangle^k H n$$

It is claimed in the paper that this flow is volume preserving for $k \geq 2$ but it is unknown if it s area shrinking or not. The higher-order Laplace-Beltrami operator is discretized recursively as $\triangle^k = \triangle(\triangle^{k-1})$.

A fourth-order flow, namely the flow based on the Laplacian of curvature is studied in [32]. The difficulties in such a flow arise from the fact that the fourth derivative involved makes the system very sensitive to perturbations and causes very small time steps. The algorithms studied here are based on a level set method.

Two second-order, two fourth-order, and two sixth-order nonlinear, and parameterization independent flows are explored in [189], for purposes of creating blend surfaces efficiently. The solution is based on a finite-difference-like method, where differential operators are approximated using quadratic fitting. The boundary conditions are gathered from the outer mesh.

In [190], an Euler-Lagrange equation is derived from the minimization of mean curvature gradient and is applied as a sixth-order flow. The solution is similar to that in [189], based on finite differences. The high-order of the flow allows for G^2 boundaries and is demonstrated by applications to surface blending, N-sided hole-filling and point interpolating. A more general framework for surface modeling based on any PDE, where differential operators are once again approximated with quadratic fitting is provided in the follow-up work [191].

1.3 Discretization and Numerical Methods

1.3.1 Design Settings

Variational design can be used on point sets, meshes, subdivision surfaces and patch based surfaces. For all cases however, the starting point is either a mesh or a point set.

Points Sets

Given a point set as initial data, one technique generating a variational surface is to fit the point set to a **single patch**. In [13], this method is employed with a functional that consists of two parts, one for a least squares approximation of the surface and the other using higher-order derivatives for smoothing. The discretization of surface derivatives is computed using finite differences. The authors explain that the reason the high order functionals result in smoother surfaces is because the finite differences involve more points around a given point using a PDE model. They mention that equations of type (1.14) help in smoothing in diagonal directions due to the respective finite difference discretizations.

In [71] and [72], two single-patch methods are presented: one for B-spline surfaces and the other for Bézier surfaces. In both cases the functional consists of a least-squares part for approximation and a smoothing part. What differs in the two cases are the approximations of the smoothing functionals. In the case of B-splines the smoothing term is the curvature variation and the approximation involves third-order derivatives. In the Bezier case strain energy is used and it is approximated with a quadrature formula. This method avoids the initial computation of a smooth network of curves and uses point data only. The same formulation is also mentioned in [75] with one modification: one can introduce extra conditions describing the parameterization such that the parameterization can be included in the variational setting as an extra parameter. This way as the energy is minimized the parameterization gets closer to isometric.

In Westgaard's thesis [184] and paper [135], the fairness functionals described within have been tested to generate a car hood as a single patch surface. It is better to generate fair boundary curves first, therefore making this a two-step process. For if the boundary curves are not fair enough, the issues propagate to the interior of the surface. It is also concluded that second-order measures may cause flat regions and may cause convex surfaces to become locally concave. Third- and fourth-order measures perform much better.

Another strategy for dealing with point data is to define a **multi-patch** surface given the constraints for patch boundaries. In [147], a composite fair surface is generated by first defining the design constraints (may that be point or boundary constraints), and solving an optimization problem that minimizes the free parameters with respect to one of the fairness metrics given in the previous section.

These metrics have also been used in [88] for local fairing of point sets followed by a B-spline fit resulting in faired such surfaces.

In [179], the functional (1.20) is minimized on a B-spline surface given by $S(u, v) = B_u^T V B_v$ where B_u and B_v are the vectors of B-spline basis functions. It is reduced to a set of linear systems where new additional data points are added only where needed. The author claims that the parts of the surface with lack of enough data points is where undulations occur, therefore it is important to add data points where necessary.

In [182], the functional in equation (1.16) is applied to a tensor product B-Spline surface. This functional is quadratic in degrees of freedom, therefore it can be cast as a constrained least squares minimization. Only linear geometric constraints are considered in order to keep efficiency high.

Equation (1.14) with $n = 2$ is used in [201] in order to fair spline surfaces. Their algorithm is based on identifying “bad” data points by using highlight line models and moving them such that the given (strain) energy is minimized. The energy is computed within a small area near the problem data point and therefore fairing is local.

In [59], two formulations for fairing are introduced for spline surfaces where one can use either one of the two functionals given by equations (1.11) and (1.18). One is a global one-step procedure that involves a numerical quadrature based on the functional and a linear system to solve. In this case many control points get modified in one-step. The other formulation is for local fairing and is an iterative procedure. In this case, one control point at a time is changed and the functional is re-computed by numerical quadrature at each step. The procedure continues until convergence.

Wesselink [183] worked with surfaces based on patches. He worked with tensor product B-spline patches and triangular Bézier and B-spline patches. The constraints applied to surfaces are divided up into positional, directional and "other" constraints. Positional constraints can force the surface to interpolate a point, a point on the surface to interpolate a line or a plane. Directional constraints can prescribe the direction of the normal. Other constraints may include continuity or curve interpolation constraints. All constraints are linear and may be expressed in terms of the concatenation vector which is a vector of all control points. All such constraints can be gathered into a sparse system. Internal energy can be written as a quadratic expression in control points, although external energy terms may be more complicated. Overall, the energy minimization problem is reduced to a non-linear programming problem.

One can also generate a **curve network** given scattered point data. For example, given a point set of $m \times n$ data points one can generate m curves in one direction and n curves in the orthogonal direction that can lead to a curve net. Usually the curve net is then filled with patches resulting in a multi-patch surface.

In [94], a three-step method for surface fairing is used that was developed by the same author in an earlier work. This method proceeds from a faired curve net to a faired tangent strip net to a net interpolating smooth surface. The tangent strips ensure the G^1 continuity between adjacent patches. When the surface is generated, each patch is designed to interpolate the four boundary curves and four boundary tangent strips while minimizing a functional. One can also compute the tangent strips in the minimization step therefore combining the last two steps. However in this case, the minimization will be performed across the whole surface instead of patch by patch.

Another technique based on this idea is given in [125]. The surface is a result of a nonlinear optimization where G^1 continuity is maintained with the use of a penalty function. Given a set of scattered data, first a G^2 network of curves is formed based on MVC. Then patches are blended to form an MVS surface. The patches may interpolate or approximate the curve network, where the former results in fairer surfaces though takes longer computation time.

Similarly, Nowacki and Reese [134] first generate a rectangular curve net. As a first step, this mesh of curves is faired based on optimization techniques followed by the filling step where the holes between the curves are filled with patches that minimize the thin plate energy and match the constraints of neighboring patches. This formulation is used for ship hull design.

In [96], given a rectangular network of curves, only the twist vectors (partial derivatives at corners of patches) are the free variables. The values assigned to these twist vectors is such that an energy functional is minimized. The energy functional is a quadratic in the second partial derivatives of the surface. Another paper that uses tools from calculus of variation for computing the twist vectors is [49]. It is noted here that the curviness of the surface is determined by the normal component of the twist vector ($S_{uv} \cdot N$) instead of the twist itself. They minimize total curvature by minimizing the normal component of the twist vector.

Meshes

One way of generating a variational surface based on a mesh is to generate a **network of curves** based on the edges of the mesh. One example of this is given in [124]. A quintic Hermite approximation is used in generation of the curve network. The discretization of equation (1.1) in terms of the Hermite curve H(t) is given by:

$$\int_0^1 \frac{(H'(t) \times H''(t))^2}{(H'(t) \cdot H'(t))^{\frac{5}{2}}} dt$$

The functional is minimized using a nonlinear optimization procedure based on gradient descent. The starting choice is computed with heuristics, where the initial curvature at a vertex is computed based on angle defect, and the normal is the average of the incident faces.

Given a triangular mesh, in [106] a minimum norm network is formed by forming a network of curves from triangle edges that minimize the norm of the second derivative of the curve. A local parameterization is used that describes the triangulation on the tangent plane at a given vertex. Once the network is obtained, triangular Bezier patches are used to fill the holes.

Starting with a mesh, one can also replace each face with a spline patch in order to generate a **multi-patch** smooth surface. In [185] the problem of fair surface design is studied where the input is an irregular mesh. In [184] the multi-patch examples refer to this case. To avoid dealing with n-sided patches, first a step of midpoint refinement is applied. Then one of the many fairness measures given plus a deviation measure (if a tentative surface shape is known.) The constraints are the interpolation and/or approximation of mesh points and C^1 joints across patches. The remaining control points are the variables computed to satisfy the minimization requirements. This problem is solved as an equality constrained quadratic programming problem by means of Lagrange necessary conditions. In [184], a case study on the design of a ship hull based in this method is provided. In [135], the multi-patch fairing is presented without the midpoint refinement step on a “Greek fishing vessel”.

In [29], finite elements are used in order to solve the energy minimization problem. In this case triangular finite elements are used. The total energy is the sum of the contributions of each element, where the contributions include C^1 continuity constraints between triangles. The energy integrals are evaluated by Gaussian Quadrature. For more on finite elements methods see Section 1.3.3.

Similarly in [52], the setup is triangle-based. The parameter domain of the surface is a triangulation. Each triangle is associated with a polynomial interpolating spline patch. The energy per patch is the simplified thin plate energy (based on second derivatives) computed per coordinate (S_i) given the surface $S = [S_1(u,v), S_2(u,v), S_3(u,v)]$. The total energy is minimized based on a Lagrange multiplier method. The results guarantee C^1 conditions on the patch boundaries and it is straightforward to add penalty terms in order to get approximate C^2 conditions.

Another way to approach the problem in case of meshes is **global refinement**, or subdivision. In [162], an approximate cost functional is computed on the control mesh of a subdivision surface. Local energies may be vertex- or edge-based and the total energy of the mesh is the sum of all such local energies. For a bending energy discretization, an edge-based method is used where the total energy is given by equation (1.30) where β is the dihedral angle, $||e||$ is the length of the edge and h_1 and h_2 heights of the two adjacent triangles.

$$\sum \frac{\beta^2 ||e||}{||h_1|| + ||h_2||} \tag{1.30}$$

Also, instead of computing a gradient in the process of reaching the minimum, one can use a vertex-move method that aims an MVS surface. This move depends on the change in turning angle in the direction of the edge. Once the energy is minimized at the current resolution, the mesh is subdivided to produce new vertices introducing new parameters for optimization.

A class of interpolatory refinement schemes for curves is given in [101]. In every step, the newly added points are placed by solving an optimization problem. Therefore, positions of added points depend on all others, i.e. a global scheme. The energy per vertex depends on a linear combination of a set of vertices before and after it. The total energy of the curve is the sum of squares of energies of all vertices. The energy decreases at each step of refinement. Extension to surfaces is given in [102].

In [102] variational subdivision surfaces are discussed. Similar to the curve case, the subdivision schemes are interpolatory, namely positions of only the newly added points are computed by energy minimization. The main difficulty in variational subdivision schemes is the computation of the local parameterization per vertex, since the neighborhoods of each vertex change with the addition of new "refined" vertices. The solution in [102] and [104] is to use "templates" for edge and face vertices that blend the parameterizations of vertices that belong to the initial mesh. Only for extraordinary vertices an arbitrary parameterization is allowed.

The application most relevant for our purposes is **mesh smoothing**. As in [157] a mesh should have two types of fairness; outer fairness that depends on the fairness of the approximated continuous surface and inner fairness that depends on the distribution of vertices within the surface. Note that inner fairness influences outer fairness.

In [156] discretization of the clothoid spline (mean curvature minimization) is achieved by a polygon (triangulation) where the curvature is linearly distributed on each segment (triangle). The geometric invariants are computed by the first and second fundamental forms given a local parameterization near a vertex. In the surface case the process starts with a closed triangle mesh. Positions of new points and the mean curvature are recomputed using two separate formulas at each iteration thus approximating the PDE. The procedure is iterated until the properties of the discrete clothoid spline (curvature minimization) is achieved. Each vertex is moved along the surface normals.

In [157] the connection between inner and outer fairness is achieved by a discrete solution of an intrinsic PDE. This PDE, given in equation (1.21), can be discretized at a vertex as in equation (1.29) where the vertex positions P_i and Q_i are replaced by mean curvature discretizations H_i, H_j at those vertices. The inner fairness is achieved by assigning a discrete Laplacian for each interior vertex such that instead of moving a point along its normal from each iteration to the next, it is moved along a line based on the Laplacian. This is an extension of the method mentioned above with less limitations.

In [92], Willmore energy is discretized as a function of the vertices of a triangular mesh. A polyhedron has its mean curvature concentrated at the edges which means that Willmore energy will be infinite on the actual mesh. Therefore an approximation of mean curvature h_v is used that is computed at each vertex (See equation (1.43) in Curvature Discretization section). Then the approximation to Willmore energy for the whole surface is given by the following where a_v is the area element equivalent to one third of the area of the one ring neighborhood around vertex v.:

$$w = \sum_v w_v, \quad w_v = {h_v}^2 a_v$$

In [181], triangulated point sets are used as input. The point-based derivatives are computed by temporarily fitting a smooth surface to the neighborhood of a vertex. Instead of a projection-based technique, a procedure mimicking the polar geodesic parameterization is used. They use the total curvature approximation given in equation (1.13). However, it is stated that they actually minimize a variation of this where the normals n and the area element (one third the area of one ring) are taken to be constant. The authors acknowledge that this does not minimize exact total curvature, yet it still gives good results.

In [103] and [104] triangular meshes are studied. The finite difference approximation of (1.2) given by

$$\sum_{p_i} \omega_i(||\Gamma_{uu}(p_j - p_i)||^2 + 2||\Gamma_{uv}(p_j - p_i)||^2 + ||\Gamma_{vv}(p_j - p_i)||)^2 \tag{1.31}$$

where weights ω_i reflect the local area element, namely the areas of triangles and Γ_* are corresponding divided difference operators, is minimized by solving a linear system. The finite differences are taken with respect to a locally isometric parameterization. The local parameterization is the best approximating quadratic polynomial in the least squares sense. A local parameterization, therefore a set of divided difference coefficients, are computed for each vertex. If the vertex is taken to be the origin in the local parameterization, the neighboring vertices are assigned parameters based on "discrete exponential map" that depends on edge lengths and angles on the projected plane. Note that in this procedure each vertex may have several parameter values depending on the number of neighborhoods it belongs to, since the neighborhood of each vertex has a different local parameterization.

1.3.2 Curvature Discretization

From a theoretical point of view, meshes do not have curvature at all since all faces are flat and curvature is not properly defined along edges since the surface is not C^2 there. However, one can estimate the curvature thinking that the mesh is a piecewise-linear approximation of an unknown underlying smooth surface using only the information provided by the mesh.

There are two main directions in estimating curvature on a mesh: discrete and continuous. Discrete approximation is based on computing the differential geometry operators that work directly on the discrete representation of the mesh. Continuous estimation consists of fitting a surface locally then computing the operators on this continuous surface. Note that these methods may result in unexpected surface behavior between sample points resulting in errors. We will discuss discrete methods in depth while we will mention only a few continuous approximations.

Gaussian Curvature

The following discretization of Gaussian curvature has been used in many papers including [124, 165, 196].

$$K = \frac{3}{A}(2\pi - \sum_{i=1}^{M} \varphi_i) \tag{1.32}$$

where A is the area of the one ring neighborhood of the center vertex. The factor $3/A$ in the formula for K comes from the area of barycentric cells. Note that using barycentric cells for area is not always very accurate. The correct area that should be used is the area of Voronoi cells. However, in this case it is possible to encounter the case where the center of the Voronoi cell (the circumcenter) may lie outside of the triangle (in the case of obtuse triangles) [148].

Discretization of Gaussian curvature presented in equation (1.32) is also called the "angle deficit" (or "angle defect") and is one of the most commonly used formulas. One good property of this discretization is that it satisfies the Gauss-Bonnet theorem which is true in the smooth case. However, it has been shown that it is not a correct estimation of the Gaussian of the polyhedral surface or the underlying smooth surface [16], [1] - although it is justified in the case of a polyhedron whose edges are the geodesics of an associated surface [117]. One reason for this problem is the fact that such a discretization does not give information on the nature of the vertex, namely whether it is a convex, saddle or a “mixed” vertex. The proposed solution in [1] is to separate the negative and positive parts of curvature. In which case, positive, negative and absolute curvatures are given by the following set of equations:

$$\kappa^+ = 2\pi - \phi^+$$
$$\kappa^- = \kappa^+ - \kappa$$
$$\kappa_{abs} = \kappa^+ + \kappa^-$$

where κ is the angle deficit, and ϕ^+ is the total angle in the convex cone of the vertex. Convex cone is defined by the edges within the one-ring of a vertex that belong to the boundary of the convex hull. Note that κ_{abs} is analogue to $|\kappa_1| + |\kappa_2|$ in the continuous case [117]. This method distinguishes between three types of vertices (for the detection algorithm, see [178]). However, it is stated in [117] that there are at least six types of vertices: convex, concave, saddle, convex-fan, concave-fan, saddle-fan. The lack of cases for all these types leads to limitations on the quality of the approximation.

In [16], the angle deficit technique to curvature estimation is studied further. They show that the estimation of Gaussian curvature by angle deficit is correct only for very specific meshes and is wrong in general. It is established that when the distance between the center point and its neighbors reduces to zero, the angular defect is equal to a degree two polynomial in principal curvatures. It is shown that this polynomial can be computed precisely for regular meshes and that it makes sense to use angle deficit only for valances four (with neighbors aligned with principal directions) and six. It is also shown that in the case of an irregular mesh error incurred by using the angle deficit is dependent on the principal curvatures.

Another angle deficit based, yet slightly different formula for discrete Gaussian curvature is given in [117] and is called the "corrected" formula:

$$K = \frac{(2\pi - \sum_{i=1}^{M} \alpha_i)}{\frac{1}{2}A - \frac{1}{8}\sum_i \cot \alpha_i {l_i}^2}$$

The concept of normal cycles from differential geometry is introduced by Cohen-Steiner and Morvan [38]. Their results involve (discrete) curvature measures defined below. These measures are defined by integration of corresponding invariants on the normal cycles. The discrete Gaussian curvature measure is defined as the function that associates with every Borel set B the following quantity $\phi^K_{V(B)}$, where $g(p)$ is the angle deficit.

$$\phi^K_{V(B)} = \sum_{p \in (B \cap P)} g(p)$$

An angle deficit based, but a fairly different method is presented in [40]. Given a one ring around vertex x with edges of each triangle in the ring labeled a_i, b_i, c_i, latter one being the edge across from x, the formula for Gaussian curvature is given by:

$$K(x) = \frac{2 \triangle_k \, \delta(x - x_k)}{\sum_i A_i} \tag{1.33}$$

$$A_i = \sqrt{s(s-a_i)(s-b_i)(s-c_i)} \tag{1.34}$$

$$s = \frac{a_i + b_i + c_i}{2} \tag{1.35}$$

$$\triangle_k = 2\pi - \sum_i \phi_i \tag{1.36}$$

$$\phi_i = \arccos(\frac{a_i^2 + b_i^2 + c_i^2}{2a_i b_i}) \tag{1.37}$$

In [119] a spherical image based discrete method is introduced for estimating the Gaussian curvature. In the continuous case, the theorem based on spherical image is the following: Given a closed path around a point on a surface, if the tails of the unit normals are placed at the given point, the heads of the normals trace out a closed curve on the unit sphere. The limit of the area within this closed curve, divided by the area of the path as the path shrinks to the point is equal to the Gaussian curvature. In the discrete case, one can use the ratio of the area of triangles formed by the normals at the center vertex over the area of the one ring around the vertex as a discretization of the Gaussian curvature.

In the same paper, an asymptotic error analysis is given for several of the methods for Gaussian curvature and normal estimation on a surface that is a graph of a function $z = f(x, y)$. The results for the non-uniform case, namely when the samples (x, y) have no pattern, are summarized in the following table:

Method	Error in Normal	Error in Curvature
Quadratic Fit	$O(h^2)$	$O(h)$
Normal Average	$O(h)$	–
Spherical Image,		
given $O(h^2)$ accurate normals	–	$O(h)$
Angle Deficit	–	$O(1)$

Table 1.3: Error analysis results from [119].

Mean Curvature

A common discretization of mean curvature at a vertex x_i is given in equation (1.38) [165]. α and β are the angles opposite the shared edge e_{ij} between x_i and

x_j.

$$H_i = \frac{1}{2}\sum_j (\cot\alpha_j + \cot\beta_j)(x_j - x_i) \tag{1.38}$$

In [41] the same discretization with an area factor is used for mean curvature normal.

It is claimed in [109] that the cotangent weight approximation for mean curvature is not appropriate for getting point-wise correct values, however they do approximate normals correctly. They propose a complementary weighting scheme that guarantees point-wise, linearly convergent mean curvature. Using this new approximation, they derive formulas for estimating the Gaussian curvature as well as a curvature tensor.

The following discretization of the mean curvature was used in [196]. The $\triangle$ is the discretization of the Laplace-Beltrami operator given in equation (1.29) which is also based on the cotangent formula.

$$H = \frac{1}{2} N \cdot \triangle P$$

Most discretizations of mean curvature involve the edge length and the dihedral angle per edge. The following are examples of such discretizations:
Integral absolute mean curvature is given by the following [47].

$$H = \frac{1}{4}\sum_{i=1}^{n} ||\vec{e_i}|||\beta_i| \tag{1.39}$$

where n is the number of neighbors, e_i is the edge between central vertex and the ith neighbor, and β_i is the angle between the normals of two adjacent triangles. Note the similarity with the "mean curvature measure" given by equation (1.41).

Similarly, in [202] the integral mean curvature is used for smoothing. Their formula however, is slightly different:

$$H = \frac{1}{4} \sum_{i=1}^{n} ||\vec{e_i}||(\pi - \beta_i)^2 \tag{1.40}$$

Note that the mean curvature at edge e_i is given by $(\pi - \beta_i)^2/4$.

The discrete mean curvature measure is a similar function, where $|\beta(e)|$ is the angle between normals of the triangles incident to the edge e. Sign of β depends on the convexity of the edge [38].

$$\phi_V^H(B) = \sum_{e \in E} length(e \cap B)\beta(e) \tag{1.41}$$

The anisotropic discrete curvature measure is given by:

$$\sum_{e \in E} length(e \cap B)\beta(e)\vec{e} \otimes \vec{e}$$

Note that the anisotropic measure generalizes the mean curvature measure. In the case where the mesh is the restricted Delaunay triangulation of a sampled smooth surface these measures converge linearly towards those of the smooth surface.

In [121], a technique for extending notions such as curvatures from differential geometry to the discrete case is presented. The method is based on spatial averaging. The value computed by spatial averaging converges to the point-wise definition as sampling increases given that the initial mesh is a good approximation of the smooth surface. The discretization of the mean curvature normal in this paper is very similar to that given in equation (1.25). Only thing that differs is the denominator of the fraction, which in this case becomes $2A_{mixed}$. A_{mixed} is described as the following: For each triangle in the one-ring, if a triangle is not obtuse, then take the Voronoi area; if it is obtuse, depending on whether the angle at center is obtuse or not take the area halved or area quartered respectively. This formulation is claimed to be more accurate. They also use A_{mixed} in the discretization of Gaussian curvature based on angle deficit.

In [117], there is a formula for discrete mean curvature attached to an edge instead of a vertex which is based on the same idea as equation (1.39). Let us represent this formula as follows:

$$H(e) = \frac{1}{2} l(a)\alpha(a),$$

where α is the plane angle of the neighboring faces and $l(a)$ is the length of edge a.

In [207], this discretization is studied in the topic of discrete shell energy. The approximation can be improved by averaging these quantities over a sufficiently large area. For example, given a triangle, one can compute the mean curvature on the triangle as:

$$H(T) = \frac{1}{2A_t}(l_1\alpha_1 + l_2\alpha_2 + l_3\alpha_3). \tag{1.42}$$

The same holds true for mean curvature normal computed at a vertex, if the normals are added into the equation:

$$H(V)n = \frac{1}{2A_v} \sum_i l_i \alpha_i n_i,$$

where l_i are edges adjacent to vertex V and A_v is the area associated with the vertex. This is equivalent (up to second-order terms) to equation (1.25).

Note that equation (1.42) can be improved if more degrees of freedom are added in terms of angular variables associated with edges based on normals at edge midpoints. All extensions to this edge based mean curvature discretization by Zorin presents work in progress [207].

Discretization of the mean curvature given in [40] has the same structure as the one for the Gaussian given in equation (1.33). Once again the edges of the neighboring triangles are labeled a_i, b_i, c_i, where the latter corresponds to the edge across from the central vertex. A_i is same as in equation (1.34). The normals of associated faces are denoted n_i.

$$H = \frac{\frac{1}{2}\sum_i \arccos(n_i \cdot n_{i+1}) \cdot b_i}{\sum_i A_i}$$

In [92], a Willmore energy based variational method is used for surface design which requires a discretization of the mean curvature. The discrete mean curvature h_v at a vertex v used here is given by the following where a_v is the one third of the area of the one ring around the vertex:

$$2h_v = \frac{-\nabla_v a_v}{a_v}. \tag{1.43}$$

Mean curvature at an edge is given by the following in [87]:

$$H_e = 2|e|cos(\theta_e/2), \tag{1.44}$$

where θ_e is the dihedral angle at e.

Principal Curvatures

In [169] Taubin uses the assignment of a 3×3 quadratic form to each vertex where a function of its eigen values (vectors) correspond to principal curvatures (directions). The construction of the quadratic polynomial is linear in the number of neighbors of a given vertex. This quadratic is defined per vertex v_i where v_j belongs to the set of neighbors of v_i, T_{ij} is the unit length, normalized projection of $v_i - v_j$ onto tangent plane and κ_{ij} is the directional curvature in the direction of T_{ij} (Equation 1.46). Directional curvature is approximated using finite differences, therefore the surface mesh should be free of noise and may require a pre-smoothing step.

$$\kappa_{ij} = \frac{2n_i \cdot (v_i - v_j)}{|v_i - v_j|^2} \tag{1.45}$$

$$M_{v_i} = \sum_{v_j} w_{ij} \kappa_{ij} T_{ij} T_{ij}^t \tag{1.46}$$

Note that this may also be generalized to any polyhedral surface. In [166] two extensions to this method are proposed. One of them is based on changing the weights w_{ij} to be proportional to angles instead of surface areas. The other is changing κ_{ij} (equation (1.45)) to reflect the average of directional curvatures in two consecutive directions. This smooths large variations in directional curvatures. However, their results show that they are not significantly better than the original formulation by Taubin.

In [138] another vertex based discretization is introduced. The main difference here is that instead of the one ring around a given vertex, the triangles inside a geodesic neighborhood of the vertex are used. Then ”votes” from these faces are accumulated to compute normal or curvature data. The same method is also used to detect creases on noisy meshes. Although the results are compared to Taubin's, it is stated that they are not solving equivalent problems since Taubin's method requires a pre-smoothing step to remove surface noise.

Theisel, et al. [173] describe a different setup where the curvature tensor is computed on a (triangle) face instead of a vertex, given the normals at each of the three vertices. This is similar to Phong shading, where two different linear interpolations are used, one for the surface, and the other for normals. Using these interpolations one can compute the partials of the surface and the normals and therefore can construct the curvature tensor. It is shown that this tensor converges to that of the underlying smooth surface when the mesh is refined. Another property of this tensor is that it depends on the length of the normals associated with the triangle as well - therefore can be used as extra parameters to improve the quality of the estimation. If the curvatures or the tensor is required at a vertex as is the case with other methods, one can average the tensor values of the neighboring triangles. However, there are problems with this formulation in the case of irregular triangulations. Namely, the principal curvatures are in general not perpendicular and the tensor has discontinuities at the triangle junctions. However, the error is shown to be not much worse than that of other methods.

Another face-based discretization is proposed in [151]. Similar to the method above, to get the curvature or other differential operators at a vertex, an average of the operators on the faces is taken. In this case however, finite differences is used. The main difference between the two methods is that the tensor per face is constant in this case, whereas it is a continuous function on the face in [173]. This is caused by the fact that the required partial derivatives are computed with finite differences instead of linear interpolation. A very important feature of this formulation is that it can be extended to higher-order derivatives, such as a tensor for the derivative of curvature.

In [81], Hamann uses the fact that a surface can be approximated locally as a graph of a function, and uses such an approximation to compute principal curvatures. Given a vertex, the one-ring around it is used to compute the approximating graph function which is a quadratic in this case. This method ensures that the paraboloid passes through the central vertex. Given a quadratic in the following form:

$$f(u, v) = (1/2)(au^2 + 2buv + cv^2),$$

the principal curvatures are the roots of:

$$k^2 - (a + c)k + ac - b^2 = 0.$$

In [56] two second-order approximation methods, namely Taubin's [169] and one very similar to Hamann's [81] are compared against a new third-order approximation. The idea is to fit a cubic surface to adjacent vertex data similar to the quadratic in [81]. This time however, normals at adjacent vertices are also used such that each vertex contributes one equation for position and two equations for the normal. Now that there are 7 unknowns instead of 3 as in the quadratic case, increase in the number of equations is expected. Note however, that higher-order terms are not included in the equation for principal curvatures but they improve the fit for lower-order terms. It is shown that this cubic-order method performs better than the quadratic methods.

Razdan and Bae [148] recently introduced a method based on approximating the neighborhood of the vertex by a bi-quadratic Bezier surface. The neighborhood in this case is the double-ring around the central vertex to assure that the surface behaves well near the vertex. This does not guarantee the central vertex to lie on the approximated surface but it adds more flexibility to the surface fit and allows one to fit higher-order surfaces by just changing the basis functions. This method is shown to perform better than others in the case of an irregular triangulation.

One can compute the principal curvatures separately using any discretization of mean (H) and Gaussian (K) curvatures using the formula: $\kappa_{i,j} = H \pm \sqrt{H^2 - K}$. In [124], given K based on angle defect they do the following for estimations of principal curvatures: If $K > 0$ then $\kappa_1 = \kappa_2 = \sqrt{(K)}$. Else $\kappa_1 = \kappa_2 = 0$.

If one needs the absolute sum of principal curvatures the following formula may be used [47]:

$$|\kappa_1| + |\kappa_2| = \begin{cases} 2|H|, & \text{if } K \geq 0 \\ 2\sqrt{|H|^2 - K}, & \text{otherwise} \end{cases}$$

In [87], the shape operator at an edge is defined by:

$$S(e) = H_e(\vec{e} \times \vec{N}_e)(\vec{e} \times \vec{N}_e)^T \tag{1.47}$$

where H_e is given in equation (1.44) and N_e is the edge normal, computed as the average of the normals of the two adjacent triangles. The vertex-based shape operator at vertex p is given as the average over adjacent edges:

$$S(p) = \frac{1}{2}\sum_e (\vec{N}_p \cdot \vec{N}_e) S(e) \tag{1.48}$$

The two largest eigen values of this 3x3 matrix are the principal curvatures.

There is also a recent shape operator construction based on discrete focal surface in [198].

1.3.3 Finite Elements

Partial differential equations (PDEs) are commonly used to describe rate of change in time or space of some physical quantity. PDEs appear in computer graphics research often: physically based simulations, mesh optimization, and smooth surface design are just a few examples. Directly related problems were covered in Section 1.2.2. One technique for solving a PDE is the Finite Elements Method (FEM). This method presents a natural choice in graphics, especially in mesh based problems, due to the readily available discretization of the domain.

Unlike engineering and physics disciplines, in graphics, visual details, speed and robustness play more important roles than accuracy and convergence. However, convergence is important in graphics for it leads to mesh independent results. One of our results is based on finite elements where we provide evidence of convergence and therefore mesh independent visual results while solving a sixth-order PDE. In this section, we give a brief overview of the general finite elements methods with references to their use in some related work followed by a review of the mixed method, the type of finite element formulation we use in our results. In the last section we review literature where mixed finite elements has been used.

Finite Element Methods

The solution to a PDE modeling some geometric or physical occurrence does not always exist due to complexities in geometry or boundary conditions. In such cases, one usually settles for an approximate solution that is close enough to the exact solution. FEM provides one numerical technique to achieving such an approximate solution.

The idea is that the problem at hand can be reformulated in a variational form, where the problem is transformed into finding a function in a certain space of functions that minimizes a certain expression. To do so, one chooses a finite set of test functions (also known as shape functions), where the minimizing function is among their linear combinations, and this approximation has small, convergent error.

The spaces of functions used in finite element analysis are called Sobolev spaces. Given a domain Ω, a Sobolev space is a subspace of $L_2(\Omega)$ where $L_2(\Omega)$ is the set of functions which are square integrable over Ω. The notation $H^m(\Omega)$ is used to describe a Sobolev space where the weak derivatives up to order m are also square integrable. A weak derivative is a derivative taken in the sense of distributions ([23]). The subset of the space $H^m(\Omega)$ where all functions satisfy zero boundary conditions is referred to as $H_0^m(\Omega)$.

FEM solves the continuous problem by first breaking the domain into simple elements (e.g. triangles) and describe simple functions per element where the unknown weights in the linear combination are placed at nodes on an element (e.g. vertices of a triangle). The system is formulated as a set of equations that are gathered from equations written per element and then combined based on the connectivity of the domain. Once the nodal values are gathered, it is simple interpolation to compute the solution at any point within an element. The accuracy can be improved with increased resolution, without the need for more complex elements and the local nature of the test functions leads to sparse systems.

The choice of elements and function spaces that these test functions lie in depend on the complexity of the problem at hand. For the problems we are solving, we use linear Lagrangian elements. As the name suggests these elements support linear functions. The degrees of freedom associated with each triangle are weights defined on the vertices. The test functions are defined as hat functions centered at each vertex of the mesh, i.e. each test function equals one only at the vertex its index matches, everywhere else the function evaluates to zero.

One important thing to cover here is the distinction between the strong and the weak formulations of a given PDE, for we will be using it in our work. Suppose we are solving a boundary value problem of the form :

$Du = f$, with $u \in \Omega$ and $u|_{\partial\Omega} = u_b$, where u_b is the boundary condition and D is some differential operator. This is the differential or the *strong* form of the problem. The *weak* formulation weakens the derivative definition by domain averaging. This is done by introducing integral terms and is expressed as: $(Du, v) = (f, v)$ where $(\cdot, \cdot)$ is a bilinear form defined as $(u, v) = \int_\Omega uvdA$ and v is a test function.

The literature and resources in finite elements methods are immense, and it would be impossible to cover all the details and variations of the method. We refer the reader to [21, 33, 164] and [22] and the references therein.

Conforming and Non-Conforming Finite Elements. Finite elements refer to the finitely many sub-domains of the domain Ω. For two dimensional problems finite elements can be triangles or quadrilaterals. For three dimensions, which is not in the scope of this work, they can be tetrahedra or cubes.

Conforming finite elements are those that lie in the Sobolev space the problem is posed in. For example, for second-order problems, finite elements in H^1 are used. In [21], Theorem 5.2 proves that given $k \geq 1$ and Ω bounded, a piecewise infinitely differentiable function v belongs to H^k, if and only if $v \in C^{k-1}$. Therefore, for second-order problems, one can use C^0 conforming elements. An example of a C^0 element is the linear triangular element where function values are prescribed at triangle vertices. Note that, for a fourth-order problem, a conforming method would require the use of C^1 elements which are rather complicated.

In contrast, non-conforming elements are those that do not lie in the same Sobolev space as the one the problem is posed in. Non-conforming elements are used in higher-order problems where obtaining conforming finite elements is difficult. Since there are no inter-element continuity requirements, the elements are much simpler. However, the convergence results from conforming elements no longer hold here and a new set of convergence and error analysis results is needed. Most work in non-conforming finite elements involves constructing these elements and proving error bounds. An example of a non-conforming element is the quadratic Morley element with the least number of degrees of freedom used in approximations in H^2. The Morley element has prescribed function values at triangle vertices and normal derivative values at edge midpoints.

Mixed Finite Elements Methods

Mixed finite elements have been studied extensively (See [6,23] for reference and [4] for a concise overview.)

The idea behind mixed finite elements is to use independent approximations for both the dependent (primal) variable and variables defined as a function of its derivatives (auxiliary variables). In order to do so, one may use different finite-dimensional spaces for each variable and achieve different accuracies. One other point to note is that the variational formulation for a mixed method finds a saddle point as opposed to a minimum.

Mixed methods are preferred over others when an auxiliary variable is of more interest [4]. For other methods this variable would have to be computed applying differentiation after the problem has been solved, which could lead to less accuracy; whereas in mixed methods auxiliary variables are computed simultaneously with primal variables. One needs to be careful, since it is not guaranteed that through mixed methods auxiliary variables will be computed more accurately. It depends on spaces used.

Another reason is the fact that mixed methods allow the use of lower-order, therefore simpler elements while solving a fourth, sixth or a higher-order problem. In contrast, with non-mixed methods one would have to use the very complicated C^1 or C^2 elements respectively in order to solve fourth or sixth-order problems [4].

On the other hand, one should note that using mixed methods increases the number of variables in the system and leads to larger matrices. Furthermore, due to the saddle point property mentioned above the resulting system will be indefinite, whereas basic finite elements matrices are positive definite.

Typically, a mixed finite element discretization would result in a linear algebraic system of the form:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{1.49}$$

with A $n \times n$, and B $m \times n$, where n is the number of primal variables and m is the number of auxiliary variables.
As covered recently in [17], such sparse linear systems can be solved robustly and efficiently using sparse direct solvers.

For stability and solvability of a mixed element formulation it is important that it satisfies the *inf-sup* condition. The *inf-sup* condition is given by:

Definition 1.1. ([6], p.12) There exists a positive constant β, independent of the mesh size h, such that: $\forall y \in Y$, $\exists x \in X \setminus \{0\}$ such that $x^t B^T y >= \beta ||x||_X ||y||_Y$.

As described in [6], the *inf-sup* condition implies that the matrix B^T is injective; i.e. $n \geq m$. However, note that injectivity is not sufficient to show that the condition holds. There is an ellipticity condition on the matrix A that is necessary as well as conditions on the norms associated with the function spaces used.

The stability of these systems may be improved by the use and design of stable mixed elements. However the use of such elements may require extra constraints and lead to more complicated systems. See [5] for use of Lagrange multipliers to impose constraints in a mixed framework which leads to a positive definite system.

Mixed Methods Literature

We can only touch upon the literature on mixed methods. One of the commonly used mixed elements method is described in [34] and is referred to as the Ciarlet-Raviart formulation. This work solves the biharmonic problem with trivial boundary conditions. It involves C^0 finite elements with piecewise polynomials of degree at least two, and a proof for linear convergence. Scholz later proved error bounds for the same formulation for the biharmonic in L_2 and L_∞ for C^0 elements associated with linear or higher-order polynomials [159]. He showed $O(h)$ for the error in the primal variable and $O(\sqrt{h})$ for the auxiliary variable in L_2, and $O(h)$ in L_∞. Another extension is by [193] who utilizes quadratic and linear finite elements for the primal and auxiliary variables respectively. In the more recent work [2], the same problem is solved with non-trivial boundary conditions, with an optimality condition achieved through an extra term based on normal derivatives in the formulation. [24] talks about a different mixed formulation to solve the biharmonic in the context of the plate bending problem. Instead of using Δx as the auxiliary variable, all second derivatives of x are used as auxiliary variables.

In [122] the biharmonic equation with two different sets of boundary conditions are considered. The conditions $x|_{\partial\Omega} = 0$ and $\frac{\partial x}{\partial n}|_{\partial\Omega} = 0$ are referred to as the clamped plate problem and the other set of conditions lead to the simply supported plate problem. A formulation similar to that of Ciarlet-Raviart for the simply supported plate problem is introduced and error estimates are provided.

[55] is commonly referred to in the context of applying mixed methods to the biharmonic problem. The authors aim at decoupling the mixed system into two harmonic equations which would require the knowledge of Dirichlet boundary conditions for the auxiliary variable. A method to reduce the computation of these Dirichlet conditions to a linear system is studied and it is extended to the Stokes problem.

In [35], a fourth-order PDE based on the gradient of the Willmore flow is used for surface restoration. The PDE is solved by providing a weak formulation for the problem and solving it using the proper finite element spaces. They also reduce the system to two second-order systems which is along the lines of mixed methods. There are two variants for imposing boundary conditions: one involves prescribing Neumann boundary conditions on the boundary row and the other one uses an exterior row of triangles in the formulation to do the same. The theory behind the mixed formulation used was expanded later in [152].

Finite elements has been utilized to solve a surface diffusion problem in [8]. The normal velocity of the flow is proportional to the Laplacian of mean curvature. This leads to a fourth-order nonlinear PDE. In this case, there are four unknowns; curvature, curvature vector, Laplacian of curvature and vector form of the Laplacian of curvature. This breakdown of unknowns is representative of the mixed method where the formulation involves only zero and second-order operators. In this work boundary conditions are either non-existent (closed surfaces) or natural boundary conditions are employed.

A direct solver is proposed in [25] as opposed to hierarchical multi grid solvers or iterative solvers for fourth-order mixed finite element problems. This efficient and parallelizable solver is based on the idea of reducing the system to block diagonal form and utilizing block Gaussian elimination.

1.3.4 Numerical Methods for Energy Minimization

Here we summarize a few numerical methods used in energy minimization problems, mostly based on the summary provided in [183].

Linear Systems

Given a linear system $Ax = b$, where A is a sparse $n \times n$ matrix and b an n-vector, there are two basic procedures for solving for x: direct and iterative.

A *direct* method consists of factorizing the matrix, commonly as an LU factorization with permutations P and Q such that $PAQ = LU$ and the solution $x = Qy$ where $Uy = z$ and $Lz = Pb$. This is slow since the factorization algorithms take usually $O(n^3)$ time. It should only be used if n is small enough. However, depending on the properties of the matrix A, this running time can be improved.

An *iterative* method is based on computing a sequence of approximations x_k that converges to the solution x. The iteration is terminated when a stopping criteria based on accuracy is reached. To increase convergence rate one may apply a preconditioner to the matrix A. Two iterative methods for solving linear systems are conjugate gradient and conjugate residual algorithms. Conjugate gradient algorithm can only be used for positive definite A and conjugate residual algorithm may be used for symmetric indefinite A.

Unconstrained Minimization

Given a functional $f : R^n \rightarrow R$ that is continuous and possesses continuous second partial derivatives, the unconstrained minimization of f can be stated as:

$$\min_{x \in R^n} f(x)$$

If the functional f is nonlinear then usually an iterative method is used. The idea of generating a sequence of approximations is the same as is in the linear case. In this case though a descent principle must hold: If the point x_k is not a solution then $f(x_{k+1}) < f(x_k)$ and if x_k is the solution then $f(x_{k+1}) = f(x_k)$. The relation that describes an iterative method is given by:

$$x_{k+1} = x_k + t_k d_k,$$

where d_k is the descending search direction usually based on first and second derivatives of f, and t_k is determined by line search in the direction d_k.

In *steepest descent*, $d_k = -\nabla f(x_k)^T$, the negative of the gradient. One problem with this is that in two successive steps the search directions end up being perpendicular to each other. In general though, this methods converges linearly. In *Newton's method* the search direction is given by: $d_k = -H(x_k)^{-1}\nabla f(x_k)^T$, where H is the Hessian. Newton's method converges quadratically, but evaluation, storage and inversion of Hessian may be time/space consuming. The *non-linear conjugate gradient* method lies between these two. The search direction d_k is computed from gradients of all previous points. In [61] quasi-Newton methods are also mentioned.

Nonlinear Programming

Given two functions $f : R^n \to R$ and $h : R^n \to R^m$, for constraints, the general nonlinear programming problem is given by:

$$\min_{x \in R^n} f(x), \text{ with } h(x) = 0$$

If $h(x)$ is a linear constraint, namely in the form $Ax = b$, then one can use the constraints to eliminate variables. Elimination is based on solving for k of the n unknowns in terms of the $n-k$ unknowns and substituting in the functional. After this step one is left with an unconstrained problem on $n - k$ variables to solve as in the previous paragraph.

Quadratic Programming

The general quadratic programming problem with equality constraints can be expressed as:

$$\min_{x \in R^n} f(x) = x^T Ax + b^T x + c, \text{ with } D(x) = e$$

One way of solving this problem is by *Lagrange necessary conditions* which are linear in Lagrange multipliers λ and the unknowns x. The Lagrange necessary conditions are given by:

$$\begin{bmatrix} A + A^T & D^T \\ D & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} -b \\ e \end{bmatrix}$$

A problem with Lagrange multiplier method is that the system will be large for large degrees of freedom and constraints and furthermore, the matrix is not positive definite to allow for more efficient methods. Since the matrix has zero elements simple iterative methods would fail, but a block iteration method may be more appropriate [61].

A *penalty based* way of solving this problem is adding the constraints to the functional to be minimized as penalty terms, such that as this new energy is minimized, not satisfying the constraints get penalized. Once again one would be left with an unconstrained optimization problem although this new functional may be of any order [61].

A *calculus of variation* strategy is based on characterizing the solution to the optimization problem as the Euler equation which is a PDE [61]. In general for a functional of order m, one obtains PDEs of order $2m$. This PDE system can be solved by common methods such as finite elements or finite differences. However, since the order is rather high, special care must be given to the numerical issues.

2 Manifold Based Smooth Surface Construction

As discussed before, the problem we are solving is to define high-order quantities on a discrete setting and use such quantities in surface modeling. By a discrete setting, we refer to meshes where the mesh is approximating a smooth high-order surface. These approximations may not converge point-wise but the ease of directly working on the provided mesh, and the simplicity of some of the mesh operations make them desirable. We will consider two such approximations in Chapters 3 and 4. However, for these formulations to work well, one needs to use fairly high resolution meshes.

In the case of a coarse input mesh, one could try subdivision to build a higher resolution mesh and apply the same discretizations as above, but this may cause problems near high valance vertices for smoothness decreases near such vertices. As an alternative, one can use the fewer input vertices and sufficiently high-order basis functions to build smooth surfaces. Given such a smooth surface, one can compute differential quantities exactly, and in this setting these approximations are in fact point-wise convergent. This way one can avoid dealing with discretizations of curvature as described in Section 1.3.2 and compute this and other differential quantities directly on the surface.

In this chapter, we use a coarse quadrilateral input mesh and bases of any order of smoothness ranging from C^2 to C^∞, in order to represent a surface of any prescribed order of smoothness. We use a manifold-based construction that circumvents the problems of lower-order smoothness near extraordinary vertices that subdivision surfaces have. We discuss other desirable properties of manifold surfaces below.

Having computed such a surface, high-order functionals can now be computed exactly in the design of a variational surface (Section 1.2.1) or a high-order flow (Section 1.2.3). In fact the functionals described in Section 1.2 that approximate geometric invariants, do so by using surface derivatives, which are very straightforward to compute in this setting. Furthermore, many settings covered under Section 1.3.1 also apply directly. Even though we have not experimented with any of the surface optimization schemes described, it is straightforward to do so given the construction.

2.1 Introduction

Manifold-based representations of surfaces offer a unique combination of features: arbitrary topology, arbitrary-order smoothness, explicit analytic expressions for local parameterizations, linear dependence on control points, k-flexibility and visual quality.

In this chapter we present our detailed study of extensions of the manifold construction of [195], and explore the properties of resulting surfaces. We extend [195] in the following ways:

- *Surfaces with boundaries.* Practical applications require modeling surfaces with boundaries. We extend the manifold construction to surfaces with piecewise smooth boundary, describing constructions for three types of local charts: smooth boundary, convex and concave corners.
- *Derivative magnitude control.* [195] describes a C^∞ construction. However, the magnitude of derivatives of local parameterizations grows rapidly with

derivative number. We show that this is a general property of constructions with locally supported basis functions and derive a lower bound for derivative growth. If only a finite number of derivatives is needed, we describe a construction for C^d-manifolds for any d, with reduced derivative magnitude.

- *Flexibility.* By construction, manifold surfaces are at least 3-flexible at points corresponding to mesh vertices. We explore flexibility of resulting surfaces at arbitrary points, where it is not assured by construction.

This chapter is organized as follows: After a brief section on directly related previous work (Section 2.2), we present in Section 2.5 our modification of the method in order to support piecewise smooth boundaries on C^∞ surfaces. In Section 2.6.1 we present the method to generate C^d surfaces with any prescribed smoothness d followed by Section 2.6.2 providing the necessary modifications to add support for piecewise smooth boundaries of same smoothness. In Section 2.7, we explore optimal derivative behavior for these constructions and provide a lower bound for derivative magnitudes. Section 2.8 is devoted to the demonstration of the flexibility properties of our surfaces. We conclude with the presentation of our results.

2.2 Previous Work

A considerable number of C^d, $d > 1$ surface constructions for arbitrary meshes of various types were proposed over years, with the important case of $d = 2$ receiving most attention. Known constructions can be partitioned into three groups, based on spline patches, subdivision surfaces and manifolds respectively. We briefly review other high-order smoothness constructions and consider manifold-based constructions in more detail.

In Hagen and Pottmann [76] C^2 interpolants of boundary position, tangent and curvature data are constructed. Gregory and Hahn [58] describe a C^2 hole-filling algorithm; Bohl and Reif [15] describe C^2 conditions on degenerate patches and how N patches can be joined at a point. C^2 spline surfaces on arbitrary meshes were constructed by Peters [139] and C^d spline surfaces for general d are described by Prautzsch in [144]. More recently, various types of constructions based on polynomial patches were proposed in [140], [114] and [98]. C^2 subdivision algorithms based on standard schemes and with zero curvature at extraordinary vertices were proposed by Umlauf [145] and Biermann, et al. [10]. Flexible C^2 modifications of subdivision surfaces are described in [208] and [110].

The idea of manifold-based constructions was introduced in [64] where a manifold surface is built from a polygonal mesh using one chart per mesh element (vertices, edges and faces). In their construction, all interior vertices are required to have valance four (which can be achieved by a subdivision-based pre-process), and faces have to be polygons with no more than six edges. In [63] and [65] this manifold-based construction is used to generate parameterizations, where manifolds are used as parameter domains for surfaces of different topologies.

In [128] another manifold-based method for generating C^d surfaces is described. This method requires a pre-process of repeated Catmull-Clark subdivision steps in order to isolate irregularities of the mesh. The number of such steps is O(log$n$) in worst case, where n is an input parameter describing the extent of the influence of control points. Parts of regular and irregular subgraphs of the resulting mesh are chosen as charts with corresponding transition functions. The shapes and sizes of charts are dependent on the input parameter n and the regularity. In order to generalize the B-spline approach with this method, each chart is then mapped to a control point which describes the shape of the resulting surface. In [129], three realizations of this general scheme is described.

Extensions of these methods for surfaces with smooth boundaries have been outlined in most constructions mentioned above. In [64] the extension involves the definition of a different set of charts for boundary edges and vertices similar to our construction. On the other hand in [128], the extension for boundaries is achieved by adding auxiliary layers of faces around the mesh boundary such that all vertices can be treated the same. Compared to previously considered boundary constructions, our construction has the following distinctive features: boundary curves can be independent of the interior, and include corner points.

An important recent direction of work [67, 68] is the construction of manifold splines based on *affine atlases.* Compared to other manifold constructions, these constructions use an affine atlas, i.e. an atlas in which all transition maps are affine. The advantages in this case are considerable: transition maps are greatly simplified and it is no longer necessary to use a partition of unity to merge local geometries together; one can use a spline construction on arbitrary triangulations (e.g. DMS splines or Powell-Sabin splines) to build a purely polynomial surface. For meshes with boundary in principle, one can construct a purely affine atlas, and for closed meshes one can construct an affine atlas after a number of points determined by the genus of the mesh are removed.

One can observe that this type of constructions reduces the task of building a surface of arbitrary topology to local constructions on arbitrary triangulations. While methods for these constructions are available, they are relatively complex, especially for high order smoothness, and while these techniques yield good results for surface fitting, they are not commonly used for *ab initio* construction of surfaces of high-order smoothness from relatively coarse control meshes, which is our primary goal. One can regard the two constructions as complimentary.

The difference between two classes of constructions can be seen from considering the simplest example: a simply connected open mesh. In the case of affine atlas construction, the process proceeds as follows: the mesh is mapped piecewise linearly to the plane ([68] uses a type of discrete conformal parameterization) and an arbitrary triangulation spline construction is used to build the surface.

We note that solving a linear system dependent on the control mesh is required, and the charts are arbitrarily shaped k-gons formed by triangles adjacent to a vertex. As a result, one needs to be able to construct splines invariant with respect to affine re-parameterizations on such sets. This task is far from trivial and the visual shape quality of most resulting surfaces is far from the quality of standard B-splines or subdivision surfaces of similar order.

This is in contrast to commonly used geometric modeling constructions, such as subdivision surfaces or surface splines. We note that while all these constructions in a sense are affine manifold splines, but not on the whole mesh. Instead, they are affine manifold splines defined on the mesh with all vertices excluded, and with no chart dependence on the control mesh. In the case of the construction of [64] and related constructions, there is a finite number of simple charts. In the simplest case [195], there is a single chart type, and each 1-neighborhood of a vertex is mapped to this chart without a global parameterization procedure.

2.3 C^∞ Manifold Surfaces from Closed Meshes

In this section we briefly describe the manifold-based construction in [195], which our construction extends.

A set M has 2D manifold structure, if a collection of charts (C_i, φ_i) is defined, where C_i are open domains in the plane and φ_i are one-to-one maps $C_i \to M$, such that the images $\varphi_i(C_i)$ cover all of M.

M is a C^∞ (or C^d) manifold if the transition maps from chart to chart $t_{ji} = \varphi_j^{-1} \circ \varphi_i$, defined for pairs of charts for which $\varphi_i(C_i)$ and $\varphi_j(C_j)$ intersect, are C^∞ (or C^d). In [195], functions $f_i^{loc} : C_i \to R^3$ are constructed defining the geometry embedding functions locally on each chart. Then partition of unity functions w_i (i.e. functions with support restricted to individual charts C_i and summing up to 1) are used to combine local geometry embeddings. On M, the complete surface is defined by $\sum_i (w_i f_i^{loc}) \circ \varphi_i^{-1}$. However, in practice it is evaluated on individual charts C_i via

$$f_i(x) = \sum_{j:\varphi_i(x)\in\varphi_j(C_j)} w_j(t_{ji}(x)) f_j^{loc}(t_{ji}(x)) \tag{2.1}$$

Figure 2.1 illustrates the construction (transition maps to a single chart are not shown for clarity). Note that the complexity of evaluation of this expression is determined by the complexity of transition maps t_{ij}, weights w_j and geometry functions f_j^l. All three components are designed to be C^∞ to guarantee that the function $f_i(x)$, i.e. the resulting surface is C^∞.

2.3.1 Charts and Transition Maps

Chart maps φ_i are defined per vertex. Each chart domain is a curved star shape C_i, obtained by mapping the ring of faces sharing vertex v_i to the plane as described below. The overlap between the images of two charts in control mesh is two faces of the mesh. Instead of constructing the maps φ_i, the maps φ_i^{-1} are constructed. The chart construction proceeds in two steps: first, the faces adjacent to a given vertex are mapped piecewise bilinearly to the plane (maps L_i to domains S_i). Then

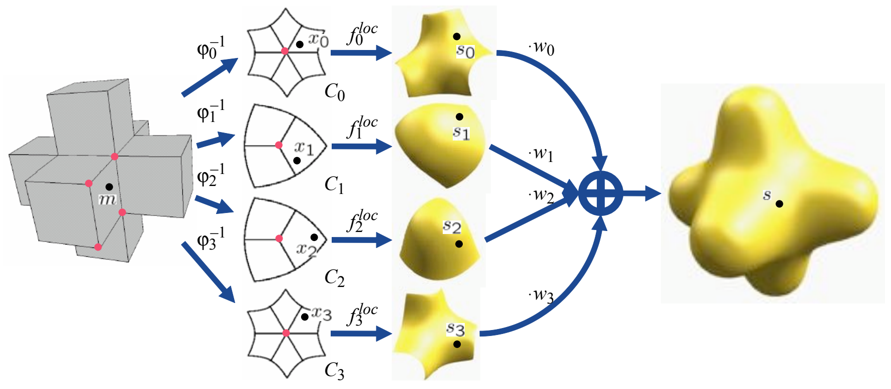

Figure 2.1: Components of the manifold-based surface construction.

a transformation c_i is applied to each wedge of the regular star S_i; c_i squeezes it so that it becomes a conformal image of a square. Therefore, the transition maps are compositions of two functions (Figure 2.3):

$$\varphi_i^{-1} = c_i \circ L_i.$$

The map c_i is a composition of two maps as well, a linear map l_{k_i} and a conformal map g_{k_i}:

$$c_i = g_{k_i} \circ l_{k_i}$$

where

$$g_{k_i} = z^{(4\alpha/k_i)}, \text{where } z = x + iy$$

$$l_{k_i} = \begin{pmatrix} \frac{\cos(\alpha\pi/4)}{\cos(\pi/k_i)} & 0 \\ 0 & \frac{\sin(\alpha\pi/4)}{\sin(\pi/k_i)} \end{pmatrix}.$$

In these formulas, $\alpha = 1$ for charts away from the surface boundary. Other values of α are used for constructing boundary charts as explained below.

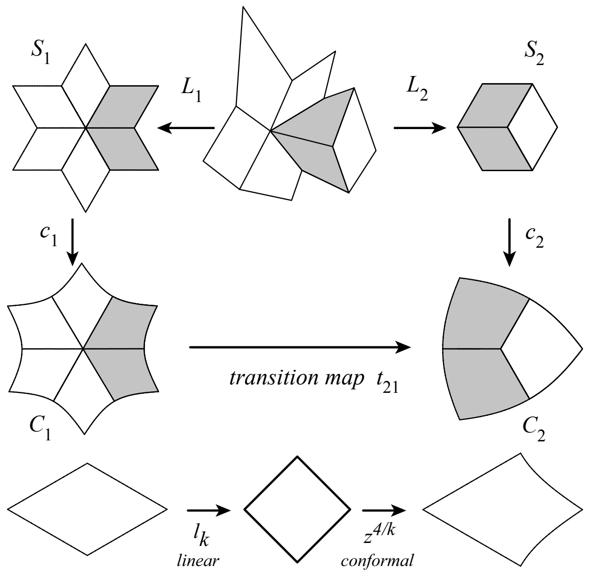

Figure 2.2: Chart map construction.

2.3.2 Partition of Unity

The partition of unity is built from identical pieces defined initially on the standard square $[0, 1]$ as a product of two identical one-dimensional C^∞ functions $\eta(u)\eta(v)$. The function η can be chosen in different ways, and the choice has a significant impact on surface behavior. We discuss possible choices in detail below. [195] uses η from [26]:

$$\eta(t) = \begin{cases} 1 & : 0 \le t \le \delta \\ \frac{h((t-\delta)/a)}{h((t-\delta)/a)+h(1-(t-\delta)/a)} & : \delta < t < 1-\delta \\ 0 & : 1-\delta \le t \le 1 \end{cases}$$

where $\delta > 0$ (1/8 is used), $a = 1 - 2\delta$ and $h(s) = \exp(2\exp(-1/s)/(s-1))$. Once the function is defined on the square, the weight for the whole chart is obtained as follows: First, a rotation by $\pi/4$ combined with the map g_k^{-1} is applied to remap

$\eta(u)\eta(v)$ to a single wedge. Then the function is defined by rotational symmetry on the rest of the chart.

2.3.3 Defining Geometry

In [195] geometry is defined using polynomials. Two steps of Catmull-Clark subdivision are applied to the mesh to define the coarse shape of the surface and then polynomials are used to fit the shape to these data points in the least squares sense. Every vertex of the refined mesh after the subdivision step can be assigned to points with bilinear coordinates $(l/4, n/4)$ in each sector S_i of the star. For each vertex v, these points in S_k are remapped to the chart domain C_i using the map c_i. There are $m = 12k+1$ points *inside* C_i which are denoted $x_0, ..., x_{m-1}$ (Figure 2.3). The $3D$ limit positions for these points are denoted as $s_0, ..., s_{m-1}$. The goal is to define a polynomial geometry function f of total degree $\leq d = \lfloor min(14, k+2) \rfloor$ such that differences $f(x_i) - s_i$ are minimized in the least squares sense (the matrix mapping sample points to polynomial coefficients is precomputed).

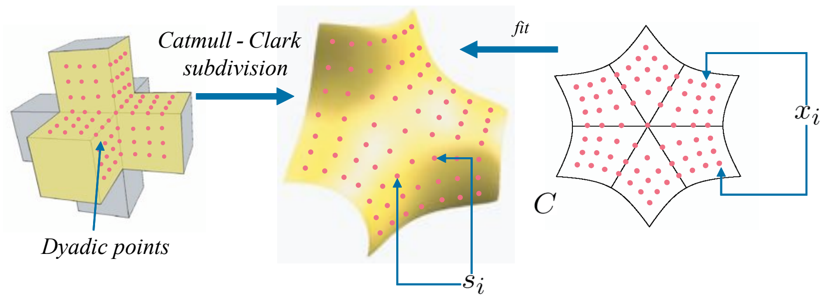

Figure 2.3: Defining geometry functions.

2.3.4 Summary

For a manifold construction of this type, we need to choose chart maps φ_i, geometry maps f_i^{loc} and functions η determining partition of unity functions w_i. In this work, we add more types of charts (C_i, φ_i) to handle surfaces with boundaries, and consider different choices for f_i and η.

2.4 Surfaces with Boundaries

One of our goals is to describe the manifold-based construction of surfaces with boundaries. A more general class of piecewise smooth surfaces can be easily obtained from this class by stitching smooth surfaces with boundaries along boundary curves. To be able to do this, the boundary curves have to be defined independently of the interior, i.e. the shape of the boundary curve should be completely determined by the control points on the boundary of the control mesh.

Recall that for a closed C^1-continuous surface in $\mathbf{R}^3$, each point has a neighborhood that can be smoothly deformed (that is, there is a C^1 map of maximal rank) into an open planar disk D. A surface with a *smooth boundary* can be described in a similar way, but neighborhoods of boundary points can be smoothly deformed into a half-disk H, with closed boundary (Figure 2.4). In order to allow *piecewise* smooth boundaries, we introduce two additional types of local charts: concave and convex corner charts, Q_3 and Q_1. A C^1-continuous surface with piecewise smooth boundary looks locally like one of the domains D, H, Q_1, or Q_3. Convex and concave corners, while being equivalent topologically, are not C^1-diffeomorphic, that is, there is no C^1 *non-degenerate* map from Q_1 to Q_3. One can show that almost all C^1-continuous surfaces obtained as linear combinations of basis functions are diffeomorphic. This implies that separate constructions are needed for convex

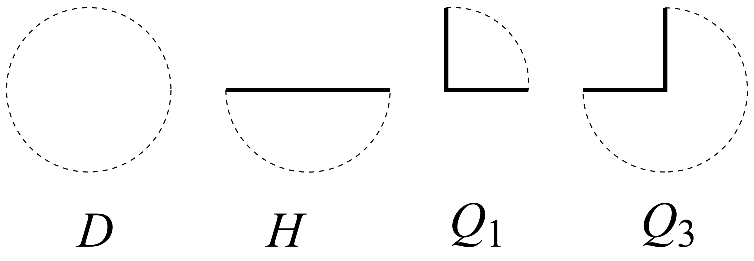

Figure 2.4: The charts for a surface with piecewise smooth boundary.

and concave boundary corners, just as it is the case for subdivision surfaces [10]. Rather than having one type of star-shaped chart domains per valence, as it was the case for closed surfaces, we need to consider three chart types as shown in the Figure 2.5.

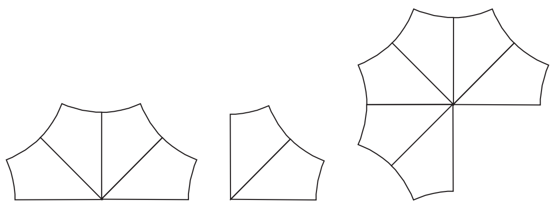

Figure 2.5: Left: valance 4 smooth boundary chart. Center: valance 2 convex corner chart. Right: valance 6 concave corner chart.

While the number of chart types one needs to consider increases, the main elements of the construction remain similar: we can use similarly constructed chart maps, partition of unity functions, and geometry functions. While geometry functions can be obtained using essentially the same fitting idea, we note that, just as it is the case for splines and subdivision surfaces, it is often desirable to have *interior independent* boundary curves which depend on boundary control points only. This property is useful whenever it is desirable to be able to modify the interior of a surface without changing the boundary or join two surfaces without gaps. In addition, this construction immediately yields a way to create interior creases, by treating tagged edges as boundary edges. On the other hand, interior-dependent constructions may yield better surface quality (Figure 2.8).

2.5 C^∞ Surfaces with Boundary

In this section, we describe the simplest extension of [195], preserving the construction in the interior in exactly the same form, and reusing most elements on the boundary. The input to our construction is a quadrilateral mesh, with tags for convex and concave corners.

The *charts and transition maps* for boundary vertices are constructed using a small change to the construction for interior vertices. We choose star-shaped areas shown in Figure 2.5. These are obtained by joining together k equal sectors for a vertex of valence k; however, instead of choosing the angular size of each sector to be $2\pi/k$, we use π/k for smooth boundary vertices, $3\pi/(2k)$ for concave corners and $\pi/(2k)$ for convex corners. The formulas for the maps are obtained by choosing α to be 1/2, 3/4 and 1/4 respectively. The partition of unity functions are constructed per quad, in the same way it is done in the interior vertex case.

The most significant difference is in the geometry definition. As discussed in Section 2.4, one can either constrain the boundary curves to be dependent only on the boundary control points, or allow boundary curves to be influenced by arbitrary points.

In both cases we start with two steps of Catmull-Clark subdivision, as in the interior case. The number of control points is now $m = 12k + 4$ instead of $m = 12k + 1$, as shown in Figure 2.6. We use the modified Catmull-Clark scheme described in [10] to handle all types of boundary vertices. The flatness parameter of this scheme can be used to control surface behavior near corner points.

The local geometry function is computed exactly as in the interior case using a polynomial fit with precomputed matrix; the only difference is in the choice of the maximal polynomial degree. In general we use monomials of total degree $\leq d = \lfloor min(14, k+2) \rfloor$ as the basis functions, excluding two empirically determined cases: for a smooth boundary vertex with valance $k = 2$ (valence for regular meshes with boundary) we choose $d = 6$. In addition, for concave corner vertices with valance $k \geq 6$ we choose degree $d = max(14, k)$. The choice of 14 as the maximal polynomial degree resulted from experiments with high valence vertices; higher degrees result in low-quality surfaces (see Figure 2.7).

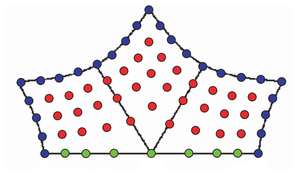
Figure 2.6: $12k+4$ sample points for a smooth boundary chart are shown in red (interior points) and green (boundary points). Blue points are not included.

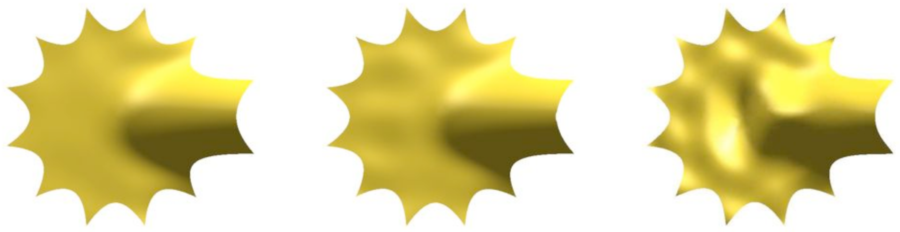
Figure 2.7: A valance 12 vertex chart with a lifted point on the right represented by polynomial fits of degree 14, 15, and 16 respectively.

2.5.1 Independent Boundary

In this case, we proceed in two steps: first, we compute the matrix for fitting polynomials only to the boundary points obtained by Catmull-Clark subdivision. Next, we compute a matrix to determine the coefficients of the remaining monomials in the basis, subject to the prescribed values on the boundary.

The smooth boundary vertices and corner vertices have to be treated slightly differently on the first step, because in the case of corners the boundary curve consists of two polynomial pieces. We assume that for a smooth boundary chart, the boundary is aligned with the u axis, and for the corner chart the two boundary pieces are aligned with u and v axes respectively, with the corner at zero. The sample points x_i have coordinates (u_i, v_i), $i = 1 \dots m$.

For a smooth boundary chart, we define the set of boundary monomials $p_i^{bnd}(u, v)$, $i = 1 \dots n^{bnd}$, to be the monomials u^{i-1} with $n^{bnd} = d + 1$. For a corner chart (either convex or concave), $n^{bnd} = 2d + 1$, $p_i^{bnd}(u, v) = u^{i-1}$, $i = 1 \dots d + 1$, and $p_i^{bnd}(u, v) = v^{i-d-1}$, $i = d + 2 \dots 2d + 1$. The remaining monomials up to total degree d, are denoted q_j, $j = 1 \dots n - n_{bnd}$. We also split the sample points into two groups: x_i^{bnd}, $i = 1 \dots 7$, and x_i^{int}, $i = 1 \dots m - 7$. s_i^{bnd} and s_i^{int} are corresponding 3d points obtained by subdivision.

The complete geometry function has the form

$$f(u, v) = \sum_{i=1}^{n_{bnd}} b_i^{bnd} p_i^{bnd}(u, v) + \sum_{i=1}^{n-n_{bnd}} b_i^{int} p_i^{int}(u, v)$$

Let $V_{c,d}$ be matrix $[p_i^c(x_j^d)]$ of the values of monomials evaluated at sample points, with c, d being one of bnd and int.

We first obtain the coefficients b_i^{bnd} by minimizing the difference $\|V_{bnd,bnd}b - s^{bnd}\|$ between the polynomial boundary curve and sample values s^{bnd}; as the number of polynomials typically exceeds the number of sample points, the solution is not unique. We use the standard approach to under-constrained optimization and minimize the sum of the squares of the coefficients to obtain a unique solution.

This solution is given by $V^{+}_{bnd,bnd}s^{bnd}$ where $V^{+}_{bnd,bnd}$ is the pseudo-inverse of $V_{bnd,bnd}$. Note that this procedure works in the same way for the smooth and corner boundary case. In the corner boundary case, it is equivalent to fitting polynomials in u to the segment of the boundary aligned with the u axis, and polynomials in v to the segment of the boundary aligned with the v axis.

Note that in all cases, the point samples on the boundary are interpolated, as the degree of the polynomials we use always exceeds the number of points. One could entirely eliminate the fitting step, and simply interpolate the points with sufficiently low degree polynomials, setting the coefficients for higher-order monomials to zero; however, using all polynomials up to degree d and minimizing the norm of coefficients appears to produce better results.

We determine the remaining coefficients b^{int} by minimizing $\sum_{i=8}^{m}(f(x_i) - s_i)^2$, while keeping the boundary coefficients b_i^{bnd} fixed. The matrix form of this expression is $\|V_{int,int}b^{int} + V_{int,bnd}b^{bnd} - s^{int}\|$, which leads to the following expression for b^{int}:

$$b^{int} = V^{+}_{int,int}(s^{int} - V_{int,bnd}b^{bnd})$$

Limitation of the interior-independent construction. For concave corners and interior-independent boundaries, the construction described above may produce somewhat undesirable shapes, independently of the parameters of the subdivision scheme used to compute the points x_i. Specifically, in cases when a concave corner is prescribed for a control mesh vertex for which the plane formed by the boundary edges meeting at the vertex is close to perpendicular to the nearby mesh

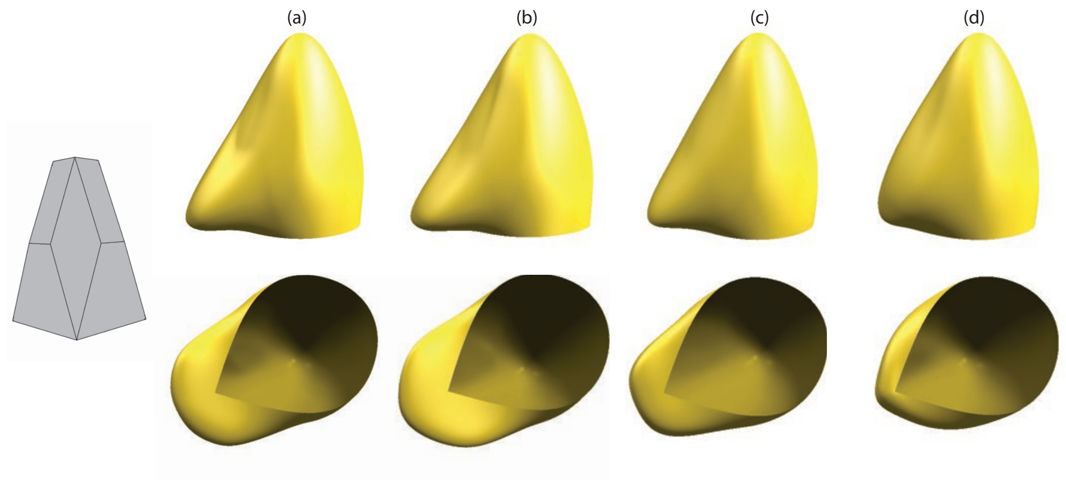

Figure 2.8: Limitation of concave corners with independent boundary. (a) C^{∞} surface, (b) C^5 surface, (c) C^3 surface, (d) Catmull-Clark subdivision surface.

faces (See Figure 2.8,(a)). In this case, even if the underlying subdivision surface makes a sharp turn to approach the boundary curve from the outside, as it is necessary for a concave corner, the polynomial approximation deviates significantly from the surface, and "overshoots" in a significant way.

The reason for this behavior can be understood as follows. Consider the curve on the local surface corresponding to line $u = 0$ in the chart. Note that all polynomials p_i^{int} vanish, so this curve on the surface is completely determined by the expression $\sum_i b_i^{bnd} p_i^{bnd}(0, v)$; as the coefficients b_i^{bnd} are computed using boundary data only, the points s_i^{int} have no effect on the curve behavior, even if it passes through corresponding points s_i^{int} in the interior of the surface. This is in contrast to the interior-dependent construction, which results in better behavior.

To large extent, this problem can be addressed if instead of C^{∞} surfaces, we use C^d for a large d based on splines, as explained in the next section. (See Figure 2.8: (b), (c))

2.6 C^d-continuous Surfaces

While for C^∞ surfaces derivatives of parameterizations are formally defined for all orders, the magnitudes of these derivatives can grow rapidly. There is a lower bound for this growth: for linear parametric surface constructions with localized influence of control points makes it impossible to have an upper bound for parameterization derivatives uniform in the derivative order (see Section 2.7). However, C^∞ construction yields derivative growth faster than optimal. High-order polynomials used in the C^∞ construction may lead to surface quality degradation for high valence (See Figure 2.9). It turns out that derivative magnitudes can be reduced and better surface quality can be achieved if we consider C^d-continuous surfaces for some finite d, constructed using splines instead of polynomials for geometry and blending functions.

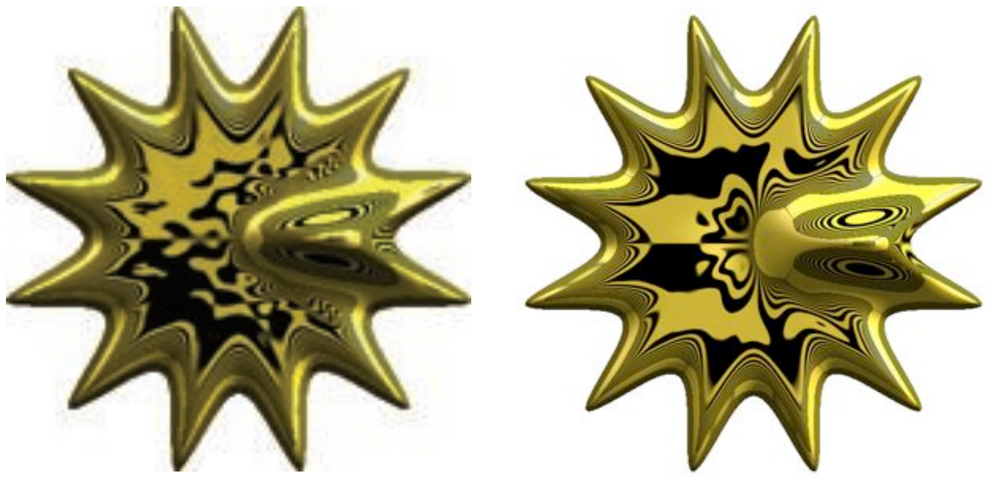

Figure 2.9: Left: A mesh with a valence 12 vertex fit using polynomials. Right: Same mesh fit using bi-cubic splines.

2.6.1 C^d Construction for Interior Charts

The differences between the C^∞ construction and C^d, $d < \infty$, construction for interior charts are small. For C^d-continuity, we replace the partition of unity function with the degree $(d+1)$ B-Spline basis function and we utilize a uniform bi-degree $(d+1)$ tensor product B-Spline fit for the geometry. Charts and transition maps remain the same as for C^∞ construction.

Similar to the C^∞ construction, we start with two steps of modified Catmull-Clark subdivision to get the data points to fit the surface to. As before, we denote resulting m points in the domain D_k as $x_0, ..., x_{m-1}$, and corresponding surface points are $s_0, ..., s_{m-1}$.

We define a grid of knots with associated control points c_{jk} such that all s_i are within the limits of this grid. One can choose to have any number of grid patches. Let g be the number of grid points. If $g = 2$, we have a single polynomial patch, and our construction of geometry functions is identical to the C^∞ case. We have observed that increasing the number of patches to more than 4 (i.e. $g > 3$) does not lead to any improvement. Therefore, we only utilize two choices of g: We use $g = 2$ only when the prescribed degree leads to an undetermined system while fitting (See last paragraph of this section). Otherwise, we use $g = 3$.

For $(g-1) \times (g-1)$ spline patches of degree $d+1$ require a grid of control points of size $(g+2e) \times (g+2e)$, where $e = \lfloor (d/2) \rfloor$.

$$f(u,v) = \sum_{j=-e}^{g+e} \sum_{k=-e}^{g+e} c_{jk} N_j(u) N_k(v) \tag{2.2}$$

In the C^∞ case (and equivalent $g = 2$ case) it is sufficient to use only subdivision points x_i in the interior of the chart. For larger values of g ($g = 3$ in particular), we found it important to include boundary points. Due to local nature of spline fit, and the star-like shape of the configuration of data points, spline control points near the boundary have small influence on the shape of the surface inside the star-shaped domain, and, as a consequence, the least-squares fit is ill-conditioned (a small change in the data point values can result in a large change in the control point position) and may lead to poor surface quality (See Figure 2.10). The conditioning becomes worse with increasing g as the support of each basis function decreases. Using an extra row of data points decreases ill-conditioning of the fit. In this case, the number of data points is $m = 20K + 1$ (or $m = 20K + 5$ for smooth boundary and corner vertices), where K is the valence of the vertex.

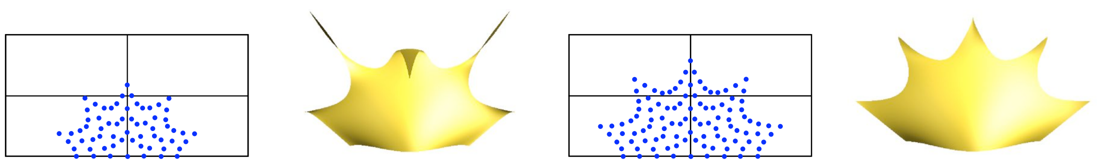

Figure 2.10: Left: A valance 5 boundary vertex chart represented by a bi-cubic spline surface using 12K+4 data points. Right: Same chart represented by a bi-cubic spline surface using an extra ring of data points.

The fit for spline patches proceeds exactly as for C^∞ case, with one important difference: because the degree of the splines $d + 1$ is not chosen automatically, but set by the user, if the number of data points m is too small and is less than the total number of degrees of freedom of the spline $(g + 2e)^2$, the fit may be under-determined.

In this case, we force $g = 2$ (i.e. a single polynomial patch) and decrease the degree d for this patch so that $m > (g + 2e)^2$. As a single polynomial patch is C^∞, the smoothness of the surface does not decrease for lower-degree splines in this case.

2.6.2 C^d Surfaces with Boundary

As with the C^∞ construction, we consider two boundary constructions: boundary curves depending on interior control points, and boundary curves depending on boundary control points only.

We do not discuss the first case in detail, as in this case the construction is identical to the interior vertex construction. The only difference is in the placement of the spline knot grid over the data points. In the smooth boundary case, the boundary of the chart is matched to the lower side of the grid patch. For a convex corner, we match the grid's lower left corner to the convex corner of the chart. The concave corner is treated like an interior vertex chart (See Figure 2.11).

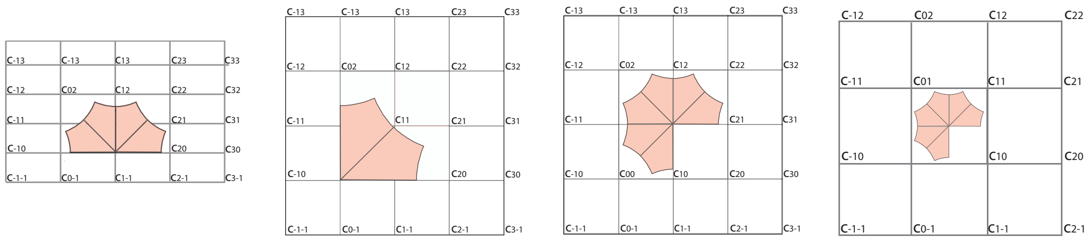

Figure 2.11: Knot numbering for smooth boundary, convex and concave charts for $g = 3$. The last image depicts the case of concave chart for $g = 2$.

In the remainder of this section, we focus on the interior-independent construction. Our construction consists of two steps similar to the C^∞ case:

1. compute a boundary curve;
2. fit the interior of the surface to the data points, keeping the boundary curve fixed.

Step 1: The Boundary Computation

We compute the boundary for the chart by fitting a spline curve to the data points s_i^{bnd} corresponding to boundary points x_i^{bnd} (In the case of corner points, there are two separate spline curves interpolating the corner). The number of control points for the boundary curve(s) for high degree d may exceed the number r of available boundary data points. In this case, we fit a cubic polynomial curve to the data points (this is always possible) and elevate the degree to $d+1$. In this section, $\bar{c}_i$ is used to denote control points of spline boundary curves and c_{jk} is used for control points of spline surfaces.

Smooth Boundary Vertex. For a smooth boundary vertex, the number of data points $r = 9$, as it is clear from Figure 2.6. When $d + 1 \leq r$, and we have enough data points on the boundary to define a curve of prescribed degree d, we use a standard B-Spline curve fit. The vector of control points $\bar{c}$ of length $g + 2e$ is obtained as $\bar{c} = M^{+}s^{bnd}$, where s^{bnd} is the vector of $r = 9$ boundary points, and $M_{ij} = N_j(x_i^{bnd})$, where N^d are the spline basis functions on the boundary. If $d + 1 > r$, that is, the degree is too high for a fit, we use a cubic fit and degree elevation.

Convex Corner. For a corner vertex, $r = 5$ for each of the two boundary curves. When $d + 1 \leq r$, we use a least-squares fit for each boundary curve. However, to ensure interpolation, we use reflection-like constraints on the boundary curve for the corner point. Specifically, we use only control points c_j^{bnd}, $j = 1 \ldots g + e$ as degrees of freedom. $\bar{c}_0$ is set to s_0^{bnd}, the data value at the control point. The remaining e control points are set to be $\bar{c}_j = 2\bar{c}_0 - \bar{c}_{-j}$ for $j = -e \ldots -1$, that is, reflection of $\bar{c}_{-j}$ about $\bar{c}_0$.

The expression for the boundary curve in this case is

$$f(x) = \bar{c}_0 \left(N_0(x) + 2 \sum_{j=1}^{e} N_{-j}(x) \right) - \sum_{j=1}^{e} \bar{c}_j \left(N_{-j}(x) + N_j(x) \right) + \sum_{j=-e+1}^{g+e} \bar{c}_j N_j(x), \tag{2.3}$$

where $\cdot^{bnd}$ subscript is omitted.

Similar to all previous cases, a $5 \times (g + e)$ matrix M^c is constructed, and the control points $\bar{c}_i$ are obtained by applying it to an appropriately modified vector of data points. $M^c_{ij} = (N_{-j}(x_i) + N_j(x_i))$, if $i \leq e$ and $M^c_{ij} = N_j(x_i)$ if $(g+e) \geq i > e$. The right hand side is given by $s_i - c_0 \left(N_0(x_i) + 2 \sum_{j=1}^{e} N_{-j}(x_i) \right)$.

If we do not have enough data points s_i for a degree d, we utilize degree elevation as necessary; we ensure that $g = 2$ in these cases.

Concave Corner. The concave corner case is identical to the convex corner case when $g = 3$: we fit two boundary curves independently, using reflections for interpolation at the corner point. When $g = 2$, the control points do not lie on the boundary as they do for g=3 (See Figure 2.11). Therefore, there does not exist a center vertex to do reflection over. Moreover, note that there are 5 data points and 4 control points per boundary curve. A direct least squares would lead to a curve that is not interpolating the corner. To guarantee that the equation will be satisfied exactly such that the curve will be interpolating the corner, we solve for one of the control points, say c_{-10} in the equation for the corner data point s_0.

$$\bar{c}_{-1} = \frac{1}{N_{-1}(x_0)} (s_0 - \bar{c}_0 N_0(x_0) - \bar{c}_1 N_1(x_0) - barc_2 N_2(x_0)) \tag{2.4}$$

Then the expression defining the corner interpolating boundary becomes

$$\bar{c}_0(N_0(x_i) - \frac{N_{-1}(x_i)}{N_{-1}(x_0)}N_0(x_0)) + \bar{c}_1(N_1(x_i) - \frac{N_{-1}(x_i)}{N_{-1}(x_0)}N_1(x_0)) + \bar{c}_2(N_2(x_i) - \frac{N_{-1}(x_i)}{N_{-1}(x_0)}N_2(x_0)) = s_i - s_0\frac{N_{-1}(x_i)}{N_{-1}(x_0)}$$

Step 2: Interior Fit

Once the boundary points are fixed, the rest of the control points are determined using a least-squares fit with constraints. Given q constraints we solve for q control points exactly. We call the vector of these points σ_2 and the vector of the rest of the control points is denoted σ_1.

We write the constraint as

$$[B_1|B_2]\sigma = c, \tag{2.5}$$

where B_1 is (q) x $(n-q)$ and B_2 is (q) x (q), and σ is the vector σ_1 appended by vector σ_2. Using this notation, we can write σ_2 in terms of σ_1:

$$\sigma_2 = B_2^{-1}(c - B_1\sigma_1) \tag{2.6}$$

Similarly, we can write the fitting process in block matrix form. Let matrices A_1 and A_2 be the matrices of basis functions $N_{jk}(x_i)$ for control points σ_1 and σ_2 respectively, evaluated at all m data points. Then the surface fit, in block matrix form, is the following:

$$[A_1|A_2]\sigma = s \tag{2.7}$$

By substituting equation (2.6) for σ_2 in equation (2.7), we get the following system of equations:

$$\sigma_1 = U^{+}t \tag{2.8}$$

where $U = (A_1 - A_2B_2^{-1}B_1)$ and $t = s - A_2B_2^{-1}c$. Once we know σ_1, we solve for σ_2 using equation (2.6). This concludes the construction.

Smooth Boundary Vertex. In the case of a smooth boundary chart, we generate grid points such that the control points on the zeroth row of the knot grid (i.e., c_{j0}) correspond to the boundary, where the indices start at $-e$ (See Figure 2.11). Then to achieve the match between interior and the boundary the following has to hold:

$$\sum_{j=-e}^{g+e} \sum_{k=-e}^{g+2} c_{jk} N_j(x_{u_i}) N_k(0) = \sum_{j=-e}^{g+e} c_j N_j(x_{u_i})$$

When simplified we gather that this is equivalent to the following for all $j \in \{-e, ..., g+e\}$:

$$\sum_{k=-e}^{g+e} c_{jk} N_k(0) - c_j = 0. \tag{2.9}$$

This is a total of $g+2e$ equations, that relate the control points of the boundary curve and those of the whole surface. Thus, $q = g + 2e$. To guarantee that the boundary is independent we choose a set of $g + 2e$ control points to satisfy these equations exactly. In the smooth boundary case the control points that lie on the row corresponding to the boundary (c_{j0}) forms this set. We call the vector of these points σ_2 and the vector of the rest of the control points is denoted σ_1. Then we follow the algorithm explained above.

Convex Corner. Similar to the boundary case, we generate the control grid such that the u-boundary corresponds to the control points c_{j0}, where the indices start at $-e$. (For the v-boundary the control points for the boundary become c_{0k}.) We also take control point c_{00} to correspond to the corner data point (See Figure 2.11). Then to achieve the match between interior and the boundary in u-direction equation (2.9) has to hold for all $j \in \{-e, ..., g+e\}$. This gives us $g+2e$

equations that guarantee match for only the u-boundary. For the v-boundary to match, the following has to hold:

$$\sum_{j=-e}^{g+e} \sum_{k=-e}^{g+2} c_{jk} N_j(0) N_k(x_{u_i}) = \sum_{j=-e}^{g+e} c_k N_k(x_{u_i}),$$

which when simplified is:

$$\sum_{j=-e}^{g+e} c_{jk} N_j(0) - c_k = 0, \quad \forall k \in \{-e, ..., g+e\}. \tag{2.10}$$

This gives us another set of $(g + 2e)$ equations. Then in total we have $(2g + 4e)$ equations. However, since the two boundary curves are not totally independent due to the fact that they both pass through the corner point, there exists an equation that is redundant. We remove any one of the two equations involving c_0. This leaves a set of $(2g + 4e - 1)$ independent equations. Thus, $q = (2g + 4e - 1)$. Similarly we pick a subset of q control points and denote them as σ_2. In this case we choose c_{j0}, $\forall j \in -e, ..., g + e$ and c_{0k}, $\forall k \in -e, ..., g + e$. Then we follow the general algorithm given above.

Concave Corner. Similar to the convex case, once the boundary curves are computed and are defined using the $g + 2e$ control points, we setup equations to constrain the interior to match the boundary, where the number of equations $q = 2g + 4e - 1$. The equations follow the structure of equations (2.9) and (2.10). The only difference is replacement of $N_j(0)$ by $N_j(.5)$ for $g = 2$ and by $N_j(1)$ for $g = 3$. Furthermore, what we include in σ_2 differ slightly based on g. For $g = 3$, since the boundary curves coincide with c_{j1} and c_{1k}, these naturally form σ_2. If $g = 2$ however, σ_2 is formed by $c_{j,-e}$ and $c_{g+e,k}$ for there is no natural row or column of control points that correspond to the boundaries and those seem to be a reasonable set to constrain for the surface-boundary match. Then we follow the general algorithm.

Limitation of Concave Corners. Similar to the C^∞ case, the concave corners do not result in surfaces with sufficiently good visual quality (See Figure 2.8). Especially if $g = 2$, i.e., one polynomial patch, it is the exact same problem that arises with the polynomial fit. This can be avoided by increasing the number of grid points, therefore increasing the number of patches, such that the control points for the boundary would affect a limited part of the surface. However, this would result in under-determined systems for low valances or high degrees. In this case, we would need more data points, possibly from another step of subdivision.

2.7 Optimality of Derivative Behavior

One of the reasons to consider the C^d construction described in Sections 2.6.1 and 2.6.2 is to obtain better behavior for lower-order derivatives than we get for the C^∞ construction. Numerical estimates show that the magnitude of the derivatives is primarily determined by the magnitude of the derivatives of partition of unity functions.

As our functions are tensor-product functions, the magnitude of these derivatives is directly related to the magnitude of the derivatives of single-variable partition of unity functions used in the tensor-product construction. It turns out that for a fixed-size support, fixed maximal value and given smoothness, it is possible to derive an explicit *lower* bounds on the sup and L_2 norms of d-th derivatives (W^d_∞ and W^d_2 norms respectively). Furthermore, one can explicitly construct partition of unity functions minimizing these norms.

In this section, we derive these lower bounds and show corresponding partition of unity functions and show that the uniform spline construction is within a constant from the minimum. While a slight quantitative improvement can be obtained by switching to the more complex non-uniform functions defined here, it is unlikely that the extra complexity is justified.

2.7.1 Minimizing W^d_∞ Norm

We consider this case most important, as in most cases point-wise derivative magnitude is of primary importance, rather than their average over the domain. The one-dimensional functions $f(t)$ (e.g., $\eta(t)$) defined on $[0, 1]$ that we use to construct the partition of unity functions should have the following properties:

- $f(0) = 0$, $f(1) = 1$;
- $f^{(i)}(0) = f^{(i)}(1) = 0$, for $i = 0 \ldots n - 1$
- $\sup f^{(n)} \to \min$

Note that we impose C^i of the function up to order $i = n - 1$ only. As we will see, the optimum is attained by a function with piecewise constant n-th derivative, with discontinuities at the endpoints: the optimal function is not contained in the set of C^n-continuous functions.

It turns out that the solution of this problem is a specific type of non-uniform spline of degree $n - 1$. We give an explicit formula for the optimal value of n-th derivative, and an explicit construction of the spline. We proceed in several steps: first, we reduce the problem to an optimal control problem. Next, we convert the optimal control problem to a moments problem, which is a special case of a moments problem for which a set of recursive formulas defining the solution is known ([107]). Finally, we derive a remarkably simple formula for the optimal derivative value using properties of Catalan numbers.

Equivalent optimal control problem. We reformulate the problem as an *optimal control* problem for a system of ODEs, introducing new functions. Let $f_0 = f$ and $f_i = f^{(i)}$, $\forall i \in \{1, 2, ..., n-1\}$. Then we can rephrase the problem as follows. *Find functions $f_i(t)$, $i \in \{1, 2, ..., n-1\}$ and $W(t)$ such that the following three conditions are satisfied.*

$$\dot{f}_i = f_{i+1}, \quad \forall i \in \{0, ..., n-2\}. \tag{2.11}$$

$$\dot{f}_{n-1} = W \tag{2.12}$$

$$\max |W| \to \min \tag{2.13}$$

We call this problem (P1). We now change the variables in this problem to convert it to an *optimal control* problem with an upper bound on n-th derivative, for which an explicit solution can be obtained using Pontryagin's maximum principle. Instead of considering fixed support $[0, 1]$, and minimizing the maximum of n-th derivative, we consider an equivalent problem for which the support size is minimized while the n-th derivative stays bounded. This problem, in the case of $n = 2$, has a natural physical interpretation: minimize the time it takes to traverse a distance, starting and stopping with zero speed, while keeping the acceleration bounded by a constant.

To convert the problem to this equivalent form, we introduce a new variable $s = s_{\max} \cdot t$. We define a new set of functions:

$$g_i(s) = s_{\max}^{-i} f_i(\frac{s}{s_{\max}}).$$

By the chain rule, the functions $g_i(s)$ satisfy $\dot{g}_i = g_{i+1}$, where $\dot{}$ now denotes differentiation with respect to s.

Also note that equation (2.12) becomes:

$$\dot{g}_{n-1}(s) = s_{\max}^{-n} W(\frac{s}{s_{\max}}).$$

We now define a new problem which we will refer to as (P2):

$$\begin{aligned}
&g_0(0) = 0, g_0(s_{\max}) = 1\\
&g_i(0) = g_i(s_{\max}) = 0,\ i = 1, ..., n-1\\
&\dot{g}_i = g_{i+1},\ i = 1, ..., n-1\\
&\dot{g}_{n-1} = u\\
&s_{\max} \to \min,\ \text{subject to } |u| \leq 1
\end{aligned}$$

Lemma 2.1. *$W(t)$ is a solution of (P1) and $u(s)$ is a solution of (P2) if and only if these functions are related as $u(s) = s_{\max}^n W(\frac{s}{s_{\max}})$ where $s_{\max} = (\max|W|)^{(1/n)}$.*

Proof. Suppose $W(t)$ is a solution of (P1). Suppose there is an $\tilde{s}_{\max} < (\max|W|)^{(1/n)}$ such that there are functions $\tilde{g}_i$ and $\tilde{u}$ that satisfy conditions (P2) with $s_{\max} = \tilde{s}_{\max}$. Then define

$$\tilde{W}(t) = \tilde{s}_{\max}^n \tilde{g}_n(\tilde{s}_{\max} \cdot t) = \tilde{s}_{\max}^n \tilde{u}(\tilde{s}_{\max} \cdot t).$$

Therefore, $\max|\tilde{W}(t)| \leq \tilde{s}_{\max}^n < \max|W(t)|$. Therefore, W is not a solution of (P1) and thus, we reach a contradiction.

Conversely, we can show in the same way that if u is a solution of (P2), then for a solution W of (P1) $\max|W| = s_{\max}^n$. We conclude that the problems are equivalent. □

We apply the Pontryagin's Maximum Principle in order to solve (P2). In general form, for linear systems it states the following:

Theorem 2.1 (Pontryagin's Maximum Principle(PMP))**.** *Consider an ODE system*

$$\begin{aligned} \dot{x}(t) &= Mx(t) + N\alpha(t) \\ x(0) &= x^0 \end{aligned}$$

where $x(t)$ is the solution trajectory, $\alpha(t)$ is the control, M is an $n \times n$ matrix and N is a length n vector. Consider a payoff function $P[\alpha(\cdot)]$ of the form

$$P[\alpha(\cdot)] = \int_0^T r(x,\alpha)dt + g(x(T))$$

where T is the terminal time, $r(x,\alpha)$ is the running payoff and g is the terminal payoff. Let ξ be the adjoint function, satisfying the following system of equations:

$$\begin{aligned} \dot{x}(t) &= \frac{\partial H}{\partial \xi} \\ \dot{\xi}(t) &= -\frac{\partial H}{\partial x} \\ \xi(T) &= \frac{\partial g}{\partial x}(x(T)) \end{aligned}$$

Then the maximizer of the Hamiltonian $H = x\xi + r$ maximizes the functional $P[\alpha(\cdot)]$, where ξ in the Hamiltonian is the adjoint.

To apply PMP to problem (P2) we let u be our control, and $g = [g_0, g_1, ..., g_{n-1}]$ be the state variable x. $g(0) = g_{init} = [0, 0, ..., 0]$ and $g(s_{\max}) = g_{final} = [1, 0, 0, ..., 0]$.

The payoff functional is $P[u(\cdot)] = \int_0^{s_{\max}} ds = s_{\max}$ The Hamiltonian has the form $H = \dot{g}\xi + 1 = \sum_{i=0}^{n-2}(g_{i+1}\xi_i) + u\xi_{n-1} + 1$ with ξ_i satisfying:

$$\begin{aligned} &\dot{\xi}_0 = 0, \\ &\dot{\xi}_i = -\xi_{i-1},\ i = 1, ..., n-1. \end{aligned}$$

We make two important observations: First, as $\dot{\xi}_{n-1} = 0$, the adjoint $\xi_{n-1}(s)$ is a polynomial of degree $n-1$, and has no more than $n-1$ roots. Secondly, H is maximized with respect to u subject to $|u| \leq 1$, if $|u| = 1$, with $\text{sign}(u) = \text{sign}(\xi_{n-1})$.

Thus, our problem is reduced to finding a piecewise constant $g_n = u$, $|u| = 1$ and $s_{\max}$ such that the system of ODEs of problem (P2) has a solution.

The equivalent moments problem. The optimization problem above can be reduced to the moments problem for the piecewise-constant function u. The functions g, and the actual spline function f can then be obtained by repeated integration and rescaling.

Observe that for $i \leq n-2$,

$$0 = g_i(s_{\max}) = \int_0^{s_{\max}} g_{i+1} ds = g_{i+1} s|_0^{s_{\max}} - \int_0^{s_{\max}} \dot{g}_{i+1} s ds = -\int_0^{s_{\max}} g_{i+2} s ds$$

By induction, we conclude that *the moments of g_i have to vanish*:

$$\int_0^{s_{\max}} g_{i+m+1} s^m ds = 0, \forall i \in 1, ..., n-2 \text{ such that } i+m+1 < n$$

In particular, if we take $i = n-(m+1)$, we get:

$$\int_0^{s_{\max}} g_n s^m ds = \int_0^{s_{\max}} u s^m ds = 0, \ m = 0, ..., n-2.$$

Similarly, if we start with the value $g_0(s_{\max}) = 1$, we obtain an extra condition on the moment of u:

$$1 = g_0(s_{\max}) = \int_0^{s_{\max}} g_1 ds = -\int_0^{s_{\max}} g_2 s ds = \ldots = (-1)^{n-1} \int_0^{s_{\max}} u s^{n-1} ds$$

As sign(u)=sign(ξ_{n-1}) and ξ_{n-1} is a polynomial of degree $n-1$ there can be no more than $n-1$ sign changes for u in the interval $[0, s_{\max}]$. Let $0 = d_0 \leq$

... $\leq d_{n-1} \leq d_n = s_{\max}$ be the points of sign change of u, possibly coinciding. One can easily see that with less than $n-1$ switching points the moment conditions cannot be satisfied. To summarize, we obtain the following moments problem for the function u:

Find n points d_i, $i = 1 \ldots n-1$ such that the function $u(t)$ taking values ± 1, changes sign at points d_i, and first $n-1$ moments of u vanish, and the n-th moment is equal to 1*.*

All moment integrals of u on each interval can be computed explicitly, so this formulation immediately leads to an algebraic system of equations.

As a simple example, consider the case $n = 3$. In this case, we have 2 switching points d_1 and d_2. The moment conditions of order 0 and 1 are:

$$d_1 - (d_2 - d_1) + (d_3 - d_2) = 0,\ d_1^2 - (d_1^2 - d_2^2) + (d_3^2 - d_2^2) = 0.$$

Solving this system with respect to d_1 and d_2 yields $d_1 = d_3/4$ and $d_2 = 3d_3/4$. The second moment condition takes the form:

$$\frac{1}{16} d_3^3 = 1;$$

that is, $s_{\max} = d_3 = (16)^{1/3}$ in this case, and the optimal W^3_∞-norm of f is 16. Corresponding spline solution $f(t)$ is a non-uniform spline of degree 3 with knots at $0, 1/4, 3/4, 1$.

Recursive form of the solution of the moments problem for general n. For $n > 3$, the moments problem leads to algebraic equations of high degree, but of a special form. [107] provides an algorithm for finding $d_n = s_{\max}$ and switching times d_i for a more general problem with $f^{(i)}(s_{\max}) = x_i$, $i = 0, \ldots n-1$.

Following [107], let

$$G_k = \frac{1}{2}[s^k + (-1)^{k+1}\tilde{u}kx_{n-1}], k \in \{1, \dots, n\},$$

where $\tilde{u}$ is the control value (+1 or -1) on the last interval $[d_{n-1}, d_n]$.

Then the functions $\gamma_i(s, \tilde{u})$ are defined recursively as

$$\begin{aligned}\gamma_0 &= -1\\ \gamma_1 &= \frac{s + \tilde{u}x_{n-1}}{2}\\ \gamma_k &= \frac{1}{k}(G_k - \sum_{i=1}^{k-1} \gamma_i G_{k-i}), k \in \{2, \dots, n\}\end{aligned}$$

The *Hankel determinant* for a vector $[a_0, \dots a_n]$ is the determinant of the matrix $[\gamma_{i+j-q}]_{i,j=1}^{p}$, where $p = \lceil n/2 \rceil$ and $q = 2p - n$.

[107] proves the following theorem:

Theorem 2.2. *Define $s_{\max}$ as the maximal real root among the roots of $\Delta_n(\gamma(s, -1)) = 0$ and $\Delta_n(\gamma(s, 1)) = 0$ $\tilde{u}$ is $+1$ if $s_{\max}$ is a root of the first equation, -1 if it is the root of the second equation, and switching points are the roots of*

$$\sum_{l=1}^{n} t^{l-1} \sum_{k=l}^{n} \frac{\partial \Delta_n}{\partial \gamma_k}(\gamma(s_{\max}, \tilde{u}))\gamma_{k-l}(s_{\max}, \tilde{u}) = 0$$

Explicit solution of the moments problem. In our case, $s_{\max}$ can be found explicitly and the equation for switching times can be written in a more explicit form. For our problem, the initial conditions have a simple form $x_0 = 1$, $x_i = 0$, for $i \in \{1, ..., n-1\}$.

Setting $\gamma_0 = -1$, and using explicit expressions for G_k, the formulas for γ_k reduce to

$$\gamma_k = \frac{-1}{2k}\sum_{i=0}^{k-1}\gamma_i s^{k-i}, \text{ for } k = 1, ..., n-1$$

$$\gamma_n = \frac{1}{2n}(-1)^{n+1}\tilde{u}n! - \frac{1}{2n}\sum_{i=0}^{n-1}\gamma_i s^{n-i}$$

Multiplying the formula for γ_{k-1} by s and subtracting from the expression for γ_k we obtain

$$\gamma_k = \frac{2(k-1)}{2k}s\gamma_{k-1} - \frac{1}{2k}s\gamma_{k-1} = \frac{2k-3}{2k}s\gamma_{k-1},$$

which leads to explicit formulas for γ_k:

$$\gamma_k = \frac{(2k-3)!!}{(2k)!!}s^k, \qquad \text{for } k \in 1, ..., n-1,$$
$$\gamma_n = \frac{(2n-3)!!}{(2n)!!}s^n + \frac{(-1)^{n+1}\tilde{u}}{2}(n-1)!$$

To obtain the equations for $s_{\max}$ in explicit form, we eliminate the dependence on s from Hankel determinants. If we multiply the i-th row by s^{p-i+1}, we obtain a matrix where all the i-th column entries are of the form $C_{ji}s^{p+1-q+i}$ where C_{ji} are constants independent of s. As multiplying a row or a column by a number multiplies the determinant by the same number, we get:

$$\Delta_n(\gamma(s,\tilde{u})) = s^{(p+1-q)p}\Delta_n(\gamma(1,\tilde{u}))$$

In vector form, we can write $\gamma(s,\tilde{u}) = \gamma(s,0) + [0,0,...,0,((-1)^{n+1}\tilde{u}(n-1)!)/2]$, where the non-zero entry is in the n^{th} position. As the determinant is linear with respect to row addition we obtain:

$$\Delta_n(\gamma(s,\tilde{u})) = s^{(p+1-q)p}\Delta_n(\gamma(1,0)) + s^{(p-q)(p-1)}\Delta_{n-2}(\gamma(1,0))(\frac{1}{2}(-1)^{n+1}\tilde{u}(n-1)!)$$

Discarding 0 and negative roots we get a unique solution when a real (non-zero) solution exists.

$$s = (\frac{\Delta_{n-2}(\gamma(1,0))}{2\Delta_n(\gamma(1,0))}\tilde{u}(-1)^n(n-1)!)^{\frac{1}{n}}$$

It is clear to see that for the solution to exist, $\tilde{u} = (-1)^n$. Therefore, we obtain the following expression for

$$s = (\frac{\Delta_{n-2}}{2\Delta_n}(n-1)!)^{\frac{1}{n}}.$$

To make this expression fully explicit, we need to compute Δ_{n-2}. We observe that

$$\gamma_k(1,0) = \frac{(2k-3)!!}{(2k)!!} = \frac{(2k-2)!}{2^{k-1}(k-1)!} \cdot \frac{1}{2^k k!} = \frac{1}{2^{2k-1}} \cdot \frac{(2k-2)!}{(k-1)!k!} = \frac{2}{2^{2k}} C_{k-1}$$

where C_{k-1} is the $(k-1)^{th}$ Catalan number.

Using the same approach we used to remove s, we get

$$\Delta_n = 2^{-2(p+1-q)p+p} \det[C_{i+j-q-1}]_{i,j=1}^{p}$$

This shows that the problem reduces to computing determinants of Hankel matrices of Catalan numbers with $C_0 = 1$. It is known that these determinants are all equal to 1 ([146]). Thus

$$\frac{\Delta_n}{\Delta_{n-2}} = 2^{-2(p+1-q)p+p-(-2(p-q)(p-1)+p-1)} = 2^{-2n+1}$$

and

$$s = (2^{2n-2}(n-1)!)^{\frac{1}{n}}$$

Remember that $s_{\max} = (\max(W))^{(1/n)}$. Therefore $W = s^n = (2^{2n-2}(n-1)!)$. Given W, the solution of the original problem is obtained by integration of W(t) over $[0,1]$ n times and is a spline of degree n, continuity C^{n-1}.

2.7.2 Minimizing W_2^d Norm

In this case, the derivation is simpler, as the problem can be cast in the classical Euler-Lagrange rather than PMP framework. In this case, we minimize

$$\int_0^1 f^{(n)}(t)^2 dt$$

subject to zero conditions on derivatives of order up to $n-1$, and $f(0) = 0$ and $f(1) = 1$. Applying standard integration-by-parts to the first variation of this functional, we obtain the Euler-Lagrange equation

$$f^{(2n)}(t) = 0$$

that is, the optimal solution is a polynomial of degree $2n-1$. $2n$ boundary conditions define this polynomial uniquely, and an explicit expression for this polynomial can be obtained by observing that its derivative $g(t) = f'(t)$ is a polynomial of degree $2n-2$ satisfying $2n-2$ homogeneous boundary conditions $g^{(i)}(0) = g^{(i)}(1) = 0$. $g(t)$ has $2n-1$ coefficients and is defined by the boundary conditions up to a constant, and as it is easy to see that $t^n(1-t)^n$ satisfies the desired boundary conditions, and, therefore, all possible $g(t)$ are of the form $Ct^n(1-t)^n$.

The function $f(t)$ can be obtained by integrating $g(t)$ over the interval $[0 \dots t]$, and computing the constant C using the condition $f(1) = 1$. This leads to the formula

$$\begin{aligned} \tilde{f}(t) &= \sum_{i=0}^{n-1} \frac{(-1)^i}{i+n} \left(\begin{array}{c} n-1 \\ i \end{array} \right) t^{i+n} \\ f(t) &= \frac{\tilde{f}(t)}{\tilde{f}(1)} \end{aligned} \tag{2.14}$$

where $\tilde{f}$ satisfies $f(t) = C \cdot \tilde{f}(t)$.

At this point a comparison of all partition of unity functions are in order. We call the function that minimizes W_∞^d the *optimal partition of unity function for degree d*. We refer to the function computed in this section which minimizes W_2^d, the *L_2-optimal partition of unity function for degree d*. Other functions we have considered are the C^∞ partition of unity function from Section 2.3, and the spline basis function of degree n (smoothness C^{n-1}) from Section 2.6.1.

In Figure 2.12 we provide a series of plots comparing these functions. In (a) we plot the partition of unity for optimal, L_2 optimal, spline and C^∞ blending functions. For this image, only for illustration purposes, we picked degree 3 as an example. In (b)-(f) we provide plots of 1st through 5th derivatives of all these functions,where one can observe clearly the differences in derivative magnitudes. It is important to note that for the plot of the k-th derivatives, the graphs for the optimal functions are for functions with optimal k-th derivative (e.g. the L_2 optimal is of degree $2k-1$).

2.8 Flexibility

A k-flexible surface is one where one can assign position and derivatives up to degree k to any point on the surface. [195] claim that their construction yields surfaces that are at least 3-flexible at control vertices. We claim that this construction yields surfaces that are at least 2-flexible at any point on the surface.

The function that describes the local surface on a chart can be expressed as:

$$f_x(u,v) = B^T(u,v)Up^{(2)} \tag{2.15}$$

where $p^{(2)}$ are the set of data points after two steps of subdivision. Including the subdivision matrix S, we can re-write equation (2.15) in terms of the control

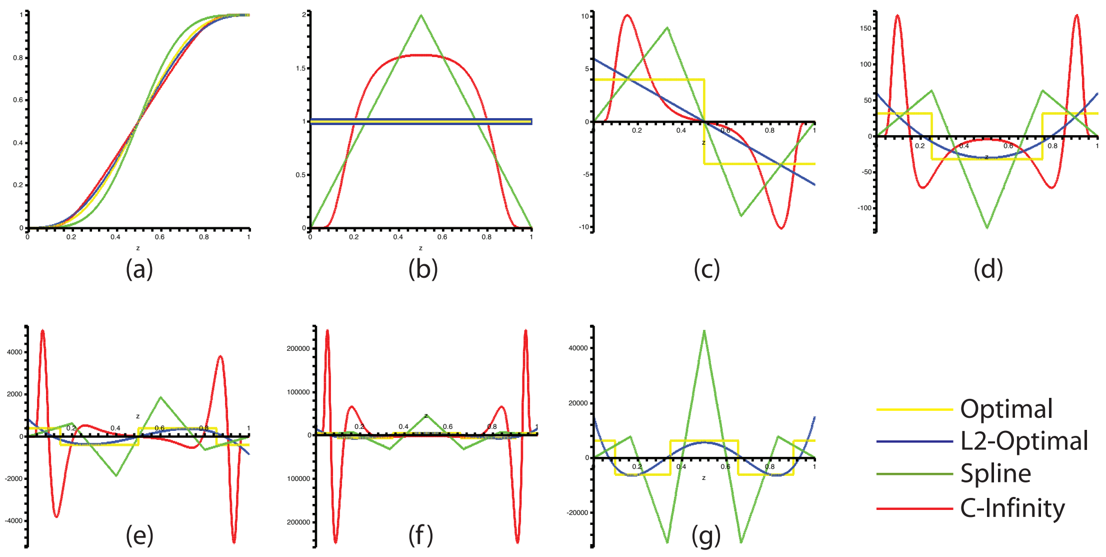

Figure 2.12: Comparison of different partition of unity functions. (a) Function plots (b)-(f): First through fifth derivative plots. (g) Fifth derivative with the plot for the C^∞ function omitted for clarity.

vertices p.

$$f_x(u, v) = B^T(u, v)USp \tag{2.16}$$

This can be viewed as some basis function times the control vertices near point (u, v). Note that in the global case a point (u, v) is affected by several charts and partition of unity functions. This is not an issue if the point (u, v) coincides with a control vertex since there is only one chart that describes the geometry at that point, and there is no need to deal with partition of unity functions. This is the main difference between our flexibility results and that of [195].

To test the 2-flexibility of the surface at a given point we need to solve a linear system of the type $Ax = b$, where x is the set of control vertices we solve for, b is the column vector of position and derivatives that we wish to assign, and A is a square matrix that has the evaluation of position and derivatives of a set of basis functions blended together with blending functions.

Since we claim 2-flexibility, our matrix A has 6 rows. (One row for position, two for first derivative and three for second derivative). We show that this matrix A for any point of a face is non-singular such that the system has a solution. Since exact computation could be an issue we look for sign changes on the determinants of A for a given face. The general structure of our algorithm for gathering these determinants is the following:

1. Generate a set of 10 basis meshes (see Section 2.8.1).
2. Construct a manifold-based surface for each mesh.
3. Given a specific face of a mesh, generate a set of equally spaced samples (i, j) per face. For each one create a matrix M_{ij} that is 6x10. This is done by evaluating the position and derivatives of the given point on each one of the ten manifold surfaces.
4. Compute determinants of all 6x6 submatrices of M_{ij} to find one that does not cause a sign change.

2.8.1 Mesh Generation

We use a set of meshes as our basis functions. We generate the meshes using the following algorithm:
1. Create a base star in the plane.
2. Lift the star to 3D.
3. Subdivide.

1. Base Star. Base star is a mesh that has a face f surrounded by *all* control vertices that affect that face in the manifold surface. When we test for flexibility at point (u, v) we require that it resides on the face f.

When testing a point on a face with one extraordinary vertex our base star is the following, although it is not the minimal possible: Take two rings around an extraordinary vertex to form the curved star shape in the plane. Connect the points forming these rings with straight lines. This forms the "base star" (see Figure 2.13) with point coordinates (x_c, y_c).

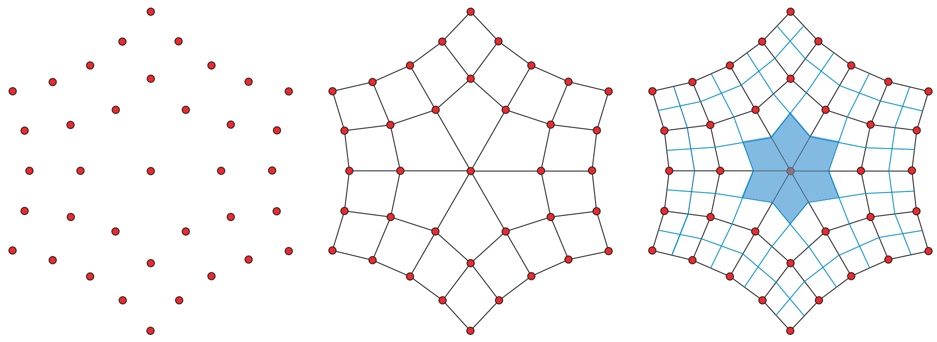

Figure 2.13: Left: Points (x_c, y_c) of the two ring. Middle: Points connected by straight lines. Right: After subdivision with faces valid for flexibility check shaded.

For a face surrounded by four extraordinary vertices, we have a different construction for the the base star. We start with a mesh as the one on the left of Figure 2.14. We then map the vertices of this mesh to the curved star taking one of the vertices as the main vertex which we colored differently in the figure. Then we connect these points with straight lines and we get the new base star (See Figure 2.14 Middle).

2. Lift. In order to generate independent meshes using the same base star, we lift the star using a polynomial basis up to third-order for the z coordinate, resulting in 10 meshes. These 10 meshes end up with the following coordinates: $(x_c, y_c, 1)$, (x_c, y_c, x_c), (x_c, y_c, y_c), $(x_c, y_c, {x_c}^2)$, $(x_c, y_c, x_c y_c)$, $(x_c, y_c, {y_c}^2)$, $(x_c, y_c, {x_c}^3)$, $(x_c, y_c, {x_c}^2 y_c)$, $(x_c, y_c, x_c {y_c}^2)$, $(x_c, y_c, {y_c}^3)$.

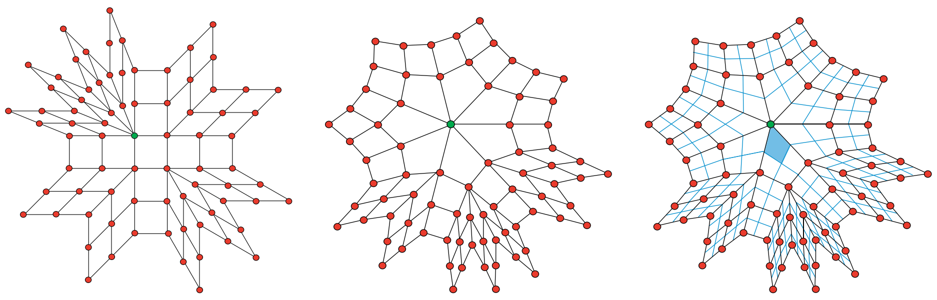

Figure 2.14: Construction of the second type of base star for valence set {7,5,6,5}.

3. Subdivide. Face f should have no influence from outside the base star. This requires f to be at least three faces away from the boundary of the mesh. To achieve this distance in the presence of one extraordinary vertex we apply one step of subdivision and choose one of the faces around that as the face f. See Figure 2.13 (right). In the case with four extraordinary vertices, we choose the face shaded in Figure 2.14 (right).

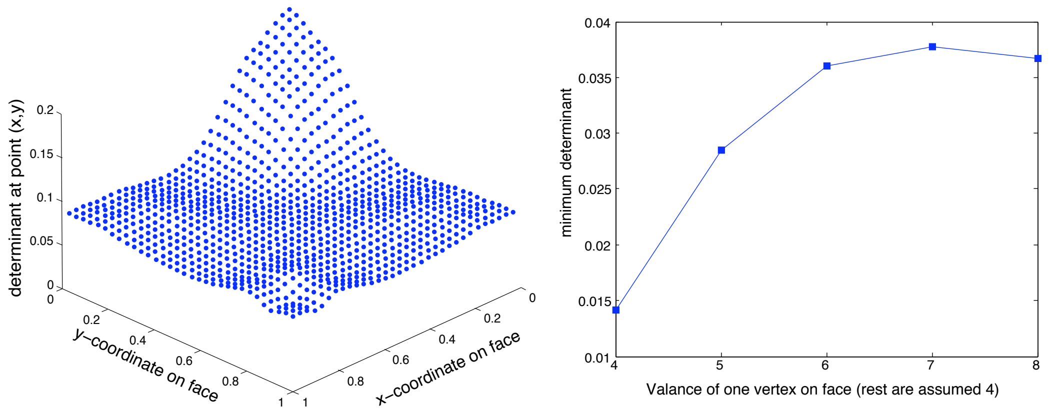

Figure 2.15: Left: Determinant distribution on a face f with one valence six vertex at (0,0). Right: Plot of minimum determinant on a face as a function of valance.

2.8.2 Flexibility Results

For any face with only one extraordinary vertex on it, the determinants stay above zero for valences up to 8 using the first submatrix that uses the first six columns, i.e. the bases up to degree 2 (Figure 2.15, right). For a face with no extraordinary vertex (all four vertices around the face have valance four) the resulting surface of the determinants is actually a plane parallel to and above $det(A) = 0$. For all other valances the resulting surface is close to flat with an elevation near the extraordinary vertex. The magnitude of this spike increases with the valance of the extraordinary vertex (See Figure 2.15, left). This concludes that for faces with at most one extraordinary vertex our surface satisfies the 2-flexibility property.

For the more complicated case with four extraordinary vertices we got very similar results. We tested for 10,000 combinations of valances for the four vertices surrounding the face f. The plots are close to flat with elevations near the "main" extraordinary vertex. Therefore as long as two extraordinary vertices are at least two faces apart, the effect of one extraordinary vertex diminishes near the other one, not causing any singularities in the matrix A of the flexibility system, resulting in a 2-flexible surface. Even if this requirement is not met in the input mesh, it is easy to correct by applying one step of subdivision. Up to valance 13, the submatrix that uses the first six columns returned positive determinants. For valances higher than that a band of negative determinants forms in the plot of the face near the extraordinary vertex if the first six columns are used. In that case we check for other submatrices using a different set of six columns. In all cases we could successfully find other submatrices where the positivity of the determinants are maintained. This holds true for valances up to 20.

2.9 Results

In this section we comment on the results of our implementation of the aforementioned methods. Figure 2.16 (left) demonstrates the behavior of piecewise smooth boundaries for convex and concave corners. It also displays the surface behavior in the presence of boundaries dependent in the interior and independent boundaries. Although an independent boundary on the convex case is more likely to be desirable, in the case of the concave corner it may not be so. Due to the limitation of concave corners mentioned previously, the deformation of the surface near the corner may be undesirable. The right side of the same image displays the behavior of convex and concave corners on a C^3 surface as well as a C^∞ surface with smooth boundaries in order to demonstrate the boundary behavior for different orders of smoothness.

Figure 2.17 displays the dependency of the quality of our surfaces on the underlying mesh. The irregularity of the reflection lines shows that the surfaces based on meshes with faces of high aspect ratios seem to behave not as well as the Catmull-Clark subdivision surfaces (above). Although close examination shows that our surfaces look better near extraordinary vertices (on the boundary (Figure 2.22) or interior), for overall quality one may need to improve the mesh as a preprocessing step before the application of our algorithm.

In Figures 2.18, 2.19, 2.20 and 2.21 we provide images of derivative magnitudes up to order five under our parameterization near extraordinary vertices. Figure 2.18 shows the behavior of derivatives near an interior vertex of valance 6 for a C^∞ surface(above) and C^5 surface(below). In the following images, we provide the distribution and magnitude of derivatives for extraordinary vertices on smooth boundaries (Figure 2.19), convex corners (Figure 2.20) and concave corners (Figure 2.21) for C^∞ and C^5 surfaces. Although the derivative magnitudes grow as the order increases in both cases, the magnitudes of C^5 boundary vertices are much less than the values of their counterparts on the C^∞ surface. This property demonstrates one reason for opting for a C^k construction as opposed to a C^∞ one.

Figure 2.22 shows the appearance of discontinuities near an extraordinary boundary vertex with increasing valance in the case of a Catmull-Clark surface, as opposed to a C^4 and a C^∞ surface where no discontinuities can be seen. The last figure shows a side to side comparison of C^∞ and C^3 surfaces in order to present that our construction of C^k surfaces results in as visually pleasing results as the C^∞ surfaces.

We also provided a close-up view of boundary behavior near corners in all cases (Figure 2.24). The distortions are quite clear one the subdivision surface. One can also see the general shape differences near the concave corners.

In Figures 2.25 and 2.26 we study a surface with an interior twist adjacent to the boundary (convex surface becomes concave). The subdivision surface displays a sharp change in lighting at the boundary connection and this change is absent in both C^d and C^∞ surfaces modeled by our construction.

2.10 Conclusions

We have presented methods for constructing surfaces of arbitrary prescribed smoothness with support for piecewise smooth boundaries of same order from meshes. Our methods are manifold-based where each vertex is associated with a chart. The resulting surfaces follow closely the subdivision surfaces resulting from the same meshes. Our resulting surfaces have desirable properties such as explicit parameterization, flexibility, linear dependence on control points and high visual quality.

Having observed that such methods lead to a growth of derivative magnitudes with order, we derived lower bounds. Each of our methods is fully implemented and our software is available.

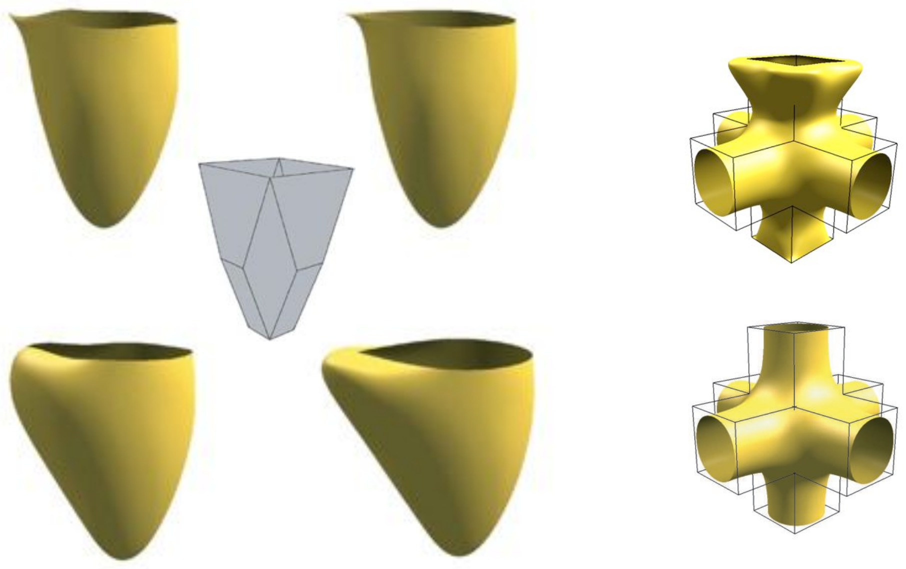

Figure 2.16: Left: Comparison of corner behavior in the case of interior-dependent (left) and independent (right) boundaries. Right: A C^3 surface with corners (above) and a C^∞ surface with smooth boundaries (below).

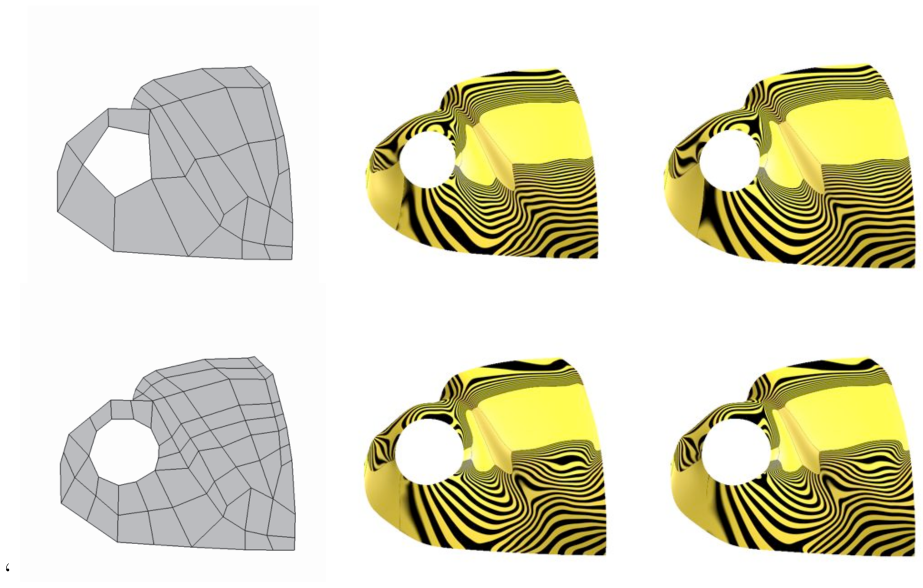

Figure 2.17: Mesh dependency of our method. Surfaces on the left are the manifold surfaces as opposed to Catmull-Clark surfaces on the right.

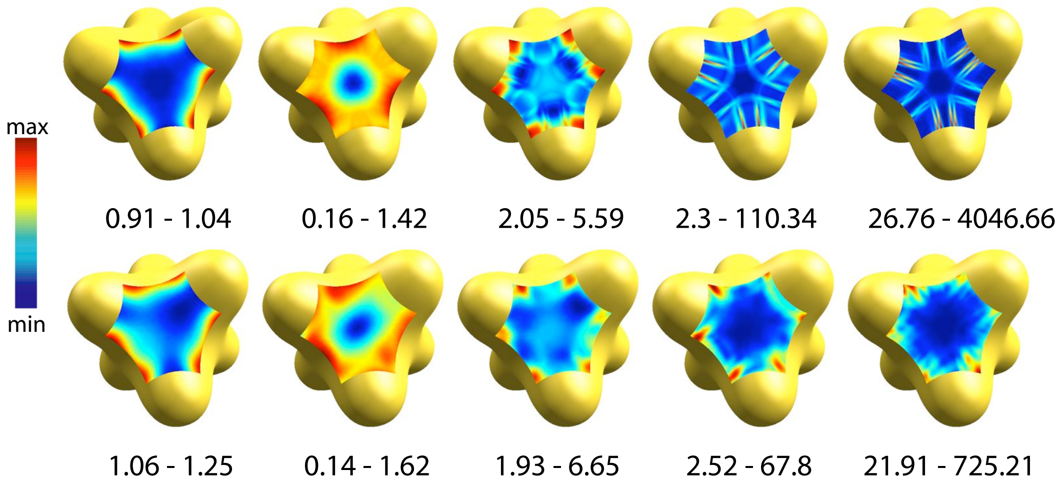

Figure 2.18: Distribution of derivative magnitudes for a C^∞ and a C^5 closed surface.

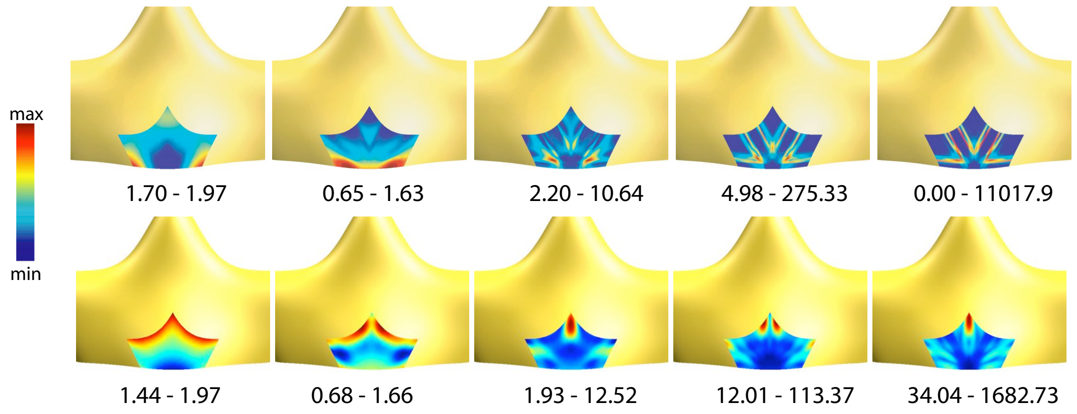

Figure 2.19: Distribution of derivative magnitudes for a C^{∞} and a C^5 surface with boundary.

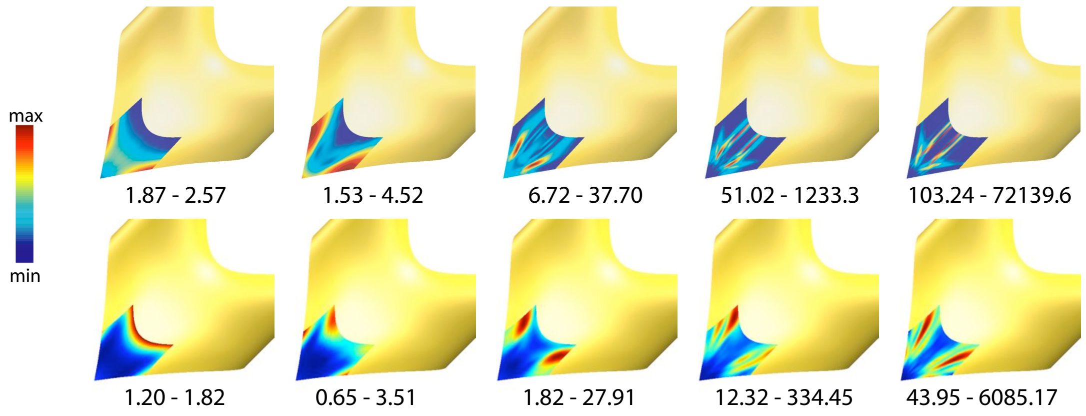

Figure 2.20: Distribution of derivative magnitudes for a C^{∞} and a C^5 surface with a convex corner.

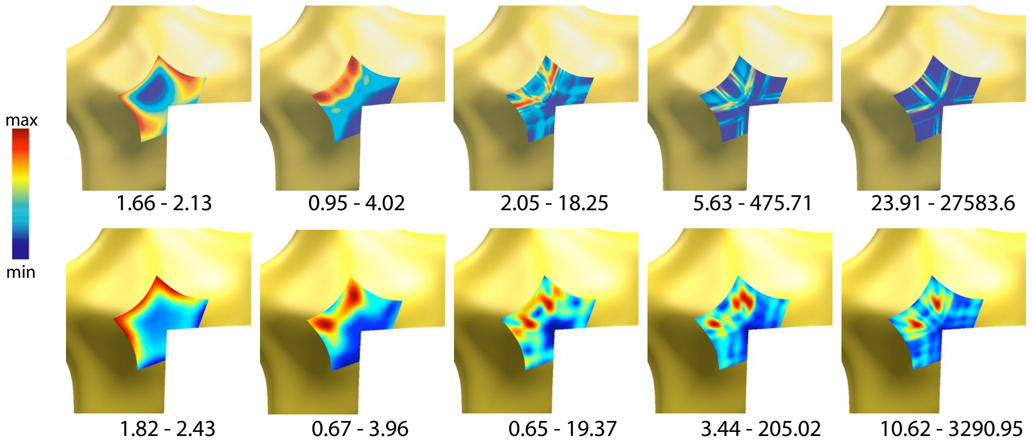

Figure 2.21: Distribution of derivative magnitudes for a C^∞ and a C^5 surface with a concave corner.

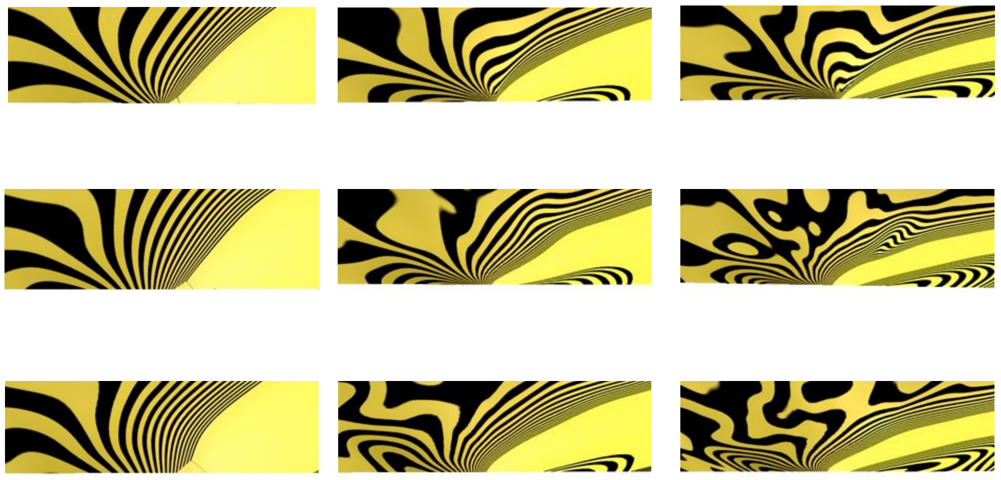

Figure 2.22: Top to bottom: Catmull-Clark, C^4, C^∞ surfaces. Left to Right: Behavior near valance 3, valance 5, valance 7 boundary vertices.

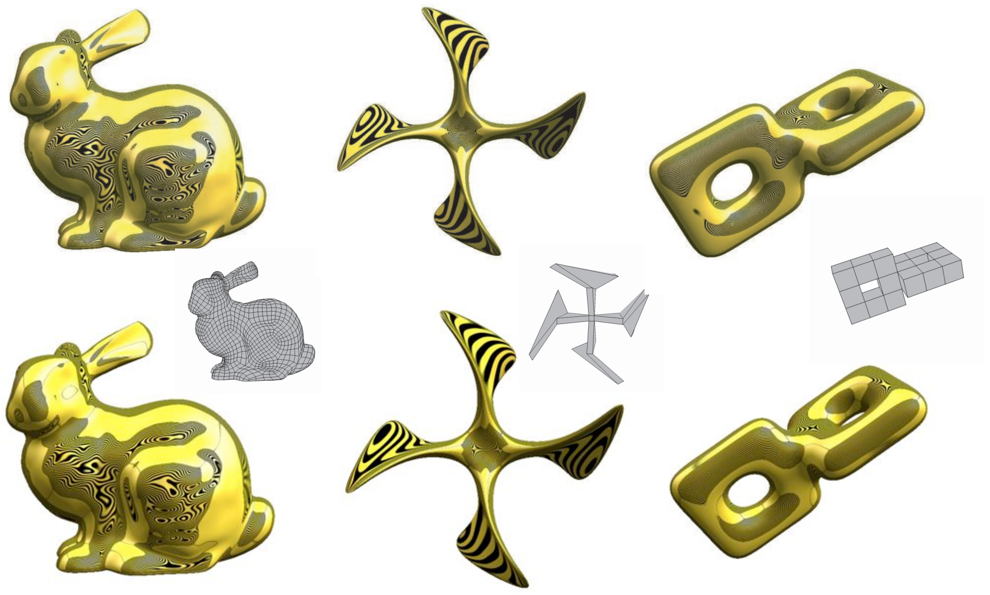

Figure 2.23: C^{∞} (above) vs C^3 (below) surfaces.

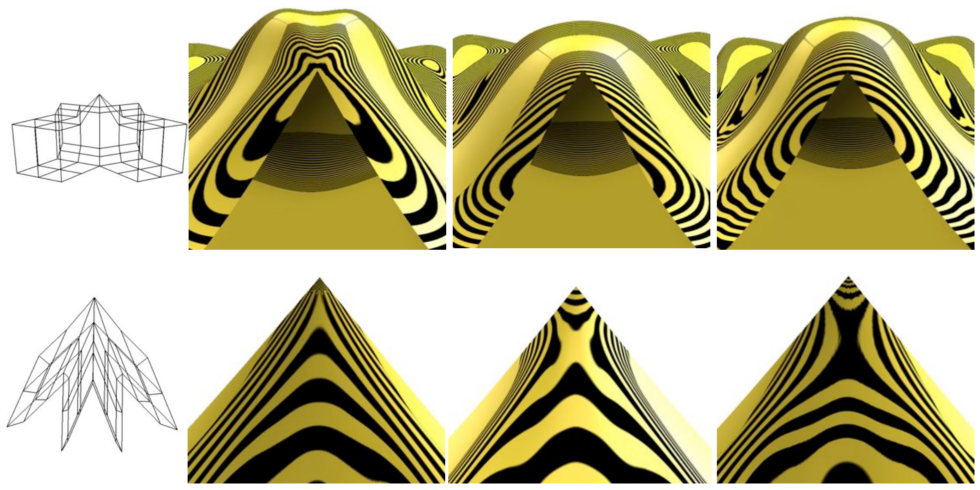

Figure 2.24: A close-up of a concave and convex corner vertices of valance 4. Left to Right: Catmull-Clark, C^2, C^{∞}.

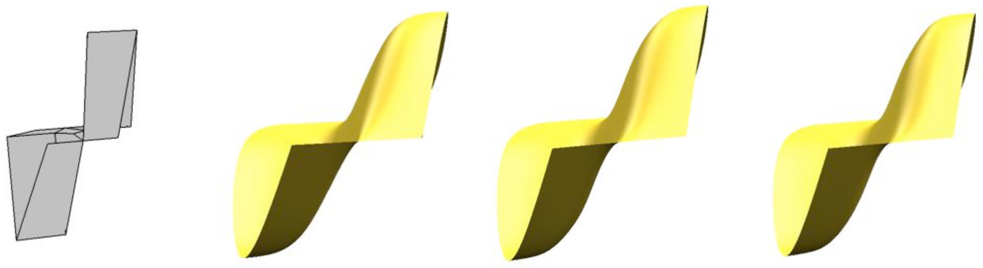

Figure 2.25: Surface behavior near a twist at the boundary vertex of valance 4. Left to Right: Catmull-Clark, C^2, C^∞. Note the sharpness of light on the Catmull-Clark surface near the twist and its absence on our surfaces.

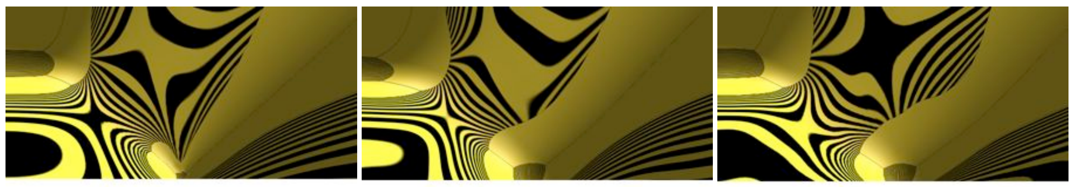

Figure 2.26: Same twist on the boundary as seen from above. Left to Right : Catmull-Clark, C^2, C^∞.

3 Shape Optimization Using Reflection Lines

An alternative to using high-order surfaces as described in Section 2 is to define an approximation to high-order quantities for meshes, with high-order surface implicit. In this and the following chapter we work in the highly refined mesh setting.

As we have seen in Section 1.1 there are numerous methods of surface quality assessment that have been proposed to make defects on a surface more apparent. Although the methods proposed look for a variety of imperfections on a surface, most methods display discontinuities in first and second differential quantities. Reflection line visualization is one of the most commonly used methods for this purpose.

Detected surface imperfections can be remedied in a variety of ways. Surface fairing as covered in Section 1.2.3 is one of the commonly used methods. But this may require further interrogation steps in order to confirm that enough fairing has been applied. On the other hand an optimization based technique, more specifically an energy minimization, would yield a surface that does satisfy the constraints as required by the functional, therefore will eliminate the need for further interrogation.

In this chapter we propose a method that is based on the variational principle reviewed in detail in Section 1.2. As the reader may recall, there is a wide variety of functionals proposed for use in a variational setting. We utilize a second-order functional namely one that depends on the gradient of reflection lines. We experimented with a very efficient first-order functional as well but the C^1 continuity at joints proved it less useful, as is the case with all such low-order functionals (See Section 1.2 for examples of low-order functionals).

The minimization of this reflection-based second-order functional is equivalent to solving a fourth-order PDE. Therefore, by solving this minimization problem on a mesh we are effectively discretizing and solving a fourth-order PDE. Our construction of this functional falls into the discrete-geometric category of PDEs that was mentioned in the introduction. In our discretization of the differential quantities we do in fact use linear finite elements as will be explained in the following.

3.1 Introduction

Many man-made surfaces have highly reflective finishes: cars, kitchen appliances, lamps and jewelry are common examples. The appearance of such objects is primarily defined by the reflections of other objects. Reflections are quite sensitive to surface shape and depend on local surface quantities (normals and curvatures) as well as viewer location.

As we have mentioned above and in Section 1.1, reflection *lines* are a widely used interrogation tool for surfaces. Another way to think about reflection lines is to consider them a special type of reflected environment, capturing the distortion introduced by the curved shape of the surface for a particular direction of features in the environment. Reflection lines are widely used in the automotive industry, and were also found to be useful in biomedical engineering as a tool for cornea shape reconstruction [80]. Advances in graphics hardware made interactive reflection line rendering widely accessible and easy to implement.

As mentioned before the process of evaluating surface quality is complimentary to shape design. In most cases, the designer defines the surface by manipulating

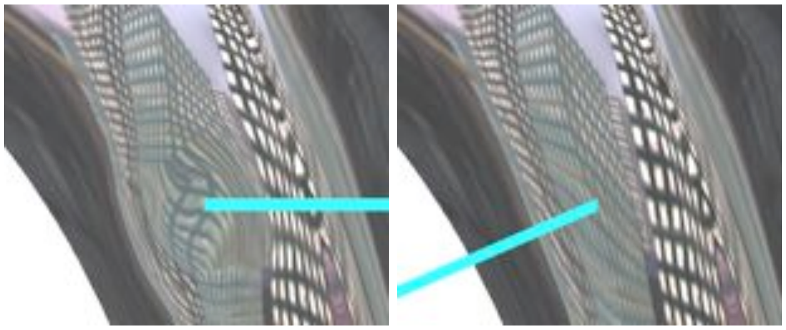

Figure 3.1: An example of reflection line optimization.

spline or subdivision control points, or other types of shape controls, then evaluates the quality using interrogation tools and repeats the process until the desired quality is achieved. The controls of the shape have an indirect effect on the quality measure and in the case of reflective surfaces, it may be hard to guess how the shape should be modified to achieve a desired effect. We take the alternative approach of formulating the surface editing problem as the minimization of a quality-based functional measuring deviation from desired behavior. Our formulation is along these lines, and can be thought of as being the opposite of surface interrogation (See Figure 3.2).

Reflection lines provide a convenient framework for building such functionals for reflective surfaces. For reasons we discuss in Section 3.3, arbitrary manipulation of reflections is not possible. Furthermore, recovering the surface from an arbitrarily chosen distortion of a reflected image is not always possible either. However, in most cases, it is possible to find a surface producing a given pattern of reflection lines. By choosing a reflection line direction, the user chooses what feature direction in the environment can be considered most important. For example, horizontal and vertical lines are most common in urban and indoor environments, and it makes sense to use these directions for surface optimization.

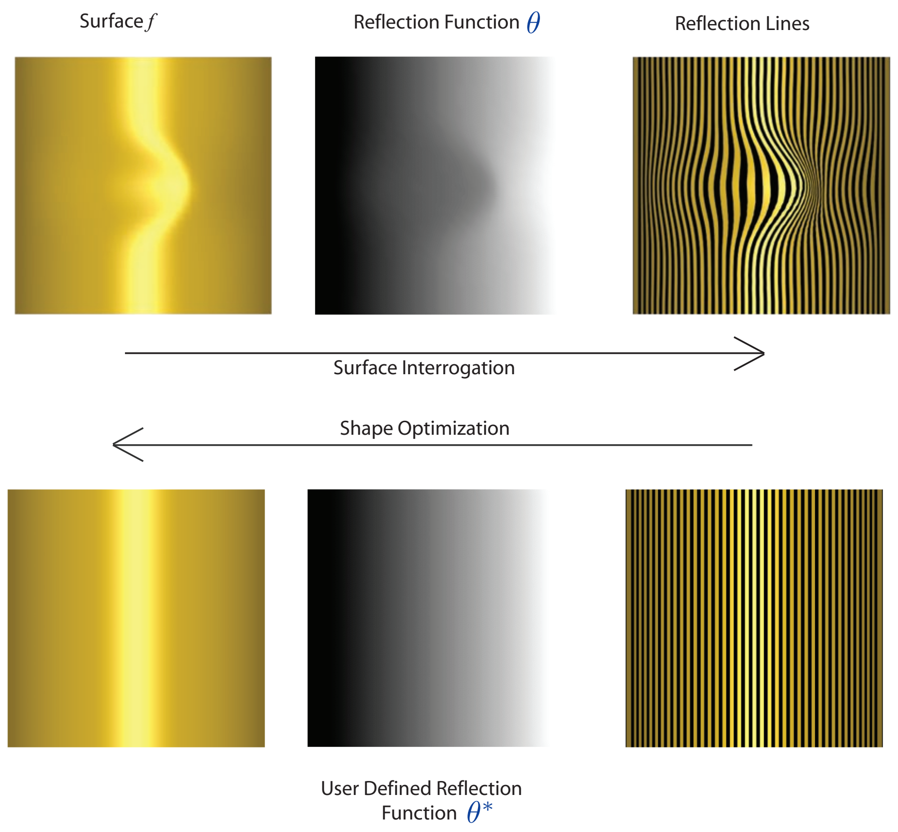
Surface f
Reflection Function θ
Reflection Lines
Surface Interrogation
Shape Optimization
User Defined Reflection Function θ^*

Figure 3.2: Relation of surface interrogation to reflection line optimization.

In this chapter, we present a system for interactive surface modeling based on reflection line manipulation. We show how to discretize reflection lines on arbitrary meshes, and demonstrate that a relatively simple discretization is sufficient for shape modeling purposes, provided that a suitable normal estimation algorithm is used. We propose a numerical technique for solving the problem at rates adequate for interactive surface manipulation. Our method is based on two insights. First, an arbitrary mesh can be locally parameterized over the image plane away from silhouette edges; this makes it possible to reduce the number of degrees of freedom used in optimization, and greatly simplifies expressions. Second, we observe that a simple triangle-based discretization of second-order quantities using only vertex degrees of freedom can be used to compute second-order derivatives of the surface parameterization, which leads to a fast and efficient matrix assembly. We demonstrate how reflection line manipulation can be used to smooth and warp reflection lines, change reflection line density, and create surfaces with a desired reflection line pattern.

3.2 Previous Work

Reflection lines are extensively studied in the geometric modeling literature. Some of the earliest work is described in [100], where a differential-geometric description of reflection lines and an analytic expression for the variation of these lines is derived. [99] describes a technique for adjusting families of spline curves defining a surface based on reflection line changes. We have already described how reflections lines are used as a surface interrogation tool in Section 1.1. Functionals similar to reflection-line functionals are common in shape-from-shading literature (e.g. [89]); however, the goal there is to reconstruct an unknown surface entirely from possibly noisy image data, rather than to modify an existing mesh.

Various types of *fairing* functionals for surfaces and their discretizations for spline surfaces, subdivision surfaces, and meshes are considered in [29,104,157,181] and many other papers. We provided a survey for such functionals and their discretizations in Section 1.2. Fairing functionals are used increasingly in interactive surface deformation settings, in particular for general meshes (e.g. [18]; see [20] for an excellent survey). An efficient, robust and accurate discretization of the Hessian of the function defined on an arbitrary mesh is central to our work. This problem is closely related to the problem of defining shape operators on meshes.

Discrete geometry approaches (e.g. [38,121,141]) play an important role in making variational techniques for meshes sufficiently fast for interactive applications. [66] contains a detailed survey of different techniques; our discretization builds on [54].

Using user-defined reflection fields for surface optimization is similar to gradient and Laplacian deformation techniques ([163], [200]) in that the optimization functional depends on the initial mesh geometry (in our case, through the reflection function).

3.2.1 Surface Interrogation as a Design Tool

There is limited background on the use of interrogation methods as design tools. Since we are proposing to use reflection lines as a design tool we will provide a short summary of this type of work in literature.

Loos, et al. [115] provide two variational methods of surface design: one based on isophote approximation errors and the other based on reflection line errors. In both cases, the surface is assumed to be the graph of a function, namely $F(u, v) = (u, v, f(u, v))$. In the case of isophotes, the idea is the following. Given an intensity function I^*, isolines of the intensity function of a given surface I^f are as close as possible to those of I^*. The error function used is :

$$\begin{aligned} J(f) &= \int (\nabla I^f - \nabla I^*)^2 \\ &= \int ({I^f}_u - {I^*}_u)^2 + ({I^f}_v - {I^*}_v)^2 \end{aligned}$$

It is possible to express $J(f)$ as a function of first and second derivatives of f which is necessary for their solver. In the case of reflection lines instead of an intensity function a new function called the “height field of reflection (HFR)” is used to make the fit. The functional $J(f)$ has the same formulation if the HFR were labeled I^f. It is the closed formula of this function and its derivatives that are different. In both cases the functional is approximated by a quadratic based on Taylor expansion and the minimization is set as a linear problem. A different technique for manipulating reflection line shape directly on a NURBS surface is described in [42].

In [3] an isophote-based method is given. The surface in this case can be described by a vector (say, vector of control points for a patch based surface). The idea is to find a displacement vector h such that the new surface defined by $s + h$ fits well to the given set of isophotes. It is a linearized system that needs to be iterated to solve for h with possible addition of other constraints at different iterations until all constraints are satisfied and the given isophotes fit the new surface well. There are also some schemes using highlight lines in constructing fairing functionals [28, 194].

Isophotes has been studied in the context of shape from shading. [118] explores the relation between the isophotes, the underlying object and the light source and proposes a variational method for estimating the surface shape from irradiance information. There is also the description of an unsuccessful attempt to minimize the geodesic curvature along an isophote.

Another isophote based shape from shading tool is explored recently in [44]. The idea is to compute three dimensional object normals using image isophotes. It starts by recovering normals in a small area where it is possible to obtain the estimate from the image. Then the normals are propagated using the isophotes nearby. The accuracy of the method is dependent on the accuracy of the initial normal estimation.

Other interrogation methods have also been used in fairing (not in the variational sense) but those methods do not necessarily result in very smooth surfaces or may require too much effort from the designer. In [100] a reflection line interpolation method based on a PDE is introduced. However, it is almost impossible to have local control on the surface shape and the linearization of the PDE causes errors. This approach only works well in cases where the surface needs to be changed very little. Most other such methods are patch based and consists of recognizing which control points have been problematic and moving them. Note however that moving control points interactively is a very lengthy process. In [90] the control points that cause issues in the polar images are moved or removed. In [91] a method based on based on the Bézout method is introduced for fixing issues that are made visible by orthotomics. In [50], knot removal and re-insertion is used on B-spline surfaces coupled with the use of curvature plots.

The formulation we use is closest to [115], which applies reflection line optimization to B-spline height fields; we apply a similar formulation to general meshes and present algorithms that allow us to achieve interactive performance.

3.3 Reflection Functionals

In this section, we present the relevant basic mathematics of reflections and functionals based on reflection lines. The formulations we use are similar to the ones that were used in [115] for optimization of reflection lines of tensor-product B-spline height fields.

3.3.1 Reflection Line Function

We consider a somewhat simplified formulation of reflection lines, with both the viewer and the light sources located at infinity. The reflection line pattern in our model is created by long line-shaped light sources aligned with a unit length vector **a** (Figure 3.3). Each light source can be identified by a direction in the plane perpendicular to **a**. If we fix a zero direction, each direction corresponds to an angle θ in the range $-\pi \ldots \pi$. For a point **p** of a surface, let **n** be the normal, and let **v** be the view direction, which we assume to be non-parallel to **a** (See Figure 3.3). In this notation, the reflection direction at **p** is given by $\mathbf{r} = (2/|\mathbf{n}|^2)\,((\mathbf{n}\cdot\mathbf{v})\mathbf{n} - \mathbf{v})$. (We do not assume the normal to be unit length).

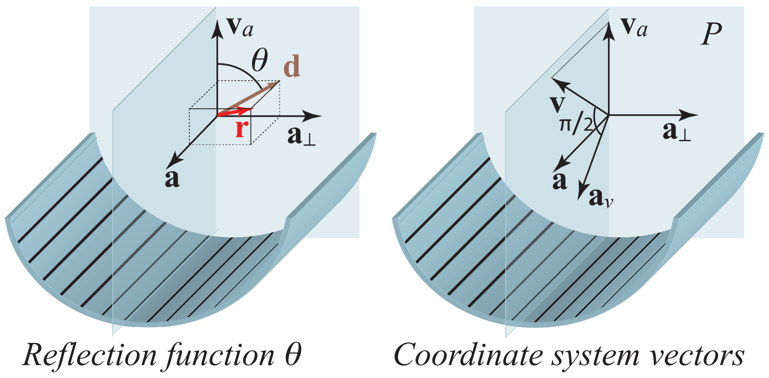

Figure 3.3: Vectors used in the definition of the reflection line function θ.

We define the *reflection line function* to be a scalar function on the surface which assigns to each point the angle θ between a zero direction and the direction **d** to the linear light source corresponding to reflected direction **r**. This direction is obtained by projecting **r** to the plane P perpendicular to **a**: $\mathbf{d} = \mathbf{r} - (\mathbf{r}\cdot\mathbf{a})\mathbf{a}$. Let $\mathbf{v}_a$ be the projection of the viewing direction **v** to the plane P, and let $\mathbf{a}_\perp$ be perpendicular to $\mathbf{v}_a$ in the plane P,

$$\mathbf{v}_a = [\mathbf{v} - (\mathbf{v}\cdot\mathbf{a})\mathbf{a}]_{norm}, \quad \mathbf{a}_\perp = \mathbf{a}\times\mathbf{v}_a$$

where $[\cdot]_{norm}$ denotes normalization (See Figure 3.3). We use $\mathbf{v}_a$ as the zero direction; in this case the reflection line function is given by

$$\theta = \arctan((\mathbf{r}\cdot\mathbf{a}_\perp), (\mathbf{r}\cdot\mathbf{v}_a)) \tag{3.1}$$

where $\arctan(y, x)$ produces values in the range $-\pi \dots \pi$. A reflection line is defined by a constant θ value. As reflection lines are view-dependent, it makes more sense to consider them as functions on the image plane, rather than the surface itself. The function is defined everywhere except at the points where $\mathbf{r}$ is parallel to the light direction $\mathbf{a}$. The gradient of θ is of primary importance: the direction of the reflection lines is perpendicular to $\nabla\theta$, and $|\nabla\theta|$ measures the local density of reflection lines.

3.3.2 Coordinate Formulation

One of the properties distinguishing the reflection line optimization from most fairing problems is that there are fixed spatial directions $\mathbf{a}$ and $\mathbf{v}$ which are a part of the problem formulation. Projections to these directions are natural choices of variables. We observe that for *silhouette* points, for which the normal is perpendicular to the view direction (See Figure 3.4), perturbing the surface does not affect the reflection line function corresponding to these points; i.e. one cannot optimize the lines near the silhouettes without moving the silhouettes which is best done by techniques of the type described in [132]. This suggests that projection to the image plane leads to the natural parameterization for the problem, as in this case silhouette points will form boundaries for optimization regions.

We choose the coordinate system aligned with the image plane ($\mathbf{a}_\perp$, $\mathbf{a}_v$, $\mathbf{v}$), where $\mathbf{a}_v$ is the normalized projection of $\mathbf{a}$ to the plane perpendicular to $\mathbf{v}$ (Figure 3.3). The coordinates along the three axes are x, y, z: we use the standard convention for y to be perpendicular to the image, x to be horizontal, and z to be the view direction. For a vector $\mathbf{t}$, we denote $(\mathbf{t} \cdot \mathbf{a}_\perp) = t^x$, $(\mathbf{t} \cdot \mathbf{a}_v) = t^y$, and $(\mathbf{t} \cdot \mathbf{v}) = t^z$.

Using this notation we reduce the components of our expression to $r_1 = (\mathbf{r} \cdot \mathbf{a}_\perp) =$

$2n^z n^x$ and $r_2 = (\mathbf{r} \cdot \mathbf{v}_a) = -2n^z n^y \sin\alpha + (n^{z2} - n^{x2} - n^{y2}) \cos\alpha$, where α is the angle between $\mathbf{v}$ and $\mathbf{v}_a$.

If we regard the surface as locally parameterized over the image plane, i.e. given by $z = f(x, y)$, we take the non-unit length normal to be $\mathbf{n} = (f_x, f_y, 1)$, where f_x and f_y are derivatives of f in two directions, and the expression for $\theta = \arctan(r_1, r_2)$ further simplifies to

$$\theta(x, y) = \arctan\left(2f_x, -2f_y \sin\alpha + (1 - f_x^2 - f_y^2) \cos\alpha\right). \tag{3.2}$$

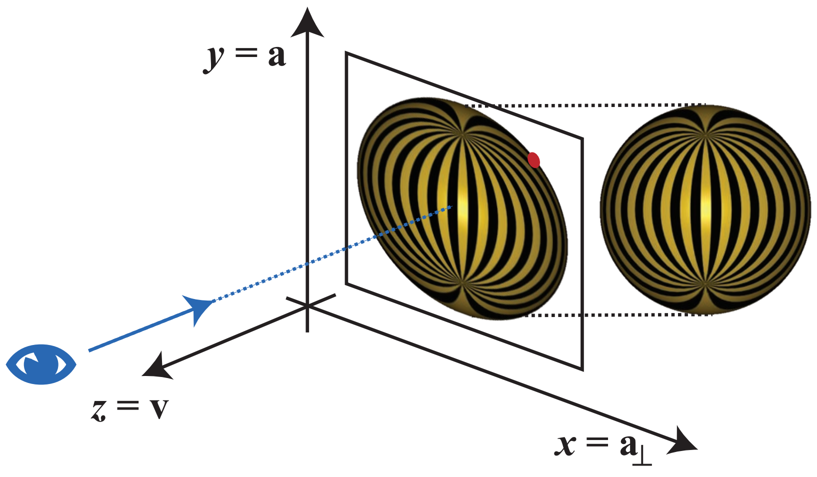

Figure 3.4: Image-plane parameterization. Red point depicts a silhouette point.

3.3.3 Optimization Problems

Given a user-defined reflection function our goal is to determine a surface which approximates this field as closely as possible. Mathematically, we can formulate the problem in several ways: exact match of the reflection function, minimization of the difference between the desired and actual reflection function, and minimization of the difference in line directions and density captured by the gradient of the reflection function ([115]). We briefly review these options here, with emphasis on the allowable boundary conditions.

One can observe that if $\theta(x, y)$ is given, equation (3.2) is a first-order PDE which can be solved using the characteristic ODE system $\dot{x} = F_{p^x},\ \dot{y} = F_{p^y},\ \dot{p}^x = -F_x,\ \dot{p}^y = -F_y$, where $F(x, y, p^x, p^y) = (1 - f_x^2 - f_y^2)\tan\theta^*(x, y) - 2f_x$. With suitable assumptions on the left-hand side and boundary conditions, the solution exists. However, as the system is first-order, only initial value problems generally have solutions. In particular one cannot expect the problem to have solutions if the values are prescribed on the boundary of a patch. We also note that if instead of specifying θ we had specified the reflection vector $\mathbf{r}$, the resulting system of two PDEs would not necessarily have a common solution even with no boundary conditions.

If the boundary of a region is fixed, the best we can do is to minimize the difference in reflection functions. Instead of fitting the angle values $\theta(x, y)$, we avoid the problems with the discontinuity of θ values by using the functional

$$\int_S (\cos\theta - \cos\theta^*)^2 + (\sin\theta - \sin\theta^*)^2 dxdy, \tag{3.3}$$

where the integral is over the image plane projection of the region of interest, with projection assumed to be one-to-one.

The Euler-Lagrange equation for this problem is second-order; therefore one hopes to be able to solve the problem with Dirichlet data on the boundary, but not with Neumann data. This implies that one cannot expect the solution to blend smoothly with the rest of the surface if the optimization is performed only on a small area (Figure 3.5, left).

Finally, instead of fitting the function values one can fit the gradient of the reflection function to the gradient of the desired function:

$$\text{Minimize} \int_S (\nabla\theta - \nabla\theta^*)^2 dxdy,\ \theta|_{\partial S} = \theta_0,\ \frac{\partial}{\partial n}\theta|_{\partial S} = \varphi_0, \tag{3.4}$$

where $\partial/\partial n$ is the derivative along the boundary normal. In this case, the corresponding Euler-Lagrange equation is fourth-order, similar to the PDE for the thin-plate energy, and one can prescribe both Dirichlet and Neumann boundary conditions ensuring smooth transition between the optimized patch and the surface (Figure 3.5, right). In this energy, we need an expression for $\nabla\theta$. As $r_1 = 2f_x$ and $r_2 = -2f_y \sin\alpha + (1 - f_x^2 - f_y^2)\cos\alpha$,

$$\nabla\theta = \frac{r_2 \nabla r_1 - r_1 \nabla r_2}{r_1^2 + r_2^2}$$

All problems that we have considered assume that a reflection line function $\theta(x, y)$ is prescribed.

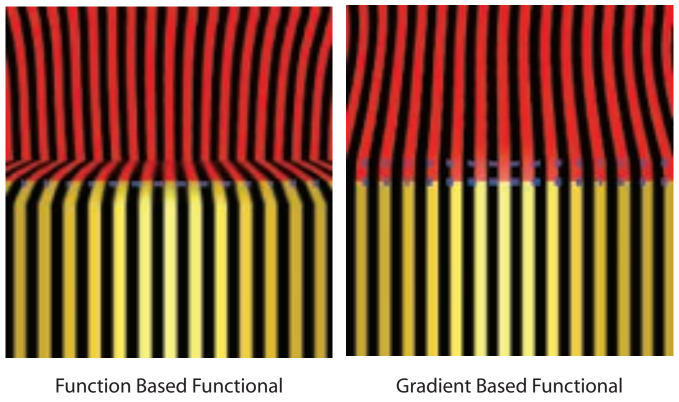

Figure 3.5: Dirichlet boundary conditions in function based functional versus Neumann boundary conditions in gradient based functional. Blue points show the fixed boundary vertices.

3.4 Discretization and Numerical Methods

We aim to design a discretization of problem (3.4) which balances accuracy, robustness and efficiency required by interactive applications.

A C^1 finite-element or arbitrary mesh C^1 spline discretization would be most straightforward but is relatively expensive. We use a more efficient and easier to implement alternative which combines a discrete-geometric technique with finite differences. We use triangle-centered discretization stencils for both first and second-order derivatives which leads to simple discretization of Equation 3.4. While there is no rigorous convergence guarantee by construction, we show excellent behavior for most mesh types.

3.4.1 Reduction to Parametric Case

As it was discussed in the previous section, our optimization problems can be solved in a functional setting by using surface parameterization over the image plane. However, while obtaining such a parameterization is computationally expensive for high-order surfaces (e.g. subdivision surfaces and splines); for meshes, the parameterization is easily obtained by a simple linear transformation: we rotate the coordinate system so that the image plane coincides with the (x, y) plane and the projection of the light direction $\mathbf{a}$ to the image plane is aligned with the y axis, i.e. use the coordinate system $(\mathbf{v}, \mathbf{a}_v, \mathbf{a}_\perp)$ (Figure 3.3). As one cannot reliably control reflections near silhouette points, we fix all vertices close to silhouettes, i.e. fix vertices of all triangles with normals $\mathbf{n}$, for which $|\mathbf{n} \cdot \mathbf{v}| < \epsilon$ (we use $\epsilon = 0.02$ in all cases).

For surfaces with no silhouettes (e.g. nearly flat patches) additional boundary conditions are necessary: typically, we want the modified surface patch to join smoothly with the rest of the surface. After preprocessing, the mesh is decomposed into disjoint pieces, each of which is a piecewise linear height field over the image plane.

Our functionals depend on the components of the gradient and Hessian of f, which we discretize next. We use *triangle-centered* discretization, i.e. a single value of the gradient or Hessian is assigned to each face rather than vertex. This leads to simple formulas for the gradients and Hessians, and makes it possible to consider a minimal number of special cases. Each discretization associates a 2×6 and 3×6 matrix of coefficients G and H with each triangle. if $f_T = (f(\mathbf{p}_1), f(\mathbf{p}_2), f(\mathbf{p}_3), f(\mathbf{q}_1), f(\mathbf{q}_2), f(\mathbf{q}_3))$ (Figure 3.6) then Gf_T and Hf_T yield the gradient and Hessian respectively.

3.4.2 Discretizing Gradients

To discretize the gradient $\nabla f = (f_x, f_y)$ over the image plane, we use standard piecewise-linear continuous finite elements.

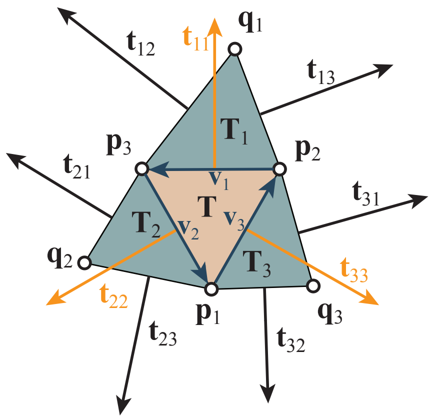

Figure 3.6: Vectors used in the gradient and Hessian definitions; all points are in the image plane. The vectors $\mathbf{t}_{ij}$ are perpendicular to corresponding triangle sides and have the same length as these sides.

We observe that the gradient can be found in the form $\sum_i c_i \mathbf{t}_{ii}$ where c_i are determined by $\sum_i c_i(\mathbf{t}_{ii} \cdot \mathbf{v}_j) = f_j - f_k$, and (i, j, k) is a cyclic permutation of $(1, 2, 3)$. This yields

$$\nabla_{discr} f_T = \frac{1}{2A} \sum_{i=1,2,3} f(\mathbf{p}_i)\mathbf{t}_{ii} \qquad (3.5)$$

where $f(\mathbf{p}_i)$ denotes the value of f at vertex $\mathbf{p}_i$. The coefficients $\mathbf{t}_{ii}/2A$ do not change and need to be computed only when the surface is rotated.

3.4.3 Discretizing Hessians

Discretizing Hessians is considerably more difficult: while for gradients a piecewise linear approximation depending only on function values at triangle vertices is adequate, for second derivatives one needs to use more vertices, or introduce additional degrees of freedom. As the total number of derivatives of order ≤ 2 is six, one needs at least six degrees of freedom per stencil to capture local behavior correctly.

Most discretizations of second-order quantities (typically, curvature) used in geometric modeling are *vertex-centered*, which is inconvenient for our purposes. To be compatible with the gradient discretization, we use a triangle-centered stencil shown in Figure 3.6. Another possible option is to use a single triangle and add edge-based degrees of freedom as it was done in [66]. We have experimented with a linearized version of this discretization. In contrast to the general curvature discretization (3 coordinates per vertex), addition of edge degrees of freedom in our setting (one degree of freedom per vertex) adds a significant computational cost. Furthermore, stability of the nonlinear solve is decreased which further decreases performance.

For triangles without vertices of valence three, it has six degrees of freedom, exactly the number needed for discretizing the Hessian.

On this stencil, an approximation to the Hessian can be constructed in a number of different ways. We use a combination of two approximations.

Triangle-averaged Discretization

Cohen-Steiner and Morvan [38] describe a general technique for computing shape operators on meshes by averaging elementary shape operators corresponding to edges. While convergence of this technique was only established in the integral sense, and for a restricted class of meshes, simplified versions of this technique were shown to work well in practice. The most common example is the well-known cotangent formula [141], which (for small deformations) is equivalent to expressions of [38], summed over a single ring of edges around a vertex [87]. Similarly, the triangle-averaged discretization on the stencil of Figure 3.6 introduced in [54] uses averaging over three edges of a triangle. By linearizing this formula, we obtain the following expression for the Hessian:

$$\frac{1}{A}\left(\sum_{i,j,j\neq i}\frac{1}{A_j}f(\mathbf{q}_j)\mathbf{t}_{ii}\otimes\mathbf{t}_{ij}+\sum_{i}\frac{1}{A_i}f(\mathbf{p}_i)\mathbf{t}_{ii}\otimes\mathbf{t}_{ii}\right) \tag{3.6}$$

where $\mathbf{t}_{ij}$ are side perpendiculars to the triangles of the mesh projected to the image plane, shown in Figure 3.6. A distinctive feature of this discretization is its robustness and simplicity: only for triangles with very small area may the coefficients in the formula be large.

This is a consistent (converging to the correct values) discretization of the Hessian for special types of meshes. Specifically, the Hessian is consistent for meshes in which vertices $\mathbf{q}_i$ are reflections of $\mathbf{p}_i$ with respect to the centers of opposite edges $i = 1, 2, 3$. This includes regular meshes and any affine transformations of regular meshes.

For general meshes, the discretization introduces mesh-dependent error in Hessian approximation, as shown in Figure 3.8. For different types of meshes, the results are mesh-dependent, no matter how fine the mesh is. Importantly for our application, the errors are low-frequency while high-frequency errors have the most effect on the visual quality of results.

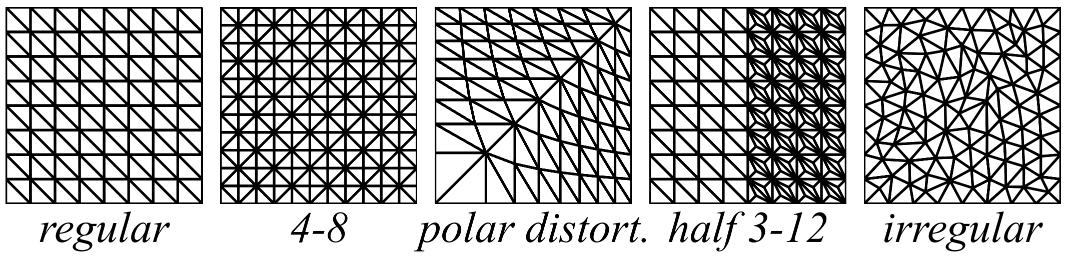

Figure 3.7: Mesh types used in convergence experiments.

Quadratic Interpolation Discretization

An alternative is to use a finite-differences: we compute a quadratic function Q satisfying $Q(\mathbf{p}_i) = f(\mathbf{p}_i)$, $Q(\mathbf{q}_i) = f(\mathbf{q}_i)$, $i = 1 \ldots 3$, and use its quadratic term coefficients to estimate the Hessian. The advantage of this method is that by construction it is consistent whenever the quadratic function is defined. This is not sufficient for convergence of the discrete problem solutions to the continuous solution, (see [66]) but improves independence of the result from mesh connectivity. Unfortunately, this technique is significantly less robust and the following proposition holds:

For six points $\mathbf{w}_i \in \mathbf{R}^2$, $i = 1 \ldots 6$, there is a unique quadratic function satisfying $Q(\mathbf{w}_i) = z_i$, for arbitrary choice of z_i, if and only if these six points are not on the same conic.

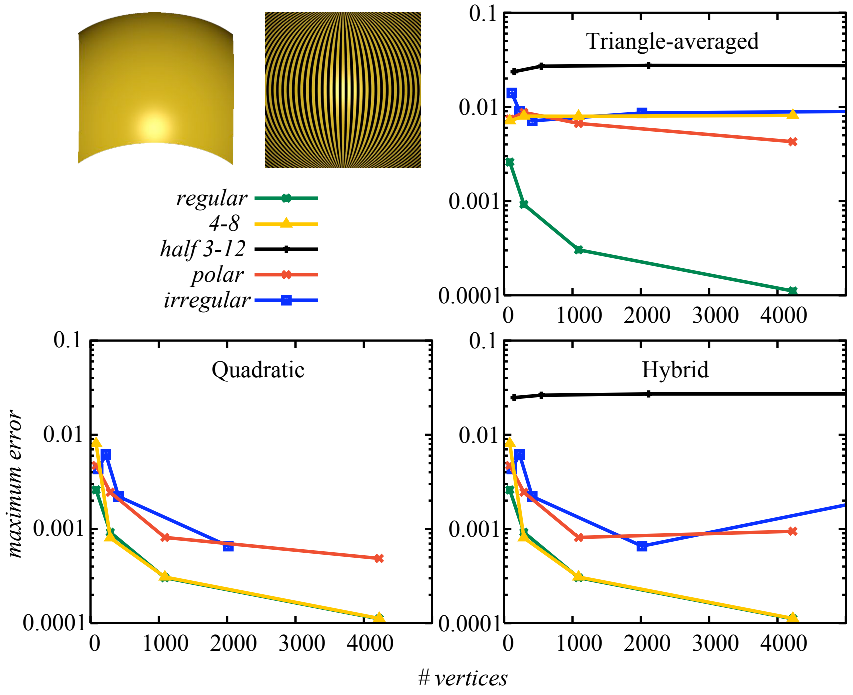

Figure 3.8: Convergence experiments: A spherical surface patch was recovered from an analytically computed reflection function gradient for different mesh connectivities and resolutions. Three discretization types are shown: triangle-averaged, quadratic fit and hybrid. Quadratic interpolation cannot be applied to meshes with vertices of valence 3; optimization also fails on higher resolution irregular meshes because it contains stencils with all vertices close to a conic. The error is measured relative to the size of the object along the view direction.

Whenever six points of the stencil are *close* to a common conic, the coefficients of the quadratic interpolant become large and Hessian estimation becomes highly unreliable (See Figure 3.9).

Quadratic Interpolation on a Six-point Stencil. The quadratic interpolation problem for a six-point stencil requires solving a linear system, which may be singular. A simple geometric condition for testing singularities of point configurations can be derived from geometric considerations, along with an explicit expression for

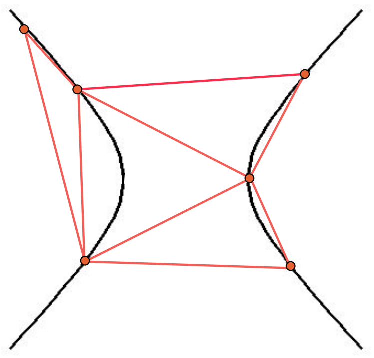

Figure 3.9: Example layout with six points lying on a conic.

the quadratic interpolant, which simplifies implementation.

We denote the six points of the stencil in the image plane by $\mathbf{w}_i$, $i = 0 \dots 5$. We need to find a quadratic function Q, such that $Q(\mathbf{w}_i) = z_i$, $i = 0 \dots 5$. Assume that all points are given in homogeneous coordinates. Then we can define Q as a 3×3 matrix: $Q(\mathbf{w}_i) = \mathbf{w}_i^T Q \mathbf{w}_i$. We observe that by linearity of the problem it is sufficient to solve it for $z_i = \delta_{ij}$, for $j = 0 \dots 5$, to obtain six basis matrices Q_j. Assume $j = 0$, i.e. $z_i = 0$ for $i \geq 1$. It well known that any five points in the plane are on a conic. Note that the problem of finding Q_0 is equivalent to the problem of finding such conic: Q_0 vanishes at all points $\mathbf{w}_i$, $i \geq 1$, i.e. defines a conic passing through these points, and conversely given a nontrivial conic with matrix M passing through $\mathbf{w}_i$, $i \geq 1$, we obtain Q_0 as $M/(\mathbf{w}_0^T M \mathbf{w}_0)$. If $\mathbf{w}_0^T M \mathbf{w}_0 = 0$, either there are multiple conics passing through the points, or the system has no solution; in either case, the system for Q_i is singular.

The conic matrix M can be computed using a matrix form of the Braikenridge-Maclaurin construction, [39]. Specifically, define $\mathbf{l}_{i,i+1} = \mathbf{w}_i \times \mathbf{w}_{i+1}$ (3d cross product applied to the homogeneous point representation). Let $R(\mathbf{a})$ be the skew symmetric matrix satisfying $R\mathbf{x} = \mathbf{a} \times \mathbf{x}$. Then M_0 is given by

$$M_0 = R(\mathbf{w}_1)R(\mathbf{l}_{34})R(\mathbf{r})R(\mathbf{l}_{23})R(\mathbf{w}_5),$$

where $\mathbf{r} = \mathbf{l}_{12} \times \mathbf{l}_{45}$ and Q_i are obtained by cyclically permuting $\mathbf{w}_i$ and applying the same formula.

Hybrid Discretization

As it can be seen in Figure 3.8, quadratic interpolation yields good estimates in most cases, but it is not robust. In practice, we observe that we have several triangles per mesh for which the six-point stencil is close to a conic, and quadratic interpolation produces low-quality results. To solve this problem, we combine two techniques: the triangle-averaged scheme is used when the quadratic interpolation is unstable, i.e. if the stencil contains five points or less, or vertices are close to a conic. We evaluate stability for a specific six-point stencil by comparing the magnitude of the discrete Hessian coefficients to $1/l_{max}^2$, where l_{max} is the maximal edge length in the stencil. If any coefficient exceeds C/l_{max}^2 (we use $C = 5$), we use triangle-averaged discretization instead of the quadratic interpolation. As it can be seen from the convergence plots, the resulting scheme retains the accuracy of the quadratic fit and yet does not suffer from its robustness problems, although it produces large errors for meshes with many degenerate cases. Such meshes appear to be unusual. Our stability criterion is motivated by the observation that if the function value is 0 at all but one vertex of a stencil, and is of the order l_{max}^2 (i.e. squared distance to other vertices) at that one vertex, one can expect to get second derivative magnitudes on the order of 1 (for a mesh close to regular). Coefficients much larger than $1/l_{max}^2$ lead to instability.

While numerically the discretization is more accurate, we note that we have observed few differences in visual quality when using the triangle-averaged discretization alone (Figure 3.10). Finally, while we found this discretization adequate for the functionals considered in this work, its performance for thin-plate or Willmore energy is not as good ([149]).

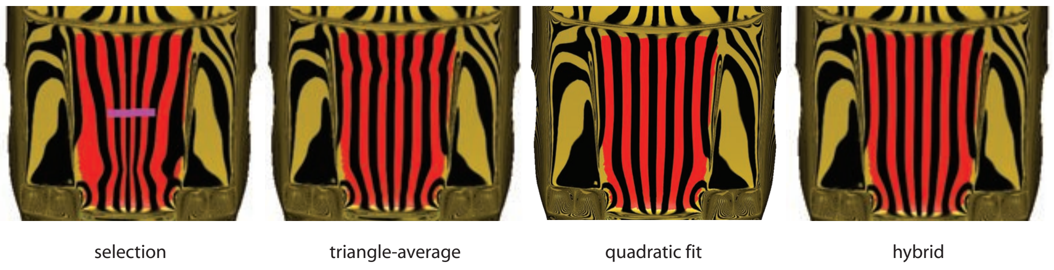

Figure 3.10: Visual comparison of triangle averaged, quadratic fit and hybrid discretizations. Leftmost image shows the initial view with the prescribed gradient direction.

3.4.4 Discretizing Normals

An essential part of an interactive system supporting reflection manipulation is rendering of reflection lines and environment maps. Hardware environment maps use vertex normals to compute reflected directions and look up values in the environment texture. As Figure 3.12 shows, standard vertex normal computation techniques produce low-quality results for complex meshes. Instead, we use a local fit to obtain better normals. For every vertex $\mathbf{p}$, we collect a ring N_1 of triangles around it, and all triangles edge-adjacent to N_1. The minimal number of vertices in such a configuration is six, unless the whole mesh has a smaller number of vertices. We compute an initial normal $\mathbf{n}_{init}$, and project vertices to the plane perpendicular to $\mathbf{n}_{init}$.

Let w be the vector of projected point positions of length K, and let $f(w)$ be the vector of function values at these points. For each vertex, we precompute a

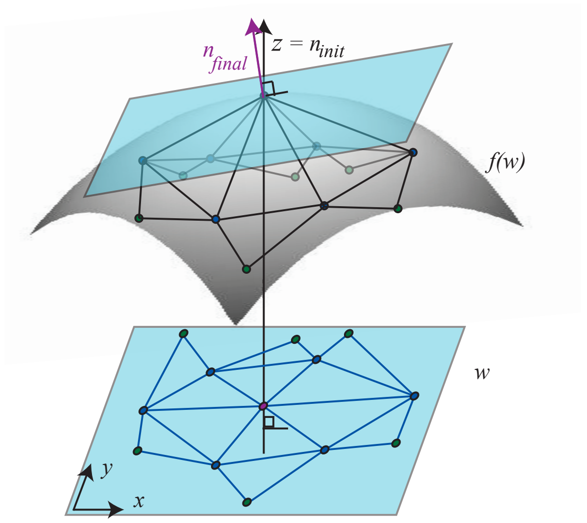

Figure 3.11: Normal estimation procedure based on quadratic fit.

$2 \times K$ matrix of coefficients C_{norm} mapping the vector $f(w)$ to the *linear* coefficients of a best fit quadratic function in the coordinate system with origin at vertex $\mathbf{p}$, with z axis aligned with the initial normal (when it is not defined uniquely, we choose the quadratic function with minimal norm of coefficients; the coefficients are computed using LAPACK function GESDD). These coefficients define a plane passing through the vertex, and we use the normal to this plane as our final normal. As long as we do not change the plane we have used for local normal estimation, the normal approximation can be recomputed rapidly as the surface is modified. If the motion is large, the quality deteriorates, and $\mathbf{n}_{init}$ needs to be recomputed.

3.4.5 Numerical Implementation

The reflection-based functionals are more complex than most expressions commonly used for surface optimization, therefore computing Hessians and gradients are also more expensive. The energy cannot be replaced by a linearized functional,

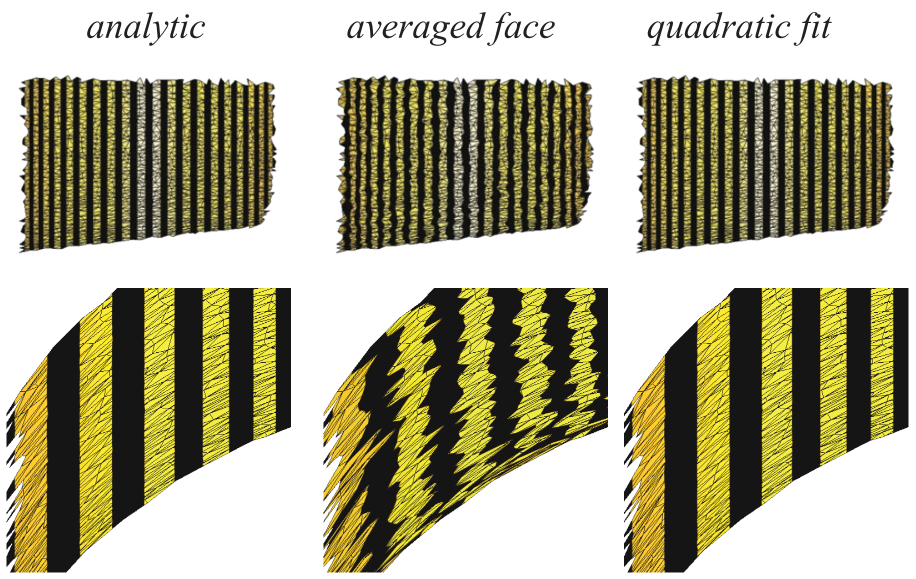

Figure 3.12: Comparison of the vertex normal quality when the surface is obtained by sampling points from a cylinder. While face averages do not perform well for this mesh, our quadratic fit procedure yields results visually indistinguishable from using analytic normals.

because then it would not capture the shape of reflection lines in most cases. Either a full non-linear Newton solve for the energy minimum or a gradient-only method would be prohibitively expensive, the former due to Hessian computation, and the latter because of the large number of iterations required.

To improve performance we use an inexact Newton method with line search. Instead of a full Hessian computation, we compute the Hessian once for the *linearized problem*, and use it instead of the full Hessian optimization at all iterations.

If we assume small values of f_x and f_y a simple calculation shows that the equation for the reflection function reduces to

$$\theta_{lin}(x, y) = 2f_x \cos\alpha.$$

The gradient of θ_{lin} remains simple, and is, up to a constant, $[f_{xx}, f_{xy}]$. For the quadratic energy based on θ_{lin}, the Hessian is easy to compute and does not depend on function values.

As the gradient problem is fourth-order and the condition number of the Hessian matrices grows as N^4, iterative solvers are not efficient; instead, we use a direct solver (PARDISO). The direct solver performs a sparse LU factorization of the matrix, and solves the system using back substitution. As the Hessian matrix does not change, the matrix factorization needs to be performed only once. For each nonlinear iteration, only gradients need to be recomputed. As shown in Figure 3.13, while using an approximate Hessian increases the number of nonlinear iterations required, each iteration becomes much faster, and there is a considerable net win in performance. This performance improvement depends on the complexity of the target reflection function gradient $\nabla\theta^*$ and varies in the range $2\times$ to $10\times$. When the target function is smooth, which is the most common case (forward optimization in Figure 3.13), the speed-ups are higher, while for less smooth targets speed-ups are lower.

3.5 Reflection Line Manipulation Experiments

Different types of reflection manipulation using discrete reflection line functionals follow the same general pattern. First, the user selects an area to modify and specifies the boundary conditions. Any boundary segment may be free, fixed, or clamped, the latter means that two layers of vertices are fixed at the boundary to ensure a smooth match of the surface with the rest of the mesh. Then the target reflection function gradient is defined using user input and the surface minimizing the reflection functional is computed. The last two steps may be repeated in the interaction loop.

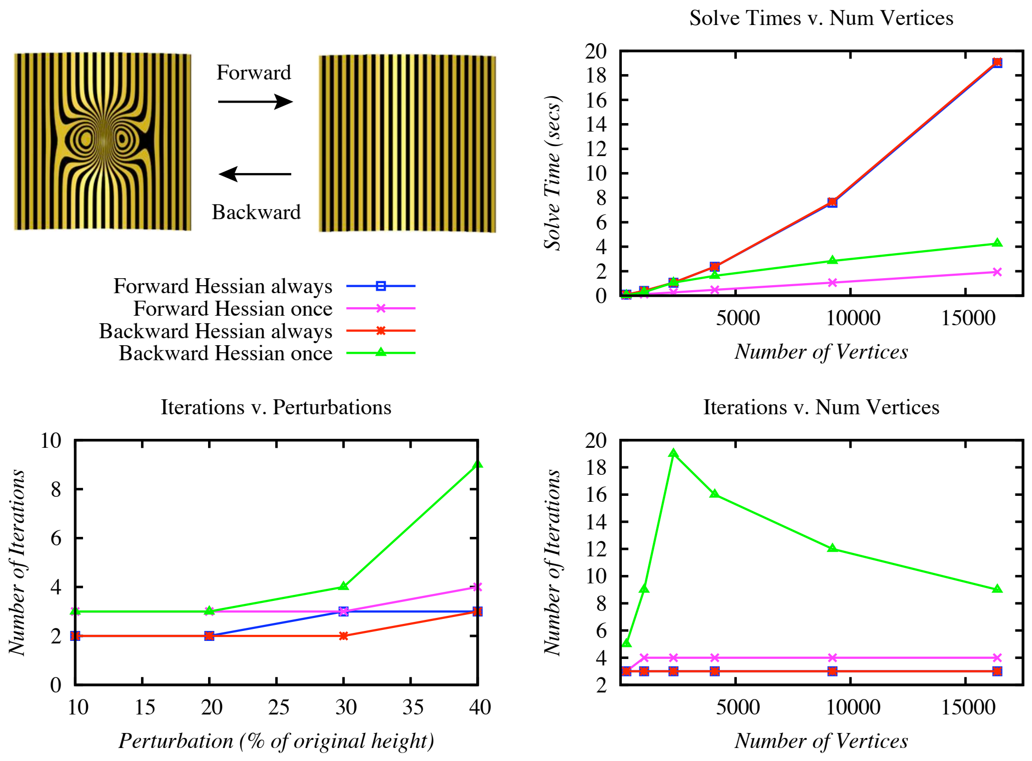

Figure 3.13: Speed-up from approximate Hessian computation, dependence on the mesh size. Two model problems: creation and elimination of a bump on a cylinder. Times are given for a Pentium D 3GHz processor.

Density and direction change. The simplest type of reflection line manipulation is requesting a fixed line direction and density in an area, i.e. specifying a fixed target $\nabla_{discr}\theta^*$ everywhere. The energy minimization in this case attempts to modify reflection lines as requested, while maintaining a smooth or continuous join of the selected patch with the rest of the surface. The user can also adjust a fall-off curve $c(t) \geq 1$ which determines how gradual the transition between the modified area and the rest of the mesh should be. The energy of each triangle is scaled by $c(d)$ where d is the distance to the center of influence.

Examples of this type of manipulation are shown in Figure 3.15, Figure 3.16 and Figure 3.18. In Figure 3.15, one can see how adjusting reflection line density makes it possible to control the appearance of reflections, in particular the sizes of reflected objects. Rotating the desired reflection line direction at a point is shown in Figure 3.16. In this example, a surface imperfection creates a "twist" in the reflection line field which can be removed by local rotation. Figure 3.18, demonstrates the difference between conventional smoothing and reflection line optimization: typically, smoothing algorithms flatten the surface which does not necessarily improve the reflection line shape.

Smoothing. Isotropic or directional smoothing can be used to transform initial $\nabla\theta$ values to the target values. This operation is similar to reflection line smoothing described in [115]. Directional smoothing is particularly useful, as it straightens reflection lines without changing the behavior in the orthogonal direction. An example is shown in Figure 3.19. In this example, the initial reflection function was smoothed using Laplacian smoothing in the image domain, and then used as the target reflection function.

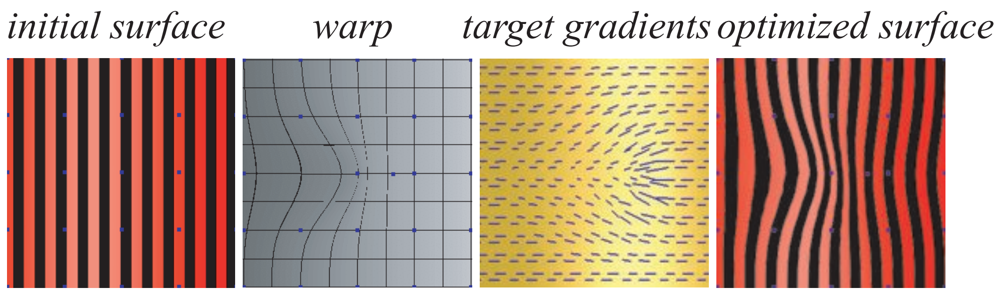

Figure 3.14: Warping stages: initial reflection lines, warped reflection function θ, target $\nabla\theta^*$ computed per triangle, reflection lines on the optimized surface.

Warping. In the case of warping, the goal is to apply an arbitrary user-specified transformation to the reflection lines. In our current implementation, the transformation of reflection lines is specified by a two-dimensional spline, however any other image warping technique can be used. The general formula for the transformed reflection function is

$$\theta_{warp}(s,t) = \theta_{init}(w^{-1}(s,t))$$

where $w(s,t) : \mathbf{R}^2 \to \mathbf{R}^2$ is the warping function.

Observe that the inverse of the warping function needs to be computed, which can be relatively expensive for smooth deformations. To avoid explicit inversion of $w(s,t)$, we implement the warp using texture mapping and image-domain operations. Rather than attempting to transform $\nabla_{discr}\theta$ values needed by the functional directly, we transform reflection function values, and then compute the gradient. First, the reflection function θ values are computed, interpolated to vertices, and used as color values to render the mesh to a texture θ_{init}. A two-dimensional spline $w(s,t) : \mathbf{R}^2 \to \mathbf{R}^2$ is created. Initially, the control points are equispaced so w is an identity. As the user moves control points, the spline is rendered to a new texture θ_{warp}, using θ_{init} as a texture map. For image-space mesh vertex positions we sample $\Delta\theta = \theta_{warp} - \theta_{init}$, and compute its gradient for each triangle using the gradient formula 3.5, with appropriate corrections applied to values to eliminate the jump between $-\pi$ and π. We add resulting $\nabla_{discr}\Delta\theta$ values to the initial $\nabla_{discr}\theta$ values. Note that by construction $\Delta\theta = 0$ if w is the identity function. An example of warping is shown in Figure 3.17; where the goal is to improve the shape of reflections at a part of a car hood.

Finally, one can use our technique to create surfaces approximating an arbitrary reflection line pattern, as shown in Figure 3.20. Any grayscale image can be used to specify $\nabla\theta^*$, as long as it is sufficiently smooth (otherwise, the approximation is likely to be poor).

3.6 Conclusions

We have described a simple and efficient technique for discretizing reflection line based functionals on meshes and demonstrated how these functionals can be used in an interactive system to optimize the shape of reflective surfaces.

One limitation of the proposed approach (which is also responsible for its comparatively high efficiency) is that the vertices of the mesh move only in the direction perpendicular to the image plane. This means that small scale surface details which make the projection to the image plane not one-to-one cannot be removed, and creates a disturbance in the surface during optimization. Although it can be applied to large perturbations, the technique is best suited for smaller adjustments of surfaces that are already relatively smooth. While we tried to eliminate obvious inefficiencies in our implementation, our code is far from optimal.

Our discretization can be easily combined with other surface optimization techniques, to be applied simultaneously or as a post-process.

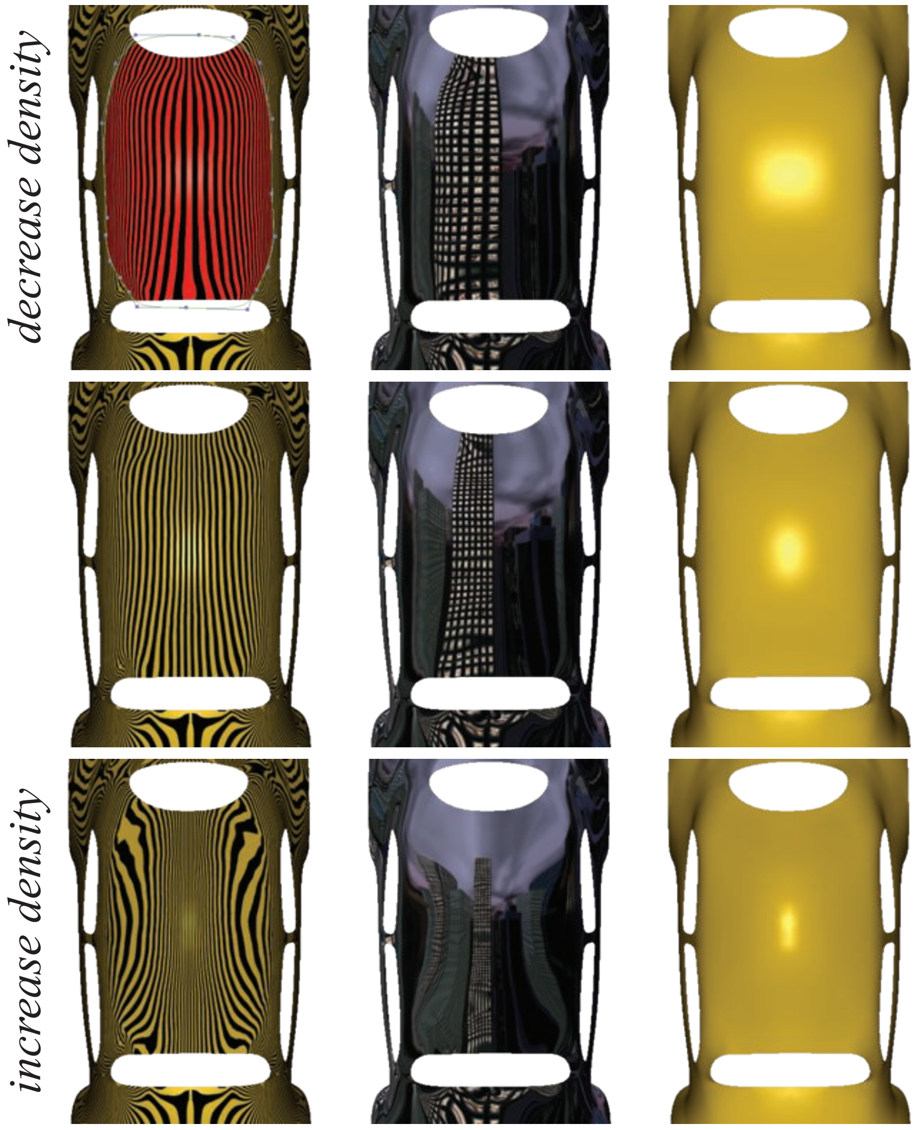

Figure 3.15: Changing reflection line density. A fixed reflection line direction is specified with density decreased at the top and increased at the bottom.

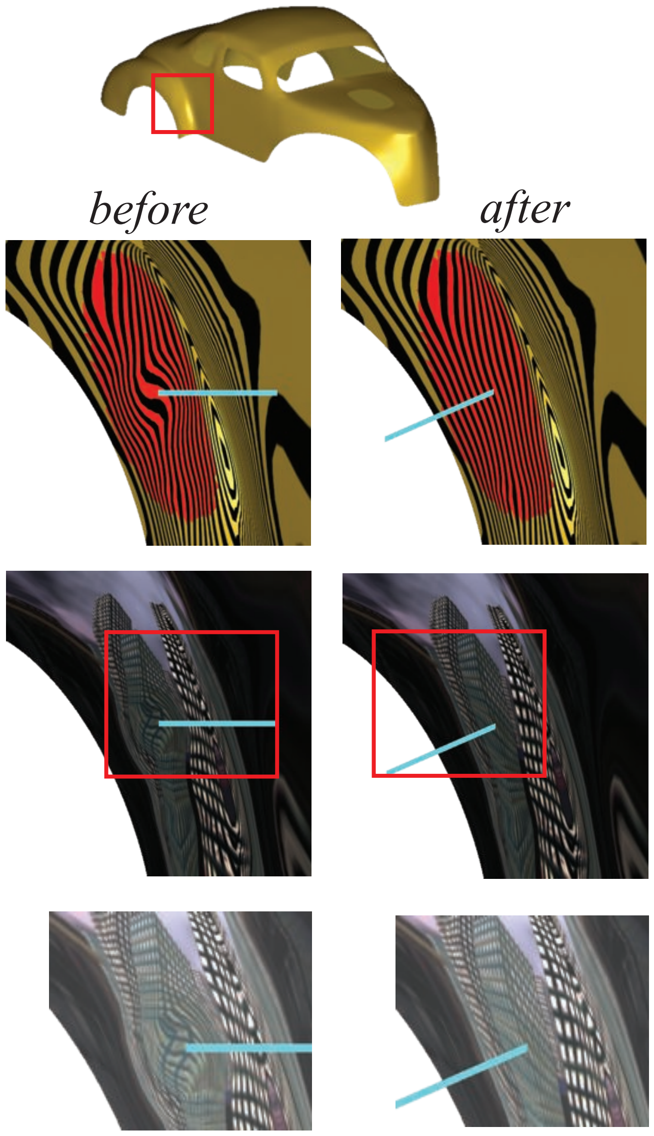

Figure 3.16: Reflection line untwisting. The reflection function gradient is rotated to get desired appearance.

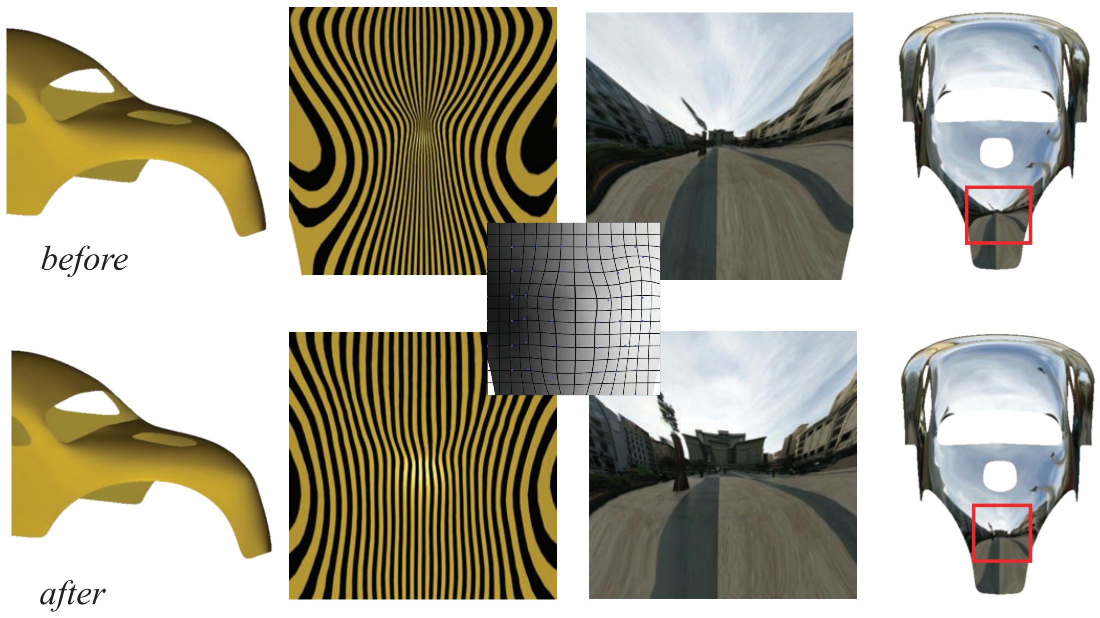

Figure 3.17: Reflection line warping on a car hood. An intermediate warp is shown in the middle. Note that the change in the shape is barely perceptible but the change in the reflection is substantial.

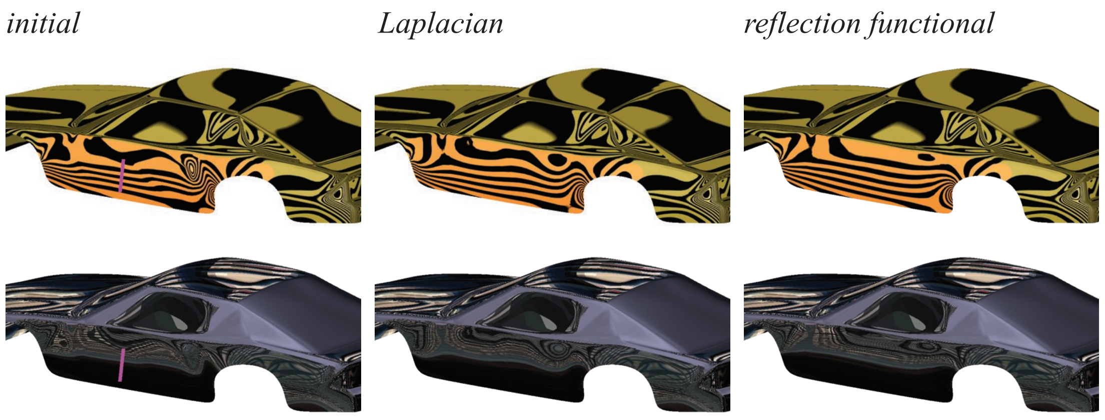

Figure 3.18: Prescribing fixed reflection line direction on a car, compared to Laplacian smoothing. Note that Laplacian smoothing retains reflection line wiggles.

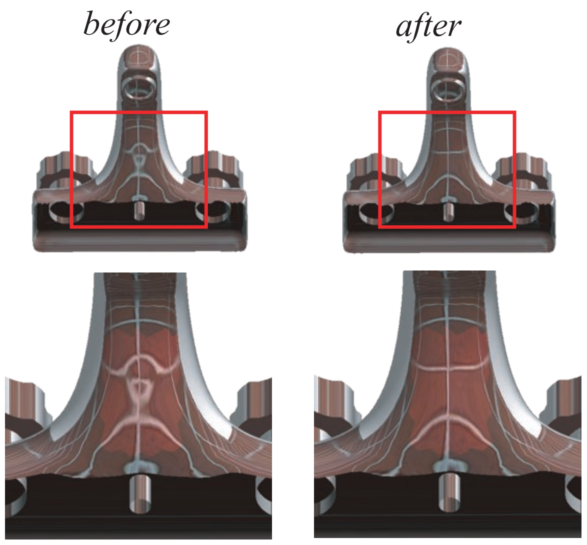

Figure 3.19: Reflection line smoothing on a faucet; the initial reflection line function is smoothed and used as the target reflection line function.

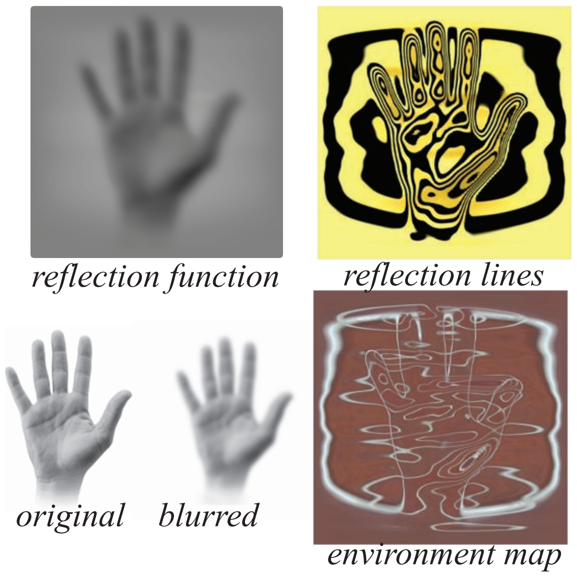

Figure 3.20: Reconstructing a surface with predefined reflection line pattern based on an blurred image.

4 Biharmonic and Triharmonic PDEs on Meshes

Finite element methods are commonly used to solve high-order PDEs as discussed in Section 1.3.3. For the solution of a high-order problem, such as the fourth-order biharmonic and the sixth-order triharmonic problems we discuss in this section, there are a few directions one can take within the finite element framework. One option is to use conforming finite elements. Even though this works well for low-order problems, the smoothness requirements on the elements for high-order problems makes this formulation rather complicated. While C^0 finite elements (the linear Lagrange element) suffices to solve a second-order problem, one needs C^1 elements for a fourth-order, and C^2 for a sixth-order problem. These elements have additional functional or differential degrees of freedom placed on edges or faces in order to satisfy inter-element continuity and therefore are complex to construct and use.

An alternative is based on splitting the high-order problem into a system of second-order problems by introducing extra variables which can be solved by simpler conforming elements such as the C^0 piecewise linear element. This eliminates the need to compute discretizations of high-order quantities, such as those described in Section 1.3.2. However, it leads to highly ill-conditioned systems, especially for problems of sixth- or higher-order, and requires extra effort to solve.

In this chapter, we present a mixed finite element formulation of biharmonic and triharmonic problems. We show that discretizations based on discrete Laplacians can be derived in the mixed element framework. We report on experimental results and discuss trade-offs of this construction.

4.1 Introduction

The polyharmonic equation, $\Delta^k x = f$, appears quite often in geometric modeling applications. The first-order Laplacian, i.e. $k = 1$ is the basis for most smoothing operators used extensively in the field. The second- and third-order Laplacians appear in the linearizations of the Euler-Lagrange equations for minimizing curvature and variation of curvature on a surface, respectively.

In the discrete setting, these equations are usually solved by applying a Laplacian discretization repeatedly on a mesh, with the cotangent formula being the most popular and the one used in [18]. Although this formulation is efficient and particularly useful in interactive applications, it supports only one particular way of specifying boundary conditions and the results often have significant mesh dependence.

In this chapter, we consider several closely related formulations for biharmonic and triharmonic PDEs. The results however, are different due to variations in boundary condition specification. Some applications, for example, hole-filling and blending, are most naturally formulated by specifying a piece of the surface outside the problem domain. This piece is typically approximated by a part of the mesh neighboring the domain. In other cases, such as curve network interpolation, the function and its normal derivatives are specified along a curve. Finite element methods require formulations with boundary conditions of the latter type. However, the conventional FEM discretization of a high-order PDE, more specifically the triharmonic equation, may lead to highly ill-conditioned and possibly singular systems (Section 4.6.2).

We introduce a novel finite element discretization, using function and derivative values outside the domain that does not lead to singular systems and exhibits consistent behavior of solutions under refinement for a variety of meshes. Furthermore, we present a regularization process to avoid the instability problem that arises with high-order finite element systems.

4.2 Previous Work

A number of FEM and discrete geometric techniques for solving the biharmonic problem has been proposed. One non-FEM based method is described in [18]. In this work the high-order Laplacians are defined recursively as in:

$$\bar{\Delta}^k(p) := \Delta(\lambda_{k-1}(p) \cdot \bar{\Delta}^{k-1}(p)) \tag{4.1}$$

where λ_k is smoothness parameter that controls smoothness of the blend at boundaries and p is a vertex on the mesh. The discrete Laplace operator used in this work is the cotangent formulation and was given in equation 1.25. Boundary conditions are prescribed by fixing $k+1$ rows of boundary vertices.

Mixed formulations of the biharmonic equation are extensively studied in finite element literature as covered in Section 1.3.3. However, most results are theoretical, limited to proving error bounds rather than providing a practical point of view.

The triharmonic problem has been studied in the parametric surface setting in [108]. Here the boundary conditions are prescribed interactively using a set of curves on or near the boundary of the surface. The vectors describing the difference between the boundary and the prescribed curves are used as the first and second Neumann boundary conditions. In this paper, the problem is solved analytically on periodic surfaces. A similar analytical solution is provided for a generalized version of the triharmonic equation in [205]. In this case the technique is applied to constructing multi-patch composite surfaces.

To our knowledge, the only other work in geometric modeling literature solving the triharmonic problem on a mesh is [18]. The authors formulate the problem as a discrete geometric PDE (equation 4.1), where the system is solved using discretized differential quantities (the discrete Laplacian in this case). Unlike [18], our method is derived using a mixed finite element discretization.On the other hand, we could derive the same system in [18] from mixed finite elements.

Even though we do not have a proof of convergence for our formulation, we have strong experimental evidence of convergent behavior not only for the unknown function, but also its first and second Laplacians which appear as auxiliary variables in our system. The importance of convergence comes from the fact that it removes mesh dependency of the discrete problem for high resolution meshes. It also guarantees that the limit surface is well defined independent of the choice of refined triangulations subject to constraints such as minimum angle, or aspect ratio preservation. This gives the user freedom to use any mesh, re-meshing algorithm or adaptive refinement algorithm without the concern for introducing visual artifacts. The importance of this property has been covered in [66, 207]. More specifically, convergence and consistency of the Laplace-Beltrami operator is studied in [186, 187] . It is concluded that the discretization schemes do not converge point-wise unless there exists a restriction on regularity of the mesh or the scheme is based on fitting. The convergent discretization studied in [186] is based on a quadratic fitting scheme, whereas ours is not.

4.3 Notation

In the rest of the chapter we use the following notation.

Let Ω be a convex domain in $\Re^2$ and $\partial\Omega$ its boundary. For simplicity we consider polygonal domains. $H^m(\Omega)$ is the Sobolev space defined on Ω, and is also the completion of $C^\infty(\Omega) \cap H^m(\Omega)$, where C^∞ is the space of infinitely differentiable functions. $H_0^m(\Omega)$ is the completion of $C_0^\infty(\Omega)$, the subspace of infinitely differentiable functions which are nonzero only on a compact subset of Ω. This may be thought of as a generalization for functions satisfying zero boundary conditions [21].

$\mathcal{T}_h = \{T\}$ is a triangulation of Ω with h defined as the maximum edge length among all triangles $T \in \mathcal{T}_h$, as in $h = \max_{T\in\mathcal{T}_h} h_T$. Each triangle T is described by a triple of indices $< i, j, k >$.

We define approximating spaces in the following way:

$$X_h = \{v \in C^0(\Omega) : v|_T \in P_1, \forall T \in \mathcal{T}_h\},$$

where P_1 is the space of all linear polynomials and $C^0(\Omega)$ is the space of all continuous functions defined on Ω. The specific basis functions φ^i defined over the domain are hat functions; i.e. $\varphi^i(v_j) = \delta_{ij}$ where v_j is the vertex with index j.

Also, below we use $\langle g, h\rangle_X$ to refer to $\int ghdA$ on 2D domain X, or $\oint ghds$ for 1D curve X.

We assume the triangulated domain Ω_0 contains Ω, and that $\partial\Omega$ consists of triangle edges. Let N_{all} be the set of vertices of the triangulation. N_Ω is the set of vertices contained in the interior of Ω, N_0 are in $\partial\Omega$, N_1 is the of vertices in $\Omega_0 \setminus \Omega$ adjacent to N_0, N_2 are vertices not in N_1, adjacent to N_1, and N_e is the rest of the vertices in $\Omega_0 \setminus \Omega$ (See Figure 4.1).

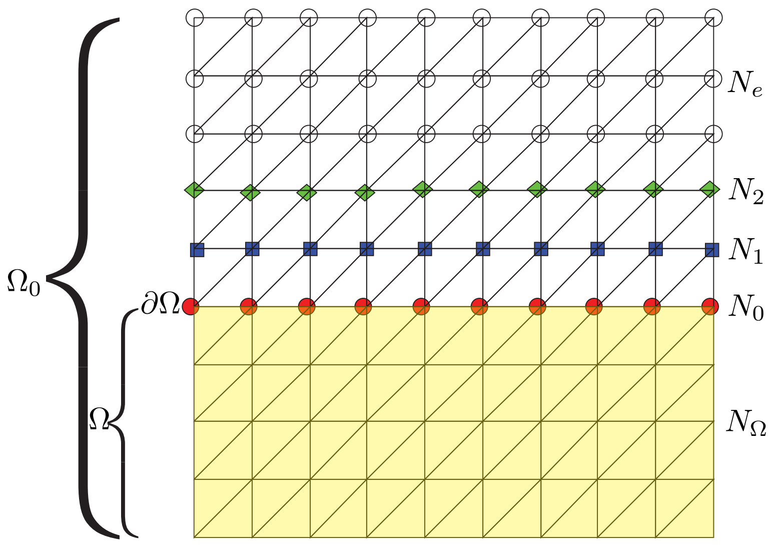

Figure 4.1: Notation for domain decomposition used in the rest of the chapter.

4.4 Boundary Conditions

As we have mentioned before, PDEs are used commonly in applications such as blending, hole-filing, and curve network interpolation.

Blending algorithms construct a smooth surface connecting two or more closed boundary curves of several given surfaces such that this new surface blends smoothly with existing ones. *Hole-filling* is similar to blending: if a part of a surface is missing (has a hole), the goal is to construct a new surface which joins smoothly with the boundary of the hole (See for example, [11, 35, 189, 190]). In these cases, the boundary conditions are typically specified by a piece of the surface, for example, a section of the existing surface near the prescribed boundary curves. In this case, the boundary data is gathered by sampling the function on the existing mesh vertices. We refer to this kind of boundary conditions as *function-based boundary conditions*.

In other applications, for example, in *curve network interpolation*, boundary conditions are specified in a more conventional form: boundary data for normal derivatives or other high-order differential quantities are prescribed on the boundary. Typically, the goal in curve network interpolation is to find a surface that smoothly interpolates a network of curves, possibly satisfying additional constraints along curves. The shape of the resulting surface is controlled by the boundary conditions prescribed on the curves and the resulting surface is tangent-plane or curvature continuous everywhere. Some research in this direction can be found in [29, 124, 131, 154]. We refer to this form of boundary conditions as *differential boundary conditions.*

4.4.1 Formalization of Boundary Conditions

In this chapter, we focus on finding a surface that is the solution of a fourth-order or a sixth-order polyharmonic equation, referred to as the biharmonic and the triharmonic respectively, specified by its boundary conditions. In this case, boundary conditions can be formulated in one of two ways.

The most convenient type of boundary conditions for solving a high-order system is to prescribe Laplacian values on the boundary.

$$x|_{\partial\Omega} = b_0 \tag{4.2}$$

$$\Delta^r x|_{\partial\Omega} = b_r, \forall r \in \{1, .., k-1\}, \tag{4.3}$$

Such a set of boundary conditions would allow the system to be broken down into a series of decoupled second-order equations each with Dirichlet conditions b_r:

$$\Delta\omega_{i-1} = \omega_i \text{ with } \omega_{i-1}|_{\partial\Omega} = b_r, \quad \forall i \in [1, ..., k-1]$$

where $\omega_0 = x$. However, this set of conditions is less desirable in practice since these boundary conditions of high-order Laplacians are not readily available and this formulation is less intuitive. For example, in the case of the biharmonic equation, the boundary conditions in this formulation are given by

$$x|_{\partial\Omega} = b_0 \tag{4.4}$$

$$\Delta x|_{\partial\Omega} = b_1 \tag{4.5}$$

In this case, while one can control second-order differentials (e.g. curvature), there is no control on first-order differentials (e.g. tangent behavior), which makes this formulation less practical.

Instead, the following set of boundary conditions is more typical in applications

$$x|_{\partial\Omega} = b_0 \tag{4.6}$$

$$\frac{\partial^r x}{\partial n^r}|_{\partial\Omega} = b_r, \forall r \in \{1, .., k-1\}, \tag{4.7}$$

where b_0 is the Dirichlet boundary condition and b_r are Neumann boundary conditions of order r and the problem is finding the function x such that $\Delta^k x = f$ subject to these boundary conditions.

An alternative formulation of boundary conditions is based on the assumption that the function x is known outside the domain. In this case, the boundary conditions are given as values of the unknown function x on $\Omega_0 \setminus \Omega$ where Ω_0 is a larger domain that contains the problem domain Ω, and x has a certain continuity. With the alternative form of boundary conditions, the polyharmonic problem can be stated as follows. Find x satisfying:

$$\Delta^k x = f, \text{ with } x \in C^{k-1}(\Omega_0)$$

$$x|_{\Omega_0 \setminus \Omega} = b \tag{4.8}$$

where b is a given function.

4.4.2 Discretization

Most methods used in geometry processing are designed to handle boundary conditions that are function-based [7,18,35]. In most cases a set of vertices on or near the boundary is fixed in order to prescribe these conditions. Even though such types of boundary conditions appear to work well, these are not commonly considered in FEM formulations. Differential type boundary conditions are standard for FEM and theoretical results in the field are all based on this type of boundary conditions.

While one can switch between these types of boundary conditions (Section 4.4.3), one can also use either type directly in a problem discretization. In discrete form, assuming x known on Ω_0 leads to fixing several (2 for biharmonic, 3 for triharmonic) rows of values on the triangulation of Ω_0 outside Ω. Assuming x and $\frac{\partial x}{\partial n}$ known on the boundary (with $\frac{\partial^2 x}{\partial n^2}$ added for triharmonic) coincides with standard FEM settings and does not require any values outside Ω.

As the triharmonic equation allows for more boundary conditions, a combination of boundary condition discretizations may be considered. For example, one can assume x specified on $\Omega_0 \setminus \Omega$ and, in addition one of ∇x or Δx on $\Omega_0 \setminus \Omega$ can be used directly in the problem formulation. We found it particularly useful to use values of Δx outside Ω. Somewhat surprisingly, the conventional FEM discretization of triharmonic equation leads to singular systems for many common meshes, including a mesh with regular connectivity. The discretization using values of x outside Ω (but not ∇x or Δx) leads to robust discretization but experimentally does not converge for a number of mesh types. This discretization is close to the one introduced in [18] using discrete Laplacians.

Our discretization uses x and Δx values outside Ω, and leads to non-singular systems and results that give empirical evidence of better convergence.

4.4.3 Conversion Between Boundary Conditions

The problem formulation for a high-order PDE includes differential type boundary conditions, with the number of such conditions increasing with increasing order. For example, there are two such conditions (Dirichlet and Neumann) for fourth-order problems, whereas three are required for sixth-order problems. In order to apply a method naturally formulated for one type of boundary conditions to a problem with another type of conditions, we need to be able to convert between different types. In this section we describe the technique to go from one type of boundary conditions to another.

Conversion from differential to function-based. Let us consider the differential-type boundary conditions for the triharmonic problem:

$$\begin{aligned} x|_{\partial\Omega} &= b_0 \\ \frac{\partial x}{\partial n}|_{\partial\Omega} &= b_1 \\ \frac{\partial^2 x}{\partial n^2}|_{\partial\Omega} &= b_2 \end{aligned} \tag{4.9}$$

There are two ways of discretizing function-based boundary conditions for the triharmonic equation as described in the previous section. One involves prescribing function values on $\Omega_0 \setminus \Omega$ which involve three fixed rows, whereas the other involves prescribing function values on two rows of $\Omega_0 \setminus \Omega$ as well as prescribing Δx on one row.

We first describe the conversion from differential boundary conditions to function based conditions involving two fixed rows of x and one row of Δx.

In this conversion, Dirichlet conditions transfer over as one would expect: One row of boundary vertices are fixed as the prescribed function b_0. Neumann conditions are more complicated to convert. We transform the differential Neumann boundary conditions to fixing another row of vertices one edge away from the boundary by casting it as a least squares fit.

We solve

$$\sum_{(ij)} (a_1^T x_i + a_2^T x_j + a_3^T x_k - b_1^T)^2 \to min$$

where (ij) are boundary edges, a^T are coefficients for computing the normal derivative of a linear function on the triangle $T =< i, j, k >$, and k is the vertex opposite the edge (ij) in the corresponding boundary triangle. b_1^T is the Neumann boundary condition prescribed to edge (i, j) in triangle T. The minimization is over x_k, and can be obtained by solving the normal equation.

Note that one can either compute and prescribe the vertex positions of the vertices one row inside the boundary, or can extrapolate to a set of vertices added in the normal direction outside of the given domain boundary. One possible advantage of the latter could be the control over mesh connectivity for this outside layer, since for an irregular mesh the vertices one row inside the boundary can be very irregularly placed (Figure 4.2).

To transform the high-order normal derivative condition, observe that, since Dirichlet boundary conditions are also enforced, one can always compute the tangential derivative on the boundary from this data. With this information, one can compute Dirichlet conditions for the Laplacian of x, denoted as Δx. Let b_3 be this new set of Dirichlet conditions for Δx.

$$\Delta x|_{\partial\Omega} = \frac{\partial^2 x}{\partial t^2} + \frac{\partial^2 x}{\partial n^2} = \frac{\partial^2 b_0}{\partial t^2} + b_2 = b_3. \tag{4.10}$$

As with x, Dirichlet conditions for Δx transfer over to the function-based for-

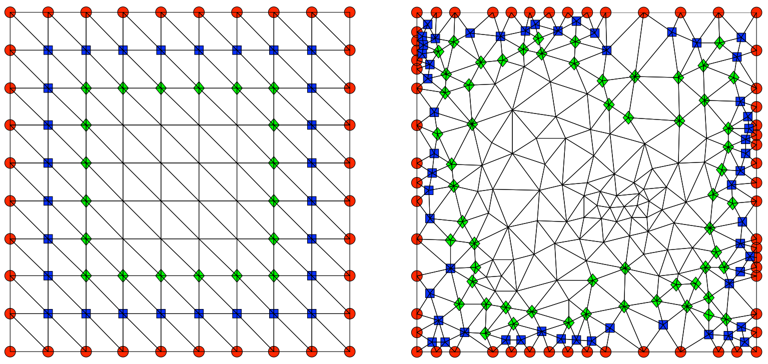

Figure 4.2: Difference in vertex layout for three layers of boundary vertices in the case of regular versus an irregular mesh.

mulation exactly.

Now we describe the conversion from differential boundary conditions to the second type of function-based conditions which is given by three fixed rows of x on $\Omega_0 \setminus \Omega$.

Given x, $\partial x/\partial n$ and $\partial^2 x/\partial n^2$ on $\partial\Omega$, we can estimate the values of x on two additional rows outside Ω, again using a least squares polynomial fit. In the simplest case, we construct a quadratic polynomial using six vertices show in Figure 4.3.

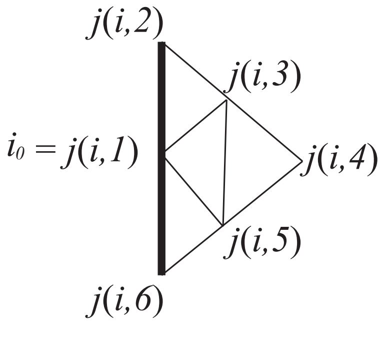

Figure 4.3: Six vertices used in quadratic polynomial fit, with dark edges showing the boundary.

This polynomial is of the form

$$\sum_{0\le l+m\le 2}(\sum_{r=1}^{6} p_{ir}^{lm} x_{j(i,r)})u^l v^m = P(x,u,v)$$

where (u,v) are coordinates in the plane, and $j(i,r)$ is used to denote the index of the vertex in position $r = 1...6$ with respect to the boundary vertex i_0.

As the dependence on x is linear, it can be written as

$$\sum_{r=1}^{6} P_{ir}(u,v)x_{j(i,r)}.$$

Denoting

$$(\frac{\partial}{\partial n}P_{ir})(u_0,v_0) = a_{ir}^{n}$$
$$(\frac{\partial^2}{\partial n^2}P_{ir})(u_0,v_0) = a_{ir}^{nn}$$

where (u_0,v_0) are the coordinates of vertex i_0, we obtain expressions for normal derivatives at i_0 given respectively by :

$$\sum_{r=1}^{6} a_{ir}^{n} x_{j(i,r)} \text{ and } \sum_{r=1}^{6} a_{ir}^{nn} x_{j(i,r)}.$$

The least squares problem we solve for values in two exterior rows is

$$\sum_{i\in N_0}(a_{ir}^{n} x_{j(i,r)} - b_1^i)^2 + (a_{ir}^{nn} x_{j(i,r)} - b_2^i)^2$$

where b_1^i and b_2^i are values of normal derivatives at vertex i as given in (4.9).

Conversion from function-based to differential. As we have explained, function-based conditions provide function values at vertices. Therefore, given function values for vertices in the neighborhood of a boundary vertex, the conversion consists of approximating derivative values at this boundary vertex. In the biharmonic case, this neighborhood involves two fixed rows in $\Omega_0 \setminus \Omega$ whereas in the triharmonic it is three fixed rows in $\Omega_0 \setminus \Omega$. One common technique for estimating differential quantities from function data is to use a fitting algorithm. For our purposes, we fit a polynomial of sufficiently high degree to the function values specified at fixed vertices in the least squares sense. Once the polynomial is obtained, boundary values of derivatives are evaluated. In order to compute *normal* derivatives, one also needs to compute the boundary normal, which is a simple cross product of triangle edges at the boundaries.

4.5 Biharmonic Equation

The biharmonic equation with the natural boundary conditions is given by

$$\Delta^2 x = f \tag{4.11}$$

$$x|_{\partial\Omega} = b_0 \tag{4.12}$$

$$\frac{\partial x}{\partial n}|_{\partial\Omega} = b_1 \tag{4.13}$$

4.5.1 FEM Formulation

The biharmonic equation corresponds to the following variational problem subject to boundary conditions:

$$\frac{1}{2}\langle \Delta x, \Delta x\rangle_{\Omega_0} \to \text{ min}, \quad x \in H^2(\Omega_0) \text{ and } x = f \text{ on } \Omega_0 \setminus \Omega. \tag{4.14}$$

where f is a given function. Note that this formulation implies x, $\frac{\partial x}{\partial n}$ matching f, $\frac{\partial f}{\partial n}$ on $\partial\Omega$, because $x \in H^2(\Omega_0)$ implies $x \in C^1(\Omega_0)$. To be able to use low-order approximation spaces to solve the problem, we reformulate (4.14) using auxiliary variables. Setting $y = \Delta x$ we obtain:

$$\frac{1}{2}\langle y, y\rangle_{\Omega_0} \to \text{ min}, \, y = \Delta x \text{ on } \Omega_0 \text{ and } x = f \text{ on } \Omega_0 \setminus \Omega \tag{4.15}$$

The Lagrangian for this constrained problem has the form:

$$\frac{1}{2}\langle y, y\rangle_{\Omega_0} + \langle \lambda, \Delta x - y\rangle_{\Omega_0} + \langle \mu(1-\chi(\Omega)), x - f\rangle_{\Omega_0}$$

where λ and μ are Lagrange multipliers and χ is the characteristic function of the domain Ω. Variation with respect to the variables yields the equations:

$$\langle y, \upsilon\rangle - \langle \lambda, \upsilon\rangle = 0 \tag{4.16}$$

$$\langle \lambda, \Delta\upsilon\rangle + \langle \mu(1-\chi(\Omega)), \upsilon\rangle = 0 \tag{4.17}$$

$$\langle \Delta x - y, \upsilon\rangle = 0 \tag{4.18}$$

$$\langle x - f, (1-\chi(\Omega))\upsilon\rangle = 0, \qquad \text{for } \upsilon \in H^1(\Omega_0) \tag{4.19}$$

In all equations, after integration by parts, one can take $\upsilon \in H^1(\Omega_0)$. We use piecewise linear approximations for all variables. For an unknown function w (one of x, y, μ), we use the approximation:

$$w = \sum_{i \in N_{\Omega_0}} w_i \varphi_i$$

The vector of degrees of freedom w can be split into four components:

$$[w_\Omega,\, w_0,\, w_1,\, w_e],$$

referring to the set of vertices N_Ω, N_0, N_1 and N_e as described in Section 4.3. Similar indexing is used for stiffness and mass submatrices. We obtain the following equations:

$$\sum_{j\in N_{all}} (y_j - \lambda_j)\langle\varphi_j, \varphi_i\rangle = 0 \tag{4.20}$$

$$\sum_{j\in N_{all}} \lambda_j(\langle\frac{\partial x}{\partial n}, \varphi_j\rangle_{\Omega_0} - \langle\nabla\varphi_i, \nabla\varphi_j\rangle) + \sum_{j\in N_{all}} \mu_j\langle(1-\chi(\Omega))\varphi_j, \varphi_i\rangle = 0 \tag{4.21}$$

$$\sum_{j\in N_{all}} x_j(\langle\frac{\partial x}{\partial n}, \varphi_j\rangle_{\Omega_0} - \langle\nabla\varphi_i, \nabla\varphi_j\rangle) - \sum_{j\in N_{all}} y_j\langle\varphi_j, \varphi_i\rangle = 0 \tag{4.22}$$

$$\sum_{j\in N_{all}} (x_j - f_j)\langle(1-\chi(\Omega))\varphi_j, \varphi_i\rangle = 0 \tag{4.23}$$

From (4.20), it follows that $y_j = \lambda_j$ and λ can be eliminated.

Stiffness and mass matrix entries are defined by the following equations:

$$S_{ij} = -\langle\nabla\varphi_i, \nabla\varphi_j\rangle$$
$$\tilde{S}_{ij} = S_{ij} + \langle\varphi_j, \frac{\partial\varphi_i}{\partial n}\rangle_{\partial\Omega_0}$$
$$M_{ij} = \langle\varphi_i, \varphi_j\rangle$$

Note that one can use lumped mass matrices as a simpler alternative, since there is no significant advantage to using full mass matrices as discussed later in the chapter. A lumped mass matrix is a diagonal mass matrix with diagonal entries given by the sum of all entries in each row. Lumping mass matrices corresponds to using a single point quadrature to compute matrix entries. Evaluating φ_i and φ_j at nodes, we obtain $M_{ij} = \frac{1}{3}\delta_{ij}|V_i|$ where V_i is the union of triangles adjacent to vertex i, and $\tilde{M}_{ij} = \frac{1}{3}\delta_{ij}|V_i^\Omega|$, where V_i^Ω is the part of V_i contained in Ω.

Then, using the same subscripts $\Omega, 0, 1$ as before, we can write the system in block matrix form as:

$$S_{\Omega\Omega} y_\Omega + S_{\Omega 0} y_0 = 0 \tag{4.24}$$

$$S_{0\Omega} y_\Omega + S_{00} y_0 + S_{01} y_1 + M_{00} \mu_0 = 0 \tag{4.25}$$

$$S_{10} y_0 + S_{11} y_1 S_{1e} y_e + M_{11} \mu_1 = 0 \tag{4.26}$$

$$S_{e1} y_1 + S_{ee} y_e + M_{ee} \mu_e = 0 \tag{4.27}$$

$$S_{\Omega\Omega} x_\Omega + S_{\Omega 0} f_0 - M_{\Omega\Omega} y_\Omega = 0 \tag{4.28}$$

$$S_{0\Omega} x_\Omega + S_{00} f_0 + S_{01} f_1 - M_{00} y_0 = 0 \tag{4.29}$$

$$S_{10} f_0 + S_{11} f_1 + S_{1e} f_e - M_{11} y_1 = 0 \tag{4.30}$$

$$S_{e1} f_1 + \tilde{S}_{ee} f_e - M_{ee} y_e = 0 \tag{4.31}$$

Above, we use the equation (4.23) which reduces to $x_0 = f_0$, $x_1 = f_1$, $x_e = f_e$ to eliminate $x_0, x_1,$ and x_e.

Equations (4.25)-(4.27) are the only ones containing μ_0, μ_2, μ_e, and as these variables are not of interest to us, they can be eliminated. As (4.24) uses only y_Ω. As (4.24) uses only y_Ω and y_0, we can also eliminate (4.30), (4.31) as these are the only remaining equations with y_1 and y_e. Now the resulting system

$$S_{\Omega\Omega} y_\Omega + S_{00} y_0 = 0$$

$$S_{\Omega\Omega} x_\Omega + S_{\Omega 0} f_0 - M_{\Omega\Omega} y_\Omega = 0$$

$$S_{0\Omega} x_\Omega + S_{00} f_0 + S_{01} f_1 - M_{00} y_0 = 0$$

in matrix form can be written as :

$$\left[\begin{array}{cc|c} -M_{\Omega\Omega} & 0 & S_{\Omega\Omega} \\ 0 & -M_{00} & S_{0\Omega} \\ \hline S_{\Omega\Omega} & S_{\Omega 0} & 0 \end{array}\right] \cdot \left[\begin{array}{c} y_\Omega \\ y_0 \\ x_\Omega \end{array}\right] = \left[\begin{array}{c} -S_{\Omega 0} f_0 \\ -S_{00} f_0 - S_{01} f_1 \\ 0 \end{array}\right]$$

The system has the form

$$\left[\begin{array}{c|c} A & B \\ \hline B^T & 0 \end{array}\right]$$

typical for saddle problems.

4.5.2 Discussion

In this section we discuss and compare our method with other techniques for solving the biharmonic problem.

In [2], a similar discretization of biharmonic equation based on Ciarlet-Raviart formulation is proposed. The boundary conditions in this formulation involve one row of fixed vertices and Neumann boundary data on boundary vertices. In fact, Neumann boundary conditions are directly incorporated into the set of equations corresponding to the discretization of the weak formulation of $\Delta x = y$.

There is a relation between our formulation and that of [18], described briefly in Section 4.2.

The first observation involves the difference in mass matrices. The method of [18] is based on a diagonal mass matrix, whereas the mass matrix in our formulation is banded diagonal. Then the first step is to use lumped mass matrices in our system. A lumped mass matrix is a diagonal matrix with diagonal entries given by the sum of all entries in each row.

Unlike [18], we use auxiliary variables in our system. One can remove these variables by a simple substitution step. It turns out that this new system differs only by a weight factor from the system presented in [18] which can be removed by diagonal preconditioning.

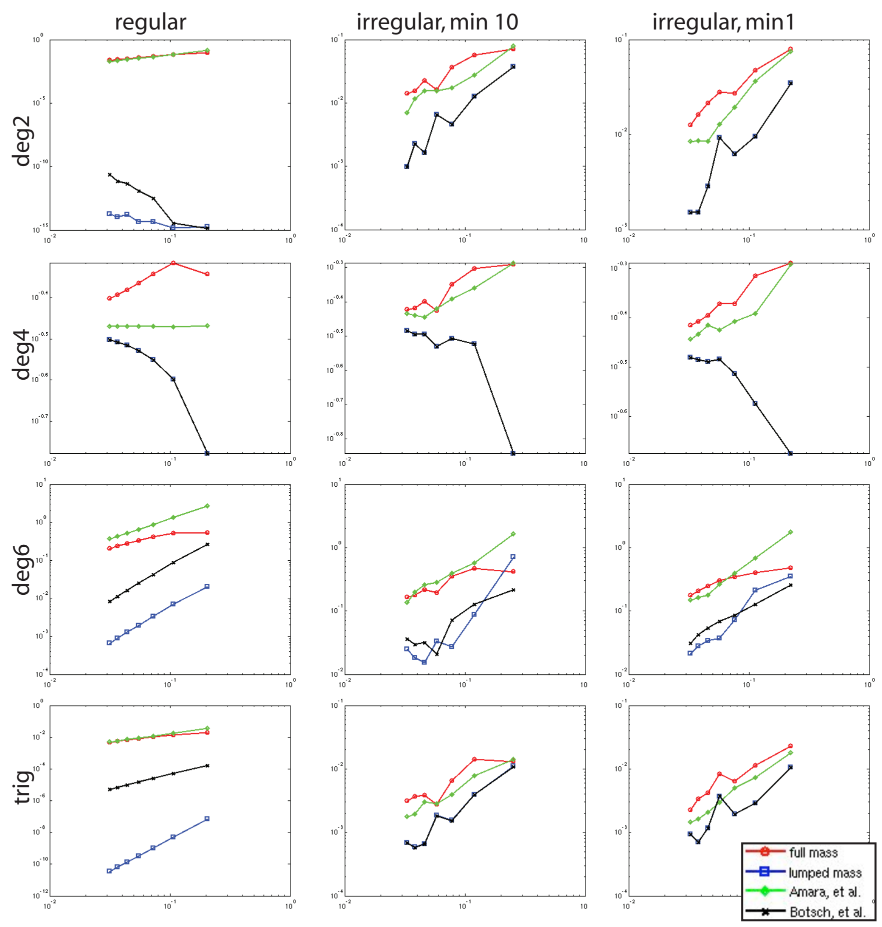

Figure 4.4: Comparison by error plots of different solutions to the biharmonic problem.

In Figure 4.4 we compare error plots for the formulations discussed here. We present results from our method with full mass matrix (red), our method with lumped mass matrix (blue), the one in [2] (green) and the one in [18] (blue). We have experimented with meshes of different connectivities from a regular mesh to an irregular mesh with minimum angle of degree 1 (See Figure 4.5). We ran recovery tests on four types of functions; a quadratic polynomial ($x^2 + xy + y^2$), a fourth-order polynomial ($x^4 + x^3y + x^2y^2 + xy^3 + y^4$), a sixth-order polynomial ($x^6 + x^5y + x^4y^2 + x^3y^3 + x^2y^4 + xy^5 + y^6$) and a trigonometric function ($cos(x) + sin(x)$).

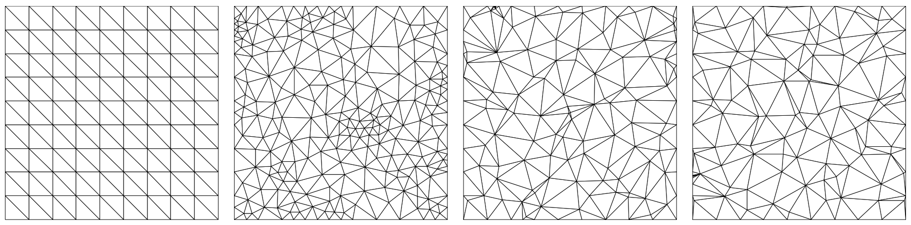

Figure 4.5: Mesh types used in tests, displayed in increasing complexity: regular, irregular with minimum angle 30, with minimum angle 10, and with minimum angle 1.

Even though the boundary conditions between our formulation and the one in [2] differ on the discretization of Neumann boundary conditions, one can see that they return very similar error values. One can also observe the convergent behavior of our method with full mass matrices and the one in [2]. Furthermore, it is clear to see the overlap of the plots for the method of [18] and ours with lumped mass matrices except for a few cases. In regular meshes, the visible difference originates from numerical errors. Note that the errors from both formulations are very small in magnitude ($< 1e{-}10$). The only other difference appears in degree six polynomial case. In this case, diagonal preconditioning leads to matching system matrices but it does not eliminate the scaling difference in the right hand sides.

4.6 Triharmonic Equation

Fourth-order problems, however useful, produce surfaces with G^1 boundaries. This low-order of smoothness can lead to dents and kinks on reflection lines and can lead to other physical problems. In Figures 4.16, 4.19 and 4.20 one can see numerous examples of these defects.

The solution to this problem is to use a higher-order PDE, for example, a sixth-order one, which would result in a G^2 surface. However, using such a high-order system may be complicated or produce ill-conditioned systems. In this section, we formulate the triharmonic problem as a mixed finite element system which allows the use of simple linear finite elements and use a regularization scheme to overcome the ill-conditioning of the system.

The triharmonic equation can be formulated as:

$$\Delta^3 x = f \tag{4.32}$$

$$x|_{\partial\Omega} = b_0 \tag{4.33}$$

$$\frac{\partial x}{\partial n}|_{\partial\Omega} = b_1 \tag{4.34}$$

$$\frac{\partial^2 x}{\partial n^2}|_{\partial\Omega} = b_2 \tag{4.35}$$

4.6.1 FEM Formulation

Triharmonic equation corresponds to minimizing $\int_\Omega (\nabla\Delta x)^2 dA$ subject to boundary conditions. Hole-filling and blending problems have function-based boundary conditions:

$$\frac{1}{2}\int_\Omega (\nabla\Delta x)^2 dA \to \min, \quad x \in H^3(\Omega_0), \quad x|_{\Omega_0\backslash\Omega} = f,$$

where f is a given function. Note that this formulation implies x, $\frac{\partial x}{\partial n}$, $\frac{\partial^x}{\partial n^2}$ matching f, $\frac{\partial f}{\partial n}$, $\frac{\partial f}{\partial n^2}$ on $\partial\Omega$, because $x \in H^3(\Omega_0)$ implies $x \in C^2(\Omega_0)$.

As in the biharmonic case, to be able to use low-order approximation spaces to solve the problem, we reformulate the problem using auxiliary variables. We extend the domain of integration of the functional to Ω_0 as $\int_{\Omega_0\backslash\Omega}(\nabla\Delta x)^2 dA$ is constant:

$$\frac{1}{2}\int_{\Omega_0}(\nabla y)^2 dA \to \min, \text{ subject to } x|_{\Omega_0\setminus\Omega} = f, \Delta x = y \text{ on } \Omega_0 \tag{P}$$

The Lagrangian for the constrained problem (P) is

$$\frac{1}{2}\langle \nabla y \nabla y\rangle_{\Omega_0} + \langle \lambda, \Delta x - y\rangle_{\Omega_0} + \langle \mu(1-\chi(\Omega)), x - f\rangle_{\Omega_0}$$

where, as before, λ,μ are Lagrange multipliers and χ is the characteristic function of the domain Ω. Variation with respect to y, x, λ, μ leads to equations:

$$\langle \nabla y, \nabla \upsilon\rangle_{\Omega_0} - \langle \lambda, \upsilon\rangle_{\Omega_0} = 0 \qquad (\text{for } y) \tag{4.36}$$

$$\langle \lambda, \Delta \upsilon\rangle_{\Omega_0} + \langle \mu(1-\chi(\Omega)), \upsilon\rangle_{\Omega_0} = 0 \qquad (\text{for } x) \tag{4.37}$$

$$= \langle \lambda, \frac{\partial \upsilon}{\partial n}\rangle_{\partial\Omega_0} - \langle \nabla\lambda, \nabla \upsilon\rangle_{\Omega_0} + \langle \mu(1-\chi(\Omega)), \upsilon\rangle_{\Omega_0} = 0 \tag{4.38}$$

$$\langle \Delta x - y, \upsilon\rangle_{\partial\Omega_0} = \langle \frac{\partial x}{\partial n}, \upsilon\rangle_{\partial\Omega} - \langle \nabla x \nabla \upsilon\rangle_{\Omega_0} - \langle y, \upsilon\rangle_{\Omega_0} = 0 \tag{4.39}$$

$$\langle x - f, (1-\chi(\Omega))\upsilon\rangle_{\Omega_0} = 0 \qquad (\text{for } \mu) \tag{4.40}$$

In all equations, after integration by parts one can take $\upsilon \in H^1(\Omega_0)$.

Remark. Note that in strong form we get:
from (4.36):

$$\Delta y = -\lambda \text{ on } \Omega_0.$$

from (4.37):

$$\Delta\lambda = \begin{cases} \mu \text{ on } \Omega_0 \setminus \Omega. \\ 0 \text{ on } \Omega. \end{cases}$$

from (4.39):

$$\Delta x = y \text{ on } \Omega_0.$$

from (4.40):

$$x = f \text{ on } \Omega_0 \setminus \Omega.$$

Setting $-\lambda = z$, we get $\Delta x = y$, $\Delta y = z$, $\Delta z = 0$ on Ω, and $x = f$ on $\Omega_0 \setminus \Omega$. Setting $y = \Delta f$, $z = \Delta^2 f = \mu$ on $\Omega_0 \setminus \Omega$ satisfies (4.39), (4.37), and (4.36) on $\Omega_0 \setminus \Omega$. We observe that the triharmonic equation is recovered as expected; in addition x is required to be C^2 to satisfy (4.39) on Ω_0, in particular satisfy $\frac{\partial x}{\partial n} = \frac{\partial f}{\partial n}$, $\frac{\partial^2 x}{\partial n^2} = \frac{\partial^2 f}{\partial n^2}$ on $\partial\Omega$.

Returning to the weak-form equations (4.36) - (4.40), we choose piecewise linear elements for x, y, z, λ, μ and test functions v.

For an unknown function w (one of x, y, z, μ); we use the approximation:

$$w = \sum_{i \in N_{\Omega_0}} w_i \varphi_i$$

The vector of degrees of freedom w can be split into five components : $[w_\Omega$, w_0, w_1, w_2, w_e]. Similar indexing is used for stiffness and mass submatrices. We obtain the following equations:

$$\sum_{j \in N_{all}} y_j \langle \nabla\varphi_j, \nabla\varphi_i \rangle + \sum_{j \in N_{all}} z_j \langle \varphi_j, \varphi_i \rangle = 0 \qquad (4.41)$$

$$\sum_{j \in N_{all}} z_j (\langle \varphi_j, \frac{\partial \varphi_i}{\partial n} \rangle_{\Omega_0} + \langle \nabla\varphi_j, \nabla\varphi_i \rangle_{\Omega_0}) - \sum_{j \in N_{all}} \mu_j \langle (1 - \chi(\Omega))\varphi_j, \varphi_i \rangle = 0 \qquad (4.42)$$

$$\sum_{j \in N_{all}} x_j (\langle \frac{\partial \varphi_j}{\partial n}, \varphi_i \rangle_{\partial\Omega_0} - \langle \nabla\varphi_j, \nabla\varphi_i \rangle_{\Omega}) - \sum_{j \in N_{all}} y_j \langle \varphi_j, \varphi_i \rangle = 0 \qquad (4.43)$$

$$\sum_{j \in N_{all}} (x_i - f_i) \langle \varphi_j, (1 - \chi(\Omega))\varphi_i \rangle = 0 \qquad (4.44)$$

Furthermore, we assume that $\partial\Omega$ is at least 2 rows removed from $\partial\Omega_0$. In matrix form we have:

$$\begin{aligned}
-Sy + Mz &= 0 \\
-\tilde{S}^T z - \tilde{M}\mu &= 0 \\
\tilde{S}^T x - My &= 0 \\
\tilde{M}^T(x - f) &= 0
\end{aligned}$$

where

$$\begin{aligned}
S_{ij} &= -\langle \nabla\varphi_i, \nabla\varphi_j \rangle \\
\tilde{S}_{ij} &= S_{ij} + \langle \varphi_j, \frac{\partial \varphi_i}{\partial n} \rangle_{\partial\Omega_0} \\
M_{ij} &= \langle \varphi_i, \varphi_j \rangle \\
\tilde{M}_{ij} &= \langle (1 - \chi(\Omega))\varphi_j, \varphi_i \rangle
\end{aligned}$$

We further simplify this discretization by using lumped mass matrices (See Section 4.5.1). Clearly $\tilde{M}_{ij} = 0$ for all vertices in the interior of Ω. From (4.44) we conclude that

$$x_i = f_i, \qquad \forall i \in N_0 \cup N_1 \cup N_2 \cup N_2 \tag{4.45}$$

From (4.42) we observe that μ_i, $i \in N_{int}$ are not a part of any equation and can be eliminated. μ_i, $i \in N_{all} \setminus N_{int}$ each appear in only one of the equations (4.42). As the values of μ are not of interest to us, these equations can be eliminated and no equations depend on μ_i any longer.

The remaining equations in matrix form can be written as:

$$S_{\Omega\Omega}y_\Omega + S_{\Omega 0}y_0 - M_\Omega z_\Omega = 0 \tag{4.46}$$

$$S_{0\Omega}y_\Omega + S_{00}y_0 + S_{01}y_1 - M_0 z_0 = 0 \tag{4.47}$$

$$S_{10}y_0 + S_{11}y_1 + S_{12}y_2 - M_1 z_1 = 0 \tag{4.48}$$

$$S_{21}y_1 + S_{22}y_2 + S_{2e}y_e - M_2 z_2 = 0 \tag{4.49}$$

$$S_{e2}y_2 + \tilde{S}_{ee}^T y_e - M_e z_e = 0 \tag{4.50}$$

$$S_{\Omega\Omega}z_\Omega + S_{e0}z_0 = 0 \tag{4.51}$$

$$S_{\Omega\Omega}x_\Omega + S_{\Omega 0}f_0 - M_\Omega y_\Omega = 0 \tag{4.52}$$

$$S_{0\Omega}x_\Omega + S_{00}f_0 + S_{01}f_1 - M_0 y_0 = 0 \tag{4.53}$$

$$S_{10}f_0 + S_{11}f_1 + S_{12}f_2 - M_1 y_1 = 0 \tag{4.54}$$

$$S_{21}f_1 + S_{22}f_2 + S_{2e}f_e - M_2 y_2 = 0 \tag{4.55}$$

$$S_{e2}f_2 + \tilde{S}_{ee}^T f_e - M_e y_e = 0 \tag{4.56}$$

where we have already used the condition (4.45).

Observe that the system can be split into independent systems for x_Ω, y_Ω, y_0, y_1, z_Ω, z_0 consisting of (4.46), (4.47), (4.51),(4.52), (4.53), (4.54), with the last set of equations containing only one unknown each.

One can easily see that the resulting reduced system, up to scaling factors is identical to the system in [18] obtained using discrete geometry point of view if applied to the mesh with vertices $N_\Omega \cup N_0 \cup N_1 \cup N_2$ with three rows of x fixed.

Further reducing the system by eliminating y_1 using (4.54), we can write the system matrix as

$$\begin{array}{|cc|c|cc|}
\hline
S_{\Omega\Omega} & S_{\Omega 0} & 0 & -M_{\Omega} & 0 \\
S_{0\Omega} & S_{00} & 0 & 0 & \text{-}M_0 \\
\hline
0 & 0 & 0 & S_{\Omega\Omega} & S_{\Omega 0} \\
\hline
-M_{\Omega} & 0 & S_{\Omega\Omega} & 0 & 0 \\
0 & -M_0 & S_{0\Omega} & 0 & 0 \\
\hline
\end{array}$$

with columns corresponding to $y_\Omega, y_0, x_\Omega, z_\Omega, z_0$ and rows to discretization of $\Delta y = z$ on Ω, $\Delta y = z$ on $\partial\Omega$, $\Delta z = 0$ on Ω, $\Delta x = y$ on Ω and $\Delta x = y$ on $\partial\Omega$. The right hand side has the form :

$$[\,0,\ -S_{01}y_1,\ 0,\ -S_{\Omega 0}f_0,\ -S_{00}f_0 - S_{01}f_1],$$

with

$$y_1 = -S_{10}f_0 - S_{11}f_1 - S_{12}f_2.$$

That is, the system has the form

$$\begin{array}{|c|c|}
\hline
A & B \\
\hline
B^T & 0 \\
\hline
\end{array}$$

for which the inf-sup condition can be studied. Experimental observations show that whether theoretically the discretization yields a convergent result or not, the convergence may be slow for some types of meshes.

A modification of discretization of problem (P) makes it possible to use more accurate approximation of boundary data. We start with the same variational formulation but use the constraint $x = f$ on $\Omega_0 \setminus \Omega$ to change the form of $\langle \Delta x - y, v\rangle = 0$.

We write this equation in different equivalent forms depending on v. If $supp(v) \cap \Omega = \varnothing$, then we substitute $x = f$ and change the equation to $\langle \Delta f - y, v\rangle = 0$. If $supp(v) \cap \Omega \neq \varnothing$, we keep it in the same form.

For the new discretization we additionally assume that $g = \Delta f$ can be either evaluated directly on $\Omega_0 \setminus \Omega$ or approximated to a sufficiently high-order using e.g. local polynomial fits.

The system (4.41)-(4.44) is modified by replacing (4.43) with

$$\sum_{j \in N_{all}} x_j (\langle \frac{\partial \varphi_j}{\partial n}, \varphi_i \rangle_{\partial\Omega} - \langle \nabla \varphi_j, \nabla \varphi_i \rangle_{\Omega_0}) - \sum_{j \in N_{all}} y_j \langle \varphi_i, \varphi_j \rangle = 0, \quad \forall i \in N_\Omega \cup N_0. \tag{4.57}$$

As we assume that $\partial\Omega_0$ is at least two edges away from $\partial\Omega$ in any triangulation, $\langle \frac{\partial \varphi_j}{\partial n}, \varphi_i \rangle$ can be dropped.

$$\sum_{j \in N_{all}} g_j \langle \varphi_i, \varphi_j \rangle_{\Omega_0} - \sum_{j \in N_{all}} y_j \langle \varphi_i, \varphi_j \rangle_{\Omega_0} = 0, \quad \forall i \in N_{all} \setminus (N_\Omega \cup N_0). \tag{4.58}$$

After this change in the equations, the matrix equations (4.52)-(4.56) are replaced with:

$$S_{ee} x_e + S_{e0} f_0 - M_\Omega y_\Omega = 0 \tag{4.59}$$

$$S_{0\Omega} x_\Omega + S_{00} f_0 + S_{01} f_1 - M_0 y_0 = 0 \tag{4.60}$$

$$M_1 (g_1 - y_1) = 0 \tag{4.61}$$

$$M_2 (g_2 - y_2) = 0 \tag{4.62}$$

$$M_2 (g_e - y_e) = 0 \tag{4.63}$$

Observe that f_2, f_e are no longer used in the system and equation (4.61) leads to $y_1 = g_1$. Resulting complete system:

$$\begin{aligned}
S_{\Omega\Omega}y_\Omega + S_{\Omega 0}y_0 - M_\Omega z_\Omega &= 0 \\
S_{0\Omega}y_\Omega + S_{00}y_0 + S_{01}g_1 - M_0 z_0 &= 0 \\
S_{\Omega\Omega}z_\Omega + S_{\Omega 0}z_0 &= 0 \\
S_{\Omega\Omega}x_\Omega + S_{\Omega 0}f_0 - M_\Omega y_\Omega &= 0 \\
S_{0\Omega}x_\Omega + S_{00}f_0 + S_{01}f_1 - M_0 y_0 &= 0 \qquad (4.64)
\end{aligned}$$

has the same matrix but a different right hand side. Effectively, an approximation to y, of the form $M_1^{-1}(S_{11}f_0S_{11}f_1 + S_{12}f_2)$ is replaced with g_i, i.e. exact or more accurate approximate values of Δf.

As discussed in Section 4.8, this leads to substantial improvements for a number of meshes. It is natural to ask if the need for explicit use of f_1 (i.e. values of x one row outside Ω can be eliminated by similar means. This may be possible by substitution of $\Delta x = \Delta f$ everywhere outside Ω even for v with $supp(v) \cap \Omega \neq \varnothing$ and integration by parts, but we did not explore this direction further.

4.6.2 Discussion

We observed that the differential-type boundary conditions prescribed for the biharmonic as described in [2] and reviewed in Section 4.5.2 cannot be used for the triharmonic problem.

Consider the system of equations resulting from the discretization of $\Delta x = y$. When written in the mixed formulation of [2], one would get:

$$\sum_{i\in N_\Omega} x_i \int \nabla\varphi^i \nabla\varphi^j dA + \sum_{i\in N_\Omega} y_i \int \varphi^i\varphi^j dA = -\sum_{i\in N_0} b_0^i \int \nabla\varphi^i\nabla\varphi^j dA -$$
$$\sum_{i\in N_0} b_3^i \int \varphi^i\varphi^j dA - \int_{\partial_\Omega} b_1\varphi^j dL, \quad \forall j \in (N_\Omega) \cup (N_0). \qquad (4.65)$$

Besides the additional fixed row of vertices in our formulation, this formulation differs in the number of equations. In this system there are n_0 more equations of this type, where n_0 is the cardinality of the set N_0. Given the Dirichlet boundary conditions for x and y in the triharmonic formulation, this set of equations lead to a system with n_1 x variables for the row of vertices one edge away from the boundary. This $n_0 \times n_1$ system is generally over- or under-determined.

Geometrically, this issue arises when more than one edge on the boundary is incident to an interior vertex. All equations written for the endpoints of these edges will involve one unknown, namely x for the interior vertex. In the case with multiple incident boundary edges, this leads to linearly dependent rows resulting in a singular matrix.

Such connectivity is common on meshes; even a regular mesh has a null space. On a rectangular domain this singularity occurs near corners. In Figure 4.6 we show two lower corners of a regular $n \times n$ mesh. The red circles display the interior vertices where singularity occurs and no solution exists. The darkened edges display the edges that are incident to the interior vertex and cause the singularity.

In this case, the resulting system matrix has eight eigen values that are zero: two per corner. The eigen vectors for these are such that the entry for the corner vertex (e.g, $p_{0,0}$ in figure) is 1, and the rest are zeros, or the two orthogonal neighbors of the corner vertex (e.g., $p_{1,0}$ and $p_{0,1}$ in figure) are 1 and -1 respectively, and the rest are zeros.

This phenomenon is a result of the small support of basis functions used and cannot be avoided in this formulation. This shows that our formulation with two fixed rows of x and one fixed row of y works better than extending to the

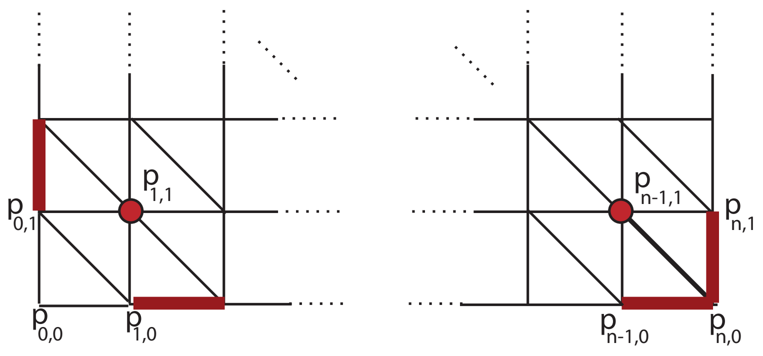

Figure 4.6: Corner singularities on a regular mesh. The red circles display the vertices where singularity occurs and the darkened edges are those that cause the singularity.

triharmonic the mixed FEM formulations proposed for solving the biharmonic equation.

Now we consider function-based boundary conditions. We have mentioned the function-based formulation described in [18], where three rows of boundary vertices are fixed. Similar to our discussion in the biharmonic case, one can show a connection between the two methods. As explained earlier in this chapter, replacing the mass matrix with the lumped mass matrix and flattening the system by removing auxiliary variables reduces our system to the one in [18], different by a scale factor.

We remove the auxiliary variables by writing the variables y_Ω, y_0, z_Ω and z_0 in terms of x only, and then substituting into the equations. When applied to the system in (4.64), we get:

$$
\begin{aligned}
&(S_{\Omega\Omega}M_\Omega^{-1}(S_{\Omega\Omega}M_\Omega^{-1}S_{\Omega\Omega} + S_{\Omega 0}M_0^{-1}S_{0\Omega})+\\
&\ S_{00}M_0^{-1}(S_{0\Omega}M_\Omega^{-1}S_{\Omega\Omega} + S_{00}M_0^{-1}S_{0\Omega}))x_\Omega =\\
&r_1(-S_{\Omega\Omega}M_\Omega^{-1}) + r_2(-S_{00}M_0^{-1}) + r_3\\
&r_4(S_{\Omega\Omega}M_\Omega^{-1}S_{\Omega\Omega}M_\Omega^{-1} + S_{00}M_0^{-1}S_{0\Omega}M_\Omega^{-1})+\\
&r_5(S_{\Omega\Omega}M_\Omega^{-1}S_{\Omega 0}M_0^{-1} + S_{00}M_0^{-1}S_{00}M_0^{-1})
\end{aligned}
\tag{4.66}
$$

where $[r_1, r_2, r_3, r_4, r_5]$ is the right hand side of the full system; i.e. $r_1 = 0$, $r_2 = -S_{01}g_1$, $r_3 = 0$, $r_4 = -S_{\Omega 0}f_0$ and $r_5 = -S_{00}f_0 - S_{01}f_1$ in system (4.64) .

Figure 4.7 shows error plots for several formulations spanning the gap between our methods. We show plots for recovery errors in four different formulations. Our method with fixing two rows of x and one row of y (in red), same formulation but with lumped mass matrices (blue), lumped matrix setup with three rows of x (black) fixed and finally the method from [18] which also has three rows of x fixed (green). One can see that our system with lumped mass matrices produces results very similar to our unmodified system with converging error but the error magnitudes are slightly larger. Note that the increase in error in the degree two recovery on a regular mesh is a result of numerical errors and is negligible considering the magnitudes ($\approx 1e-14$). When we fix three rows however, convergence is lost and the system is equivalent to that presented in [18].

4.7 Regularization

The problem with the instability of mixed finite element systems is well studied in literature [4, 6, 23]. As we also observed, the condition numbers increase very quickly to very large values as one increases the resolution of the input mesh,

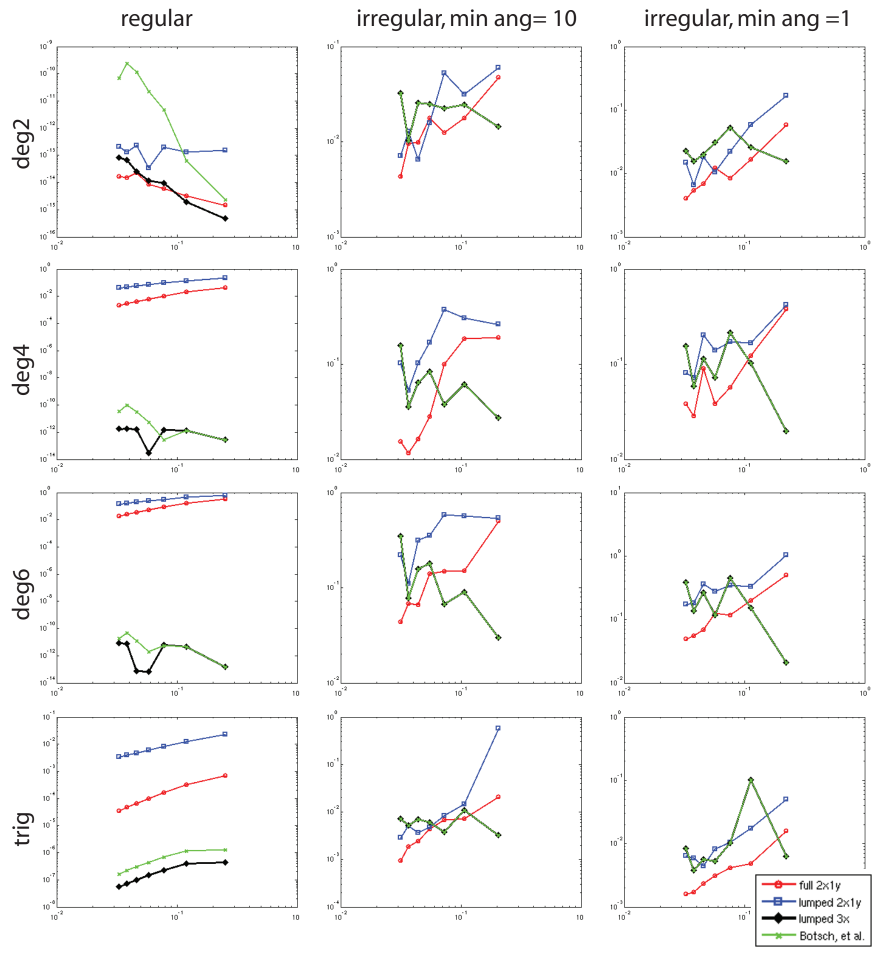

Figure 4.7: Plots for some variations of our method, using function-based boundary conditions.

especially in cases of irregular connectivity (See Figure 4.8).

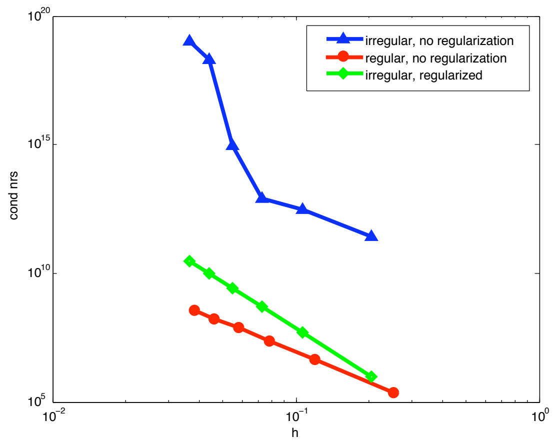

Figure 4.8: Effect of regularization on the condition number of the mixed FEM system for irregular meshes.

To avoid this instability problem, we present a regularization scheme based on Tikhonov regularization, also known as ridge regression in statistics. Regularization maintains that the regularized solution is not too sensitive to perturbations. The simplest version of Tikhonov regularization involves modifying the ill conditioned system matrix by adding an identity matrix (I), scaled by some parameter(α) commonly referred to as the regularization parameter (Equation 4.67).

$$x = (A + \lambda I)^{-1} b \tag{4.67}$$

Another similar technique is to replace the identity with a different matrix, for example, the gradient matrix. The rationale behind this additional matrix is the conversion of the linear system to a minimization problem. Instead of solving the system directly, we find an x that minimizes the following weighted combination of the norms of the residual and the solution.

$$x = argmin\{||Ax - b||^2 + \lambda^2||Lx||^2\} \tag{4.68}$$

In this formulation the added term is referred to as a side constraint([83]), and maintains that the solution or a linear function of the solution stays small. The regularization parameter then affects the importance of this side constraint.

The main difficulty of regularization is the accurate selection of regularization parameters, since there is no reliable way to determine regularization parameters when there is no information available on residuals and smoothness [133]. Many methods for finding a good regularization parameter were proposed in literature. However, all methods strongly depend on the problem being solved [84].

One of the most commonly used methods for determining the regularization parameter is the L-curve method described in [82]. This method utilizes the plot for norm of the solution versus the norm of the residual, and locates the "corner" of this graph to compute the regularization parameter.

Given the singular value decomposition of the matrix $A = USV^T$, Tikhonov regularization replaces the matrix A with

$$\bar{A} = (A^TA + \lambda^2 I)^{-1}A^T$$

which has the singular value decomposition

$$\bar{A} = V((S^2 + \lambda^2 I)^{-1}S)U.$$

The singular values of this new matrix is then given by

$$\bar{\sigma}_i = \sigma_i/(\sigma_i^2 + \lambda^2).$$

The idea behind the method described in [82] is to replace the original singular values with these in order to get a better conditioned system. One can compute values for $||Ax - b||$ and $||Lx||$ using these new singular values coupled with the orthonormal matrices from the singular value decomposition in order to plot the L-curve. The *corner* is the point of highest curvature of the L-curve.

We have experimented with regularization tools from [83, 85] which implement this method. We have observed that using this technique in our setting leads to non-converging or slowly converging solutions. In fact, in some cases the L-curve did not even have the characteristic L-shape for the optimal parameter to be defined correctly. In Figure 4.9 we show examples of L-curves we have observed in our experiments. One can see that while in some cases the shape has a well-defined corner (leftmost example, using an irregular mesh with minimum angle 30), in some others there are multiple corners (middle two examples, using irregular meshes with minimum angles 10 and 1 respectively), or no corners or an L-shaped curve at all (rightmost example, using a regular mesh). The red dashed line points to the "corner" as selected by the algorithm. Furthermore, since this method requires a full SVD decomposition of the system matrix it is space and time inefficient.

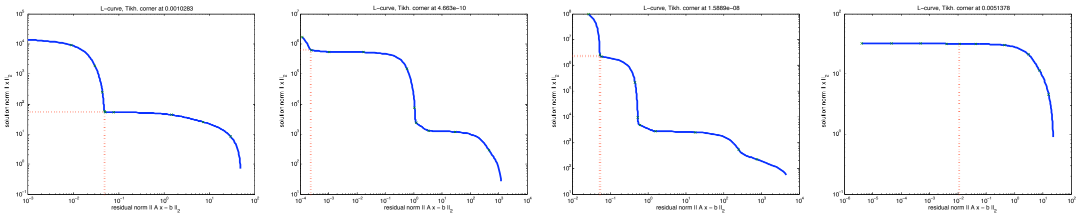

Figure 4.9: Some resulting L-curves from our experiments.

Instead, we utilize a similar but faster setup for our purposes. We compute a regularization parameter for a given problem by prescribing a condition number to each mesh resolution. Specifically, we pick a condition number for the coarsest mesh and the rate of increase with mesh resolution.

In our experiments, using a condition number of about $1e{+}6$ for a given value of a coarse mesh resolution (.1) and a rate of h^{-6} or h^{-8} worked best. Mesh resolution is computed as h/L, with h shortest edge length and L total mesh size. Given this starting condition number κ_0^*, the maximum edge length h_0 of the coarsest mesh and the maximum edge length h_i for a given mesh, we can compute the new prescribed condition number κ_i as:

$$\kappa_i^* = \kappa_0^* (h_i/h_0)^{(-s)}$$

where s is the prescribed rate of divergence.

Now that we have prescribed condition number κ^*, we base our formulation on the following expression relating the regularization parameter and the condition number of the regularized matrix.

$$\lambda = \frac{\sigma_{max}}{\kappa^*} - \sigma_{min}$$

Note that this equation is satisfied exactly when no regularization is used ($\lambda = 0$ and $\kappa^* = \kappa = \frac{\sigma_{max}}{\sigma_{min}}$). We observe that in the case of an ill-conditioned problem, σ_{min} is very close to zero. Then we can omit this term and compute an approximate regularization parameter given by:

$$\lambda = \frac{\sigma_{max}}{\kappa^*}$$

This simple modification increases the efficiency of our scheme greatly, since now the regularization parameter λ depends on the largest singular value only. Given that we start with a sparse matrix, we can use an algorithm that computes the largest singular value only, such as *svds* in MATLAB, instead of a full singular value decomposition. If computing the largest singular value is still prohibitively expensive, it can be estimated analytically on regular grids, and used for arbitrary grids with similar mesh resolution (h/L) values.

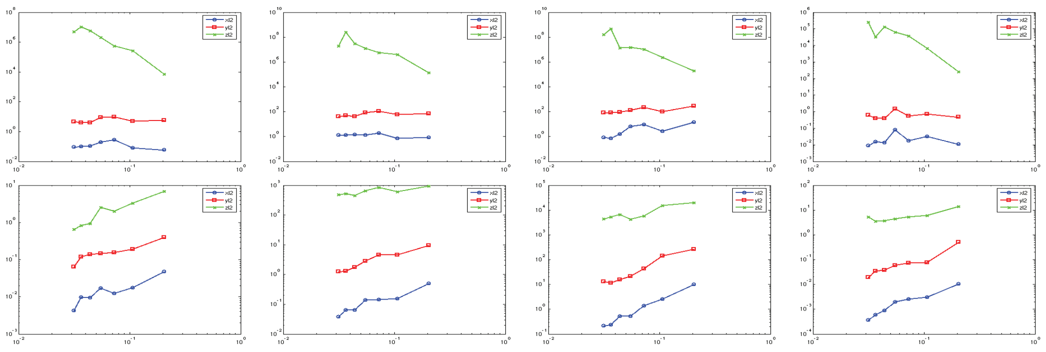
Figure 4.10: Error plots with (below) and without (above) regularization.

In Figure 4.10 we show a series of error plots without regularization (above) and with regularization(below). We observe that without regularization, errors do not converge, especially for auxiliary variables. The error plots are for an irregular mesh with minimum angle 10, and shows errors in recovery of a quadratic polynomial, a sixth-order polynomial, a trigonometric and a sixth degree spline bump from left to right. The exact functions used and more error plots showing our convergent scheme are provided in Section 4.8.

We have mentioned the relation between our mixed formulation with auxiliary variables and the flattened system with no auxiliary variables in [18]. Having observed that regularization works well with our formulation, we experimented with applying regularization to the flattened system. There are two choices for applying regularization when dealing with the system with no auxiliary variables. One can choose to remove the auxiliary variables first, and then apply regularization or one can apply regularization to the full matrix and then remove the auxiliary variables. In order to use the former formulation we regularize the flattened system given in equation (4.66). For the latter formulation, we need to reorganize the matrix so that the mass matrices lie on the diagonal. Doing so circumvents introducing extra dependencies among the variables which guarantees that the system can be flattened. For our implementation we used the following re-organized system:

$$\left[\begin{array}{cc|cc|c} M_\Omega & 0 & 0 & 0 & S_{\Omega\Omega} \\ 0 & M_0 & 0 & 0 & S_{0\Omega} \\ \hline S_{\Omega\Omega} & S_{\Omega 0} & M_\Omega & 0 & 0 \\ S_{0\Omega} & S_{00} & 0 & M_0 & 0 \\ \hline 0 & 0 & S_{\Omega\Omega} & S_{\Omega 0} & 0 \end{array}\right] \left[\begin{array}{c} y_\Omega \\ y_0 \\ \hline z_\Omega \\ z_0 \\ \hline x_\Omega \end{array}\right] = \left[\begin{array}{c} -S_{\Omega 0} f_0 \\ -S_{00} f_0 - S_{01} f_1 \\ \hline 0 \\ -S_{01} y_1 \\ \hline 0 \end{array}\right]$$

When regularized, the matrix of this system gets updated on the diagonal only, i.e. only the mass matrices get updated, and the last zero diagonal block entry becomes non-zero. Let us denote the regularized mass matrices by $\tilde{M}_*$, and let $\tilde{I}$ be an identity matrix scaled by the regularization parameter. Then the regularized system of equations becomes:

$$\begin{aligned} \tilde{M}_\Omega y_\Omega + S_{\Omega\Omega} x_\Omega &= -S_{\Omega 0} f_0 \\ \tilde{M}_0 y_0 + S_{0\Omega} x_\Omega &= -S_{00} f_0 - S_{01} f_1 \\ S_{\Omega\Omega} y_\Omega + S_{\Omega 0} y_0 + \tilde{M}_\Omega z_\Omega &= 0 \\ S_{0\Omega} y_\Omega + S_{00} y_0 + \tilde{M}_0 z_0 &= -S_{01} y_1 \\ S_{\Omega\Omega} z_\Omega + S_{00} z_0 + \tilde{I} x_\Omega &= 0 \end{aligned}$$

Let the right hand side be the vector $[r_1, r_2, r_3, r_4, r_5]$. We then flatten this system to get the following equation and then solve.

$$
\begin{aligned}
&(S_{\Omega\Omega}\widetilde{M^{-1}}{}_{\Omega}(S_{\Omega\Omega}\tilde{M}_{\Omega}^{-1}S_{\Omega\Omega}+S_{\Omega 0}\tilde{M}_{0}^{-1}S_{0\Omega})+\\
&S_{00}\tilde{M}_{0}^{-1}(S_{0\Omega}\tilde{M}_{\Omega}^{-1}S_{\Omega\Omega}+S_{00}\tilde{M}_{0}^{-1}S_{0\Omega})+\tilde{I})x_{\Omega}=\\
&r_1(S_{\Omega\Omega}\tilde{M}_{\Omega}^{-1}S_{\Omega\Omega}\tilde{M}_{\Omega}^{-1}+S_{00}M_{0}^{-1}S_{0\Omega}\tilde{M}_{\Omega}^{-1})+\\
&r_2(S_{\Omega\Omega}\tilde{M}_{\Omega}^{-1}S_{\Omega 0}\tilde{M}_{0}^{-1}+S_{00}\tilde{M}_{0}^{-1}S_{00}\tilde{M}_{0}^{-1})\\
&r_3(-S_{\Omega\Omega}\tilde{M}_{\Omega}^{-1})+r_4(-S_{00}\tilde{M}_{0}^{-1})+r_5
\end{aligned}
$$

We have observed that regularization does not have the same effect on flattened systems regardless of when one applies it, resulting in non-convergent results in both cases. In Figure 4.11 we show error plots from recovery experiments of three different functions on an irregular mesh.

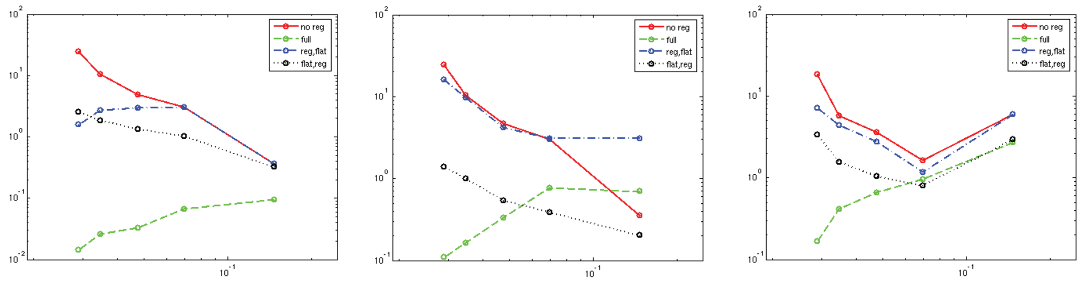

Figure 4.11: Experiments in regularization for flattened systems. Left to right: degree two polynomial, degree six polynomial and a trigonometric function.

We used a degree two polynomial, a degree six polynomial and a high frequency trigonometric function for these experiments (exact functions used can be found in Section 4.8). One can see that applying regularization before (blue curve) or after (black curve) the flattening step does not lead to any substantial improvement: although error is less in magnitude when regularization is applied to the flattened system, in both cases error magnitudes diverge, as they do in the non-regularized case.

4.8 Results

We start with experiments exploring the convergence of our method. The error plots in Figure 4.12 show the change in L_2 error magnitudes in logarithmic scale when our mixed method and the method in [18] are used. The results are gathered from recovery tests, where the boundary conditions and non-homogeneous right hand sides (when the polyharmonic equation does not equal zero) are sampled from a known function and the error is computed as the difference between exact and computed results. We ran tests on several types of functions; a quadratic polynomial ($x^2 + xy + y^2$), a sixth-order polynomial ($x^6 + x^5y + x^4y^2 + x^3y^3 + x^2y^4 + xy^5 + y^6$), a high frequency trigonometric function ($cos(5x) + sin(10y)$), and degree 6 spline tensor product patch. The spline patch is a sixth degree bump of height 1 and with span [-2.5,2 .5] $\times$ [-3.5 1.5].

For each function we ran tests on four different sets of meshes with increasing resolution. We used regular meshes, irregular meshes with minimum angle 30, with minimum angle 10, and with minimum angle 1 as shown in Figure 4.5. We increase the resolution by generating a denser set of points and triangulating where the only constraint was the bound on minimum angle. For our method we show error plots for both primal and auxiliary variables. Error in the primal variable x is displayed with color blue, first auxiliary variable y is displayed in red, and the second auxiliary variable z is displayed in green. For the method in [18] error plots are for x only.

One can observe that our method converges consistently for all functions and mesh connectivities. The only visible divergence happens in the quadratic function-regular mesh pair (Figure 4.12, but in this case the function was fully recovered (note the error magnitudes), and the increasing error is due to numerical error. On the other hand, the error magnitudes for the method in [18] stays low, however converges for only one function (Figure 4.12, (g)).

We compute the rate of convergence k, as in $O(h^k)$, of error magnitudes for x, y and z in each of our test cases. This is documented in Table 4.1. Three entries per cell refer to the errors in x, y and z respectively. One can observe that while we can almost always achieve faster than linear convergence for x, but the rate of convergence goes down for y and z.

	regular			min ang=30			min ang=10			min ang = 1		
degree 2	rec	rec	rec	1.6	1.4	1.3	1.3	1.1	1.3	1.4	1.0	0.6
degree 6	1.7	1.7	0.9	1.5	1.3	0.3	1.4	1.1	0.3	1.4	1.2	0.4
trigonometric	2.0	1.9	1.5	2.4	2.0	1.1	2.2	2.4	1.0	2.6	1.3	0.9
spline (deg 6)	0.7	1.9	1.1	1.4	1.6	0.5	1.5	1.6	0.5	1.8	0.8	0.6

Table 4.1: Rate of convergence for the mixed method for various functions and mesh connectivities, as computed from plots in Figure 4.12.

We have also observed that using different functions within the same polynomial space does not effect convergence rates. For example, in Figure 4.13 we present error plots for four different sixth degree polynomials. From left to right we have: $x^6+x^5y+x^4+y^2+x^3y^3+x^2y^4+xy^5+y^6$ (same as above), x^6+y^6, $-10x^6+10x^4y^2-10x^2y^4+10y^6$, and $1000x^6+y$, all computed on an irregular mesh with minimum angle 10. One can see that although error magnitudes are different, the rate of convergence as displayed by the slopes of curves is approximately the same for all functions. We have experimented with test polynomials with both positive and

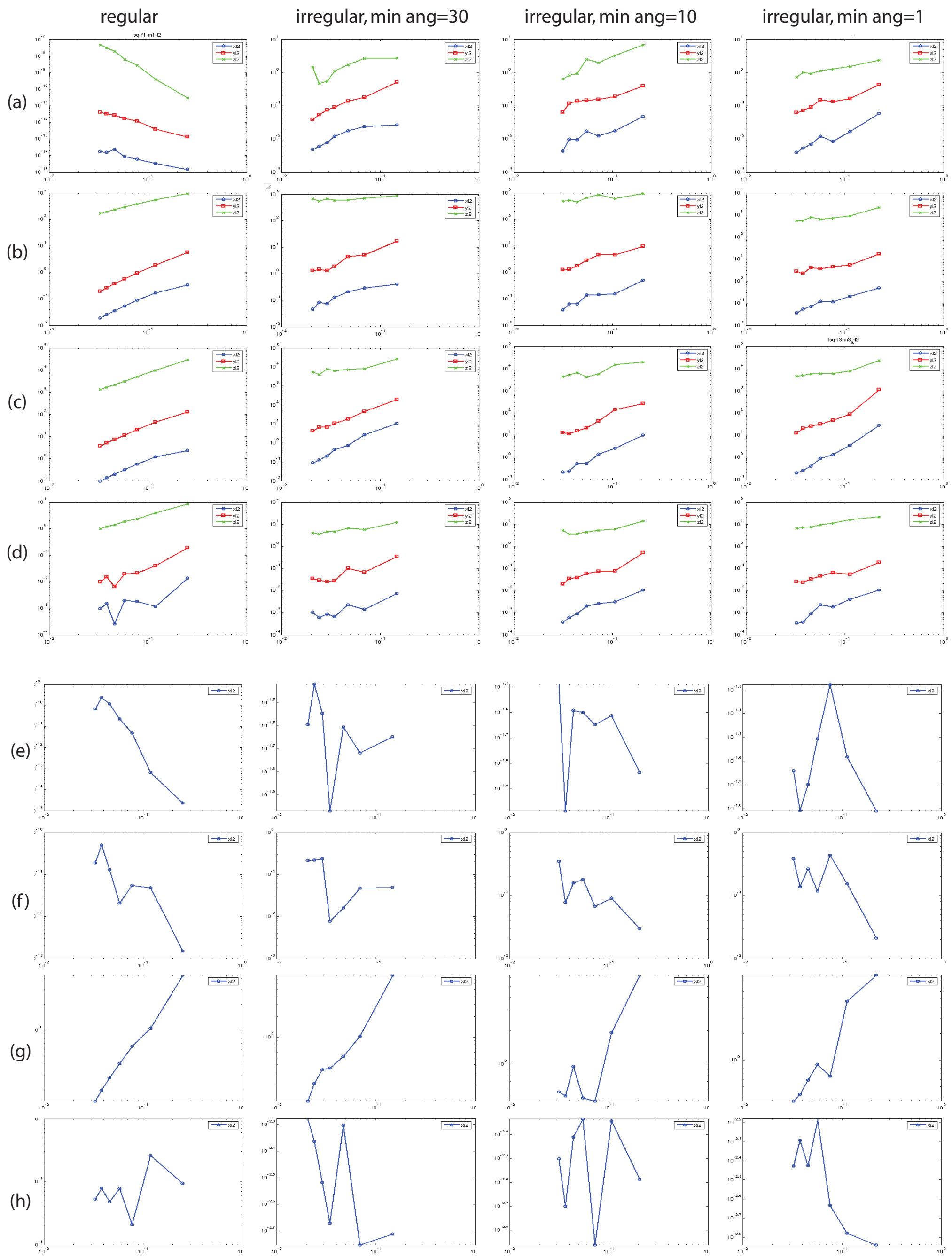

Figure 4.12: Convergence plots for the mixed method.(a),(e): quadratic; (b),(f): sixth-order; (c)(g): high frequency trigonometric function; (d),(h): degree 6 spline bump.

negative coefficients, with different local behaviors (convex, concave,and saddle) and obtained similar results.

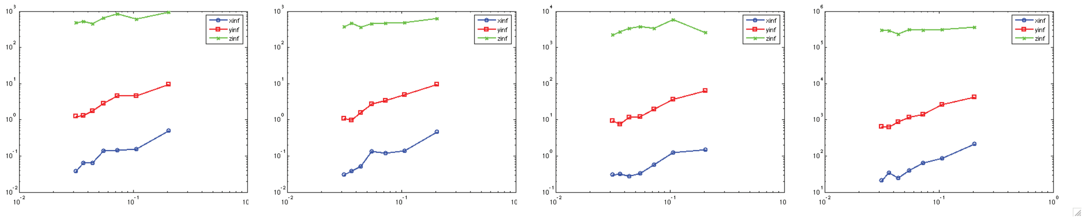

Figure 4.13: Error plots of four different functions from degree six polynomial space. Note that functions from same polynomial space result in approximately same convergence rates.

One advantage of having a convergent method is that for sufficiently fine meshes, the solution is nearly independent of mesh connectivity. As seen in Figure 4.14, our method produces almost identical results for any mesh whereas the one in [18] produces different results depending on the type of mesh used. In Figure 4.14, rows a-b show the results of the recovery of a quadratic; rows c-d show the recovery of a degree 6 spline bump, and rows e-f show the recovery of a high frequency trigonometric function; same as those explicitly stated in the beginning of this section. One can also see that magnitudes of the L_2 error under each image.

Another set of examples for which our method demonstrates a high degree of mesh independence is shown in Figure 4.15. In this set of examples, we compute a surface interpolating between two curves.

The importance of G^2 boundaries resulting from sixth-order systems as opposed to G^1 boundaries resulting from a fourth-order system has been mentioned in Section 4.6. Here we motivate this using visual examples. In Figure 4.16 we present results from prescribing boundary conditions as sampled from a torus, comparing the results of the biharmonic (first column) and the triharmonic (second column)

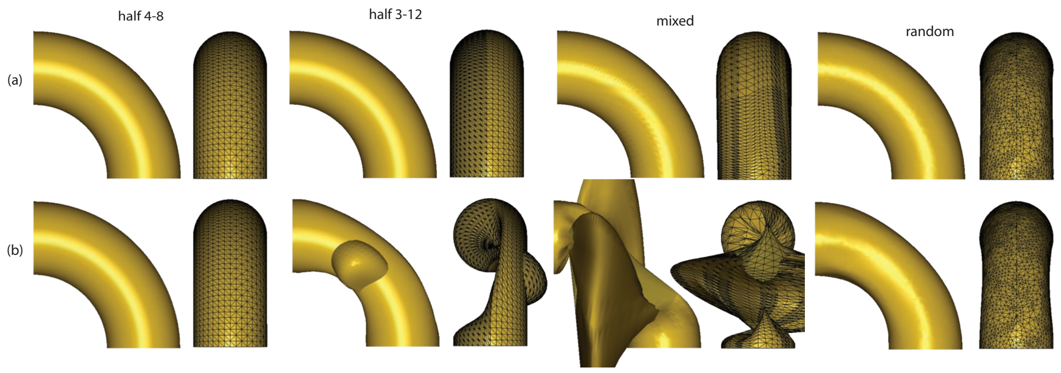

Figure 4.14: Results of curve blending tests on various meshes using our mixed FEM method and the one in [18].

equations. One can observe that for the quarter torus the difference is barely visible. However, as we increase the size of the blend surface by taking cross sections for larger sections of a torus, such as a half torus and a three quarters torus, the difference becomes clear. This is a result of the additional boundary conditions for the second normal derivatives. The existence of such conditions for the triharmonic helps the resulting surface to approximate the torus more closely, and avoid possible self intersections.

In Figure 4.17 we present a variety of shapes one can build using different boundary conditions for the first normal derivative. In the triharmonic tests second normal derivatives on the boundary are sampled from the torus. In all cases the first Neumann boundary conditions are sampled from the torus but then scaled by a scale factor k as noted on the images.

A more interesting set of examples is shown in Figure 4.18. One can see a larger variety of shapes that can be designed using different second normal boundary conditions. In all these experiments the first Neumann boundary conditions are sampled from the torus and not changed. All visible effects are due to the scale

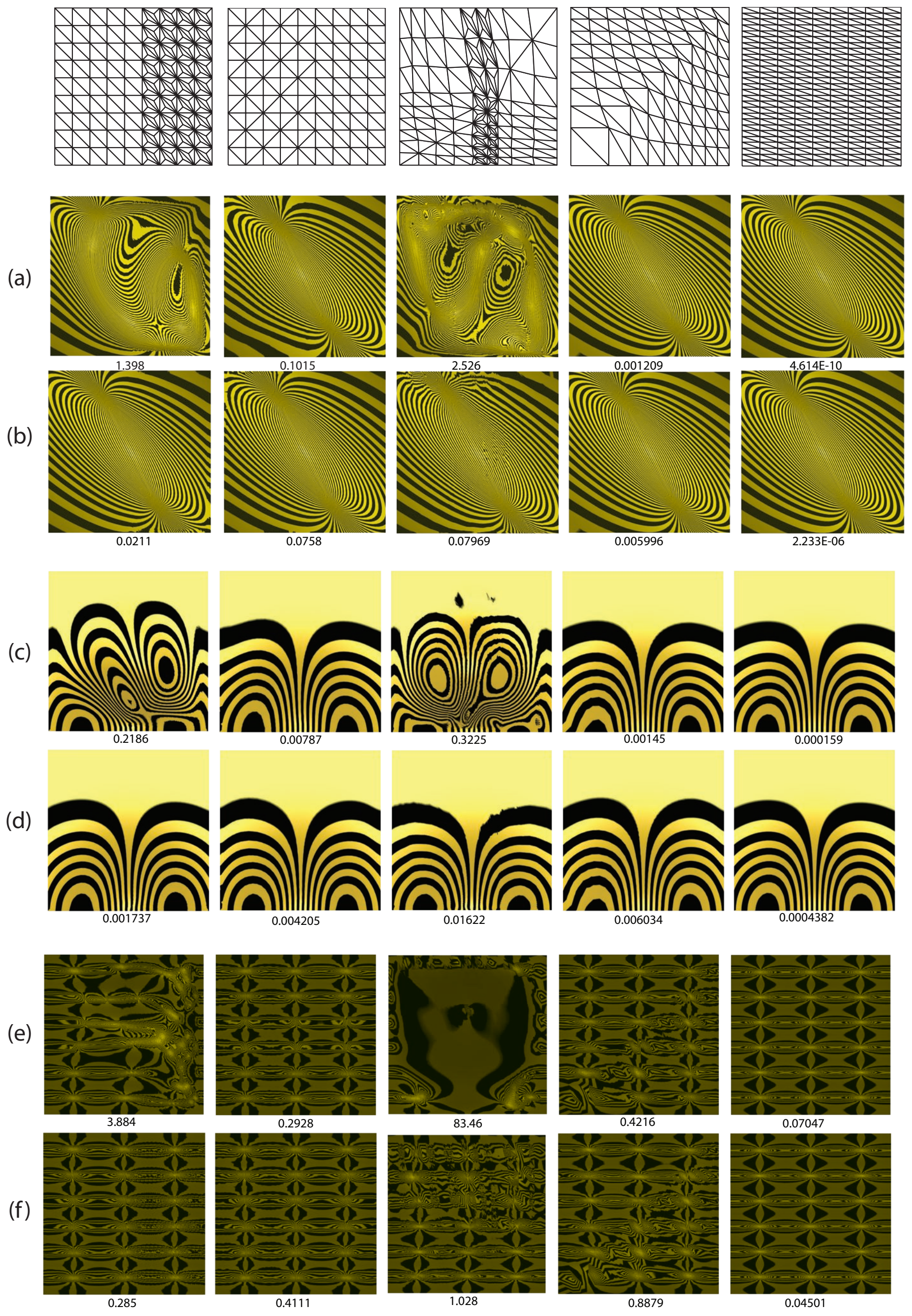

Figure 4.15: Results of some recovery tests on various meshes using our mixed FEM method ((b),(d),(f)) and the one in [18]((a),(c),(e)). L_2 error is shown below each case.

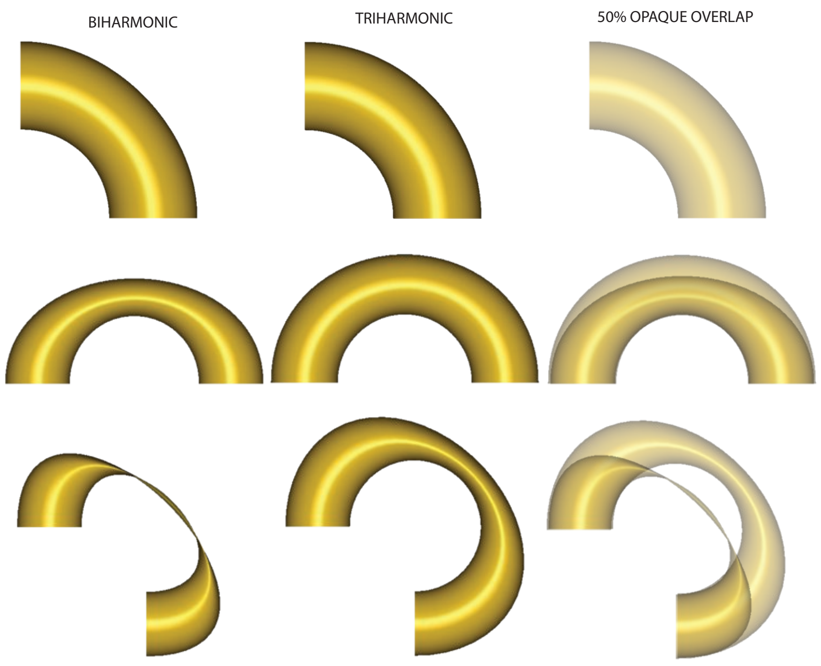

Figure 4.16: Differences between the solutions to the biharmonic and triharmonic equations when prescribed boundary conditions from different sections of a torus.

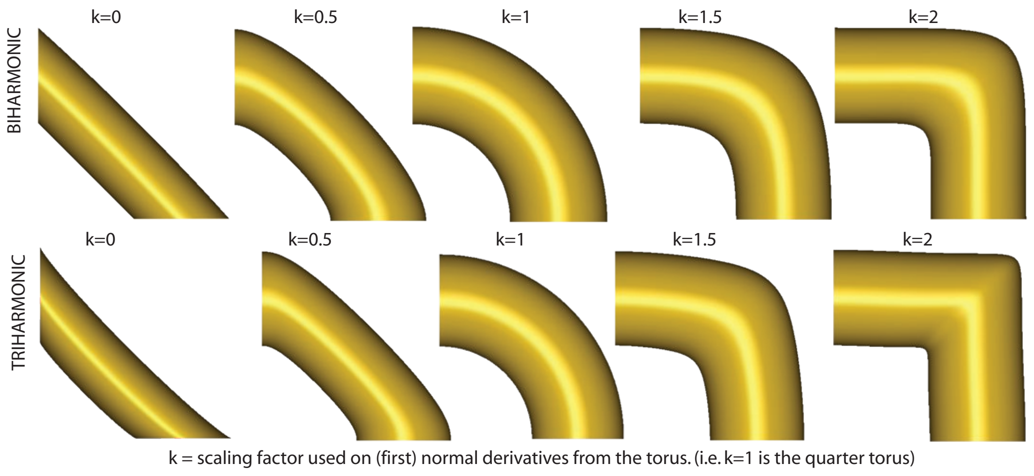
k=0
k=0.5
k=1
k=1.5
k=2
BIHARMONIC
TRIHARMONIC
k = scaling factor used on (first) normal derivatives from the torus. (i.e. k=1 is the quarter torus)

Figure 4.17: Variety of shapes achieved through different prescribed Neumann boundary conditions for triharmonic versus biharmonic.

factor on the second normal derivatives prescribed on the boundary. One can view this as an effect on thickness. This is the extra control a sixth-order formulation provides.

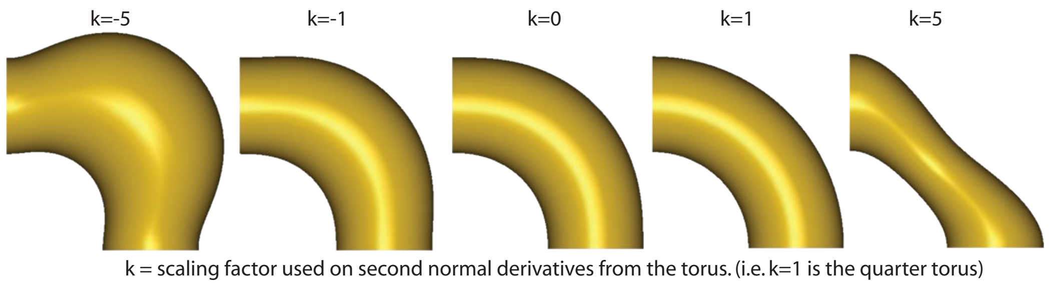
k=-5
k=-1
k=0
k=1
k=5
k = scaling factor used on second normal derivatives from the torus. (i.e. k=1 is the quarter torus)

Figure 4.18: Variety of shapes achieved through different prescribed second-order boundary conditions.

The next set of examples shows our method applied to a model hole-filling problem. A half of a sphere is removed, and a replacement surface is reconstructed from boundary data only. The difference between the triharmonic and the biharmonic solutions to the hole-filling problem are clearly visible. While there is a visible difference in the overall shape, we also show reflection lines to emphasize the local defects in the solution to the fourth-order problem. One can clearly see the kinks in reflection lines at the junction curve on the top row (biharmonic, G^1 boundaries) where no such distortion appears on the bottom row (triharmonic, G^2 boundaries)

We have experimented with a variety of blending settings. In Figure 4.20 we provide numerous examples.

In the first row of images, we show blend surfaces interpolating between a torus with square cross-section and a cylinder. The first two columns show curvature maps of the blend surfaces where the region blend surface connects with the square torus is clearly visible as a sharp change in hue. The last two columns show

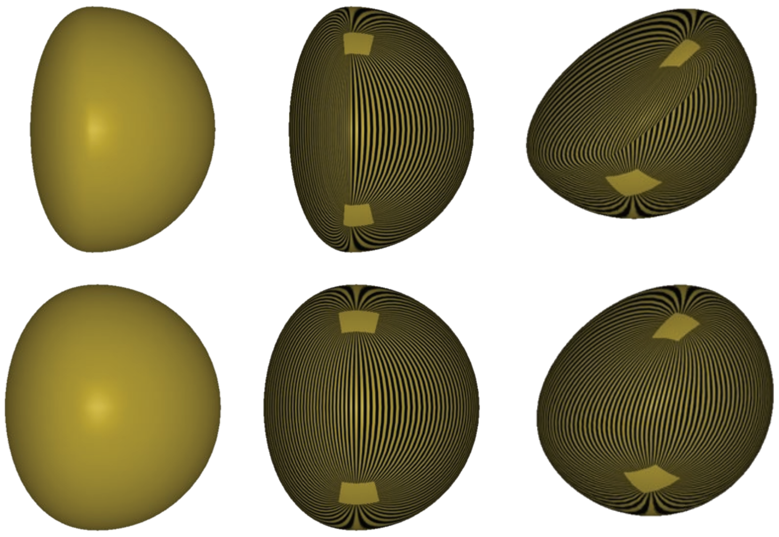

Figure 4.19: Example of hole-filling, where the hole results from removing the half sphere. Above: biharmonic, below: triharmonic.

the blend surface in red, and displays the reflection lines to highlight the local irregularities. One can see an abrupt change in the density of reflection lines going from the blend surface to the square torus.

In the second row, we compute a blend surface between a torus with the cross section of a high frequency trigonometric curve and a cylinder. There is also a phase difference that results in a twist-like effect. Once again curvature maps show the change in curvature values at the junction curve for the fourth-order solution. The reflection lines also display a similar change in density.

The third row shows blend surfaces interpolating two coaxial closed curves lying on parallel planes. One curve is a square and the other a cross section of a cylinder with larger radius. The curvature maps as well as the reflection lines show clearly the problem at the transition area in the biharmonic solution. We provide reflection lines as viewed from several other directions. The blend surface is shaded in red as before.

We present a more complicated setting in the last row, where the two coaxial curves on parallel planes are a square and a high frequency closed curve. The shape, reflection lines and the curvature maps display the differences between the fourth-order biharmonic and the sixth-order triharmonic solution.

4.9 Future Directions

We propose that a similar mixed formulation can be used to solve non-linear problems of high-order in a flow setting as well. For example, the sixth-order flow described in [190] would be a suitable candidate.

Below we sketch the starting point the mixed finite element formulation for solving another, simpler sixth-order flow: $\Delta^2 H = 0$, where H is the mean curvature.

The flow we are interested in is formulated as:

$$\frac{\partial \vec{x}}{\partial t} = -\triangle^2 H \cdot \vec{n} \tag{4.69}$$

where $\vec{x}$ is a 3-dimensional vector describing the surface over the domain Ω, H is the mean curvature and $\vec{n}$ is the normal. We also prescribe a set of boundary conditions and an initial condition.

$$\begin{aligned} \vec{x}|_{\partial\Omega} &= \vec{b_0} && (4.70)\\ \frac{\partial \vec{x}}{\partial n}|_{\partial\Omega} &= \vec{b_1} && (4.71)\\ \frac{\partial^2 \vec{x}}{\partial n^2}|_{\partial\Omega} &= \vec{b_2} && (4.72)\\ \vec{x}|_{\tau=0} &= \vec{x}^0 && (4.73) \end{aligned}$$

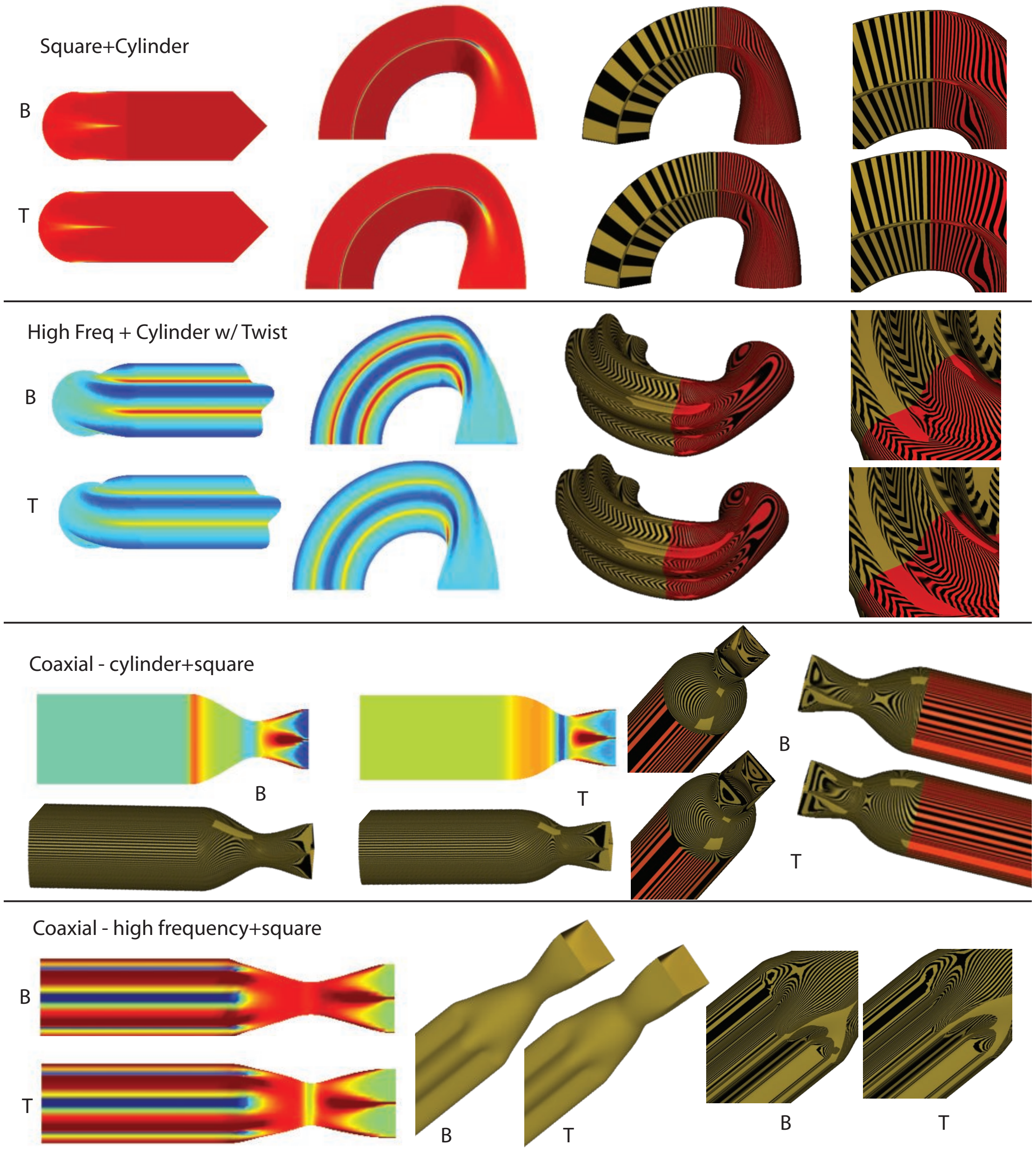
Square+Cylinder
B
T
High Freq + Cylinder w/ Twist
B
T
Coaxial - cylinder+square
B
T
B
T
Coaxial - high frequency+square
B
T
B
T
B
T

Figure 4.20: Curvature maps and reflection lines on blend surfaces between two curves as described in images. The letters B and T refer to biharmonic and triharmonic solutions respectively.

Then to solve for $\vec{x}$ using a mixed FEM formulation as described in this chapter, one can introduce a set of auxiliary variables and write the system as the following.

$$H = y = \quad \triangle \vec{x} \cdot \vec{n} \tag{4.74}$$

$$z = \quad \triangle y \tag{4.75}$$

$$\frac{\partial \vec{x}}{\partial t} = \quad -\triangle z \cdot \vec{n} \tag{4.76}$$

Similar treatment of boundary conditions should be used here as discussed in Section 4.4.3. Writing the above equations in weak form and incorporating the boundary conditions will lead to a similar linear system to the ones discussed in this chapter to be solved at each time step τ of the non-linear optimization.

4.10 Conclusions

We have presented a method to solving a high-order PDE without using high-order conforming finite elements. Using mixed finite elements, we split a high-order problem into a set of lower-order systems that can be solved my simpler elements. We use this technique to solve the fourth-order biharmonic problem and the sixth-order triharmonic problem. In both cases we reduce the system to a series of second-order problems solvable by simple, linear finite elements. Furthermore, we proposed a regularization scheme to solve the ill-conditioned systems that high-order PDEs lead to. We have presented empirical results showing the better convergence behavior of our method as compared to existing methods. We presented comparison based results for using a fourth and a sixth-order system in surface blending and hole-filling applications. We have outlined how mixed elements can be used for a nonlinear sixth-order flow.

Conclusion

Modeling of high quality surfaces is the core of geometric modeling. Such models are used in many computer-aided design and computer graphics applications. Irregular behavior of higher-order differential parameters of the surface (e.g. curvature variation) may lead to aesthetic or physical imperfections. In this work, we presented three techniques for constructing surfaces with high degree of smoothness.

In one formulation, we constructed a manifold based surface with support for piecewise smooth boundaries of same order. We use a coarse quadrilateral input mesh and bases of any order of smoothness ranging from C^2 to C^∞, in order to represent a surface of any prescribed order of smoothness. While our resulting surfaces follow closely a subdivision approximation, they also support properties such as flexibility, explicit parameterization and local control. Having observed that such methods lead to a growth of derivative magnitudes with order, we derived lower bounds.

As an alternative, we described a discrete-geometric construction for a simple and efficient method for discretizing reflection line based functionals on meshes and demonstrated how these functionals can be used in an interactive system to optimize the shape of reflective surfaces. The functional we use is fourth order and provides explicit control over the surface. We describe an efficient and accurate discretization. We implemented tools to change direction and densities of reflection lines, as well as smoothing and warping them. One can also approximate any given pattern by reflection lines on a surface using this tool.

Lastly, we presented a mixed finite element method for solving two polyharmonic systems: the biharmonic and the triharmonic. We described a method to split the high order problem into lower order problems that can be solved by simpler elements, such as the linear conforming element used in our construction. While such high order problems lead to ill-conditioned systems, we provided a regularization scheme to overcome this difficulty. We showed that our formulation exhibits convergent behavior which in turn results in mesh independent solutions.

Bibliography

[1] L. Alboul, G. Echeverria, and M. Rodrigues. Discrete curvatures and gauss maps for polyhedral surfaces. In *European Workshop on Computational Geometry(EWCG)*, pages 69–72, Eindhoven, the Netherlands, 2005.

[2] M. Amara and F. Dabaghi. An optimal C^0 finite element algorithm for the 2D biharmonic problem: theoretical analysis and numerical results. *Numer. Math.*, 90(1):19–46, 2001.

[3] R. K. E. Andersson and B. E. J. Dahlberg. Interactive techniques for visual design. In *Topics in surface modeling*, pages 95–113. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[4] D. N. Arnold. Mixed finite element methods for elliptic problems. *Comput. Methods Appl. Mech. Eng.*, 82(1-3):281–300, 1990.

[5] D. N. Arnold and F. Brezzi. Mixed and nonconforming finite element methods: implementation, postprocessing and error estimates. *RAIRO Modél. Math. Anal. Numér.*, 19(1):7–32, 1985.

[6] F. Auricchio, F. Brezzi, and C. Lovadina. *Encyclopedia of Computational Mechanics*, volume 1, chapter 9. Wiley, 2004.

[7] C. Bajaj and G. Xu. Adaptive fairing of surface meshes by geometric diffusion. In *IV '01: Proceedings of the Fifth International Conference on Information Visualisation*, page 731, Washington, DC, USA, 2001. IEEE Computer Society.

[8] E. Bänsch, P. Morin, and R. H. Nochetto. A finite element method for surface diffusion: the parametric case. *J. Comput. Phys.*, 203(1):321–343, 2005.

[9] A. Belyaev and S. Yoshizawa. On evolute cusps and skeleton bifurcations. *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 134–140, May 2001.

[10] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 113–120. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[11] M. I. G. Bloor and M. J. Wilson. Generating blend surfaces using partial differential equations. *Computer Aided Design*, 21(3):165–171, 1989.

[12] M. I. G. Bloor and M. J. Wilson. Using partial differential equations to generate free-form surfaces. *Computer Aided Design*, 22(4):202–212, 1990.

[13] M. I. G. Bloor, M. J. Wilson, and H. Hagen. The smoothing properties of variational schemes for surface design. *Computer Aided Geometric Design*, 12(4):381–394, 1995.

[14] A. I. Bobenko and P. Schröder. Discrete willmore flow. In *Eurographics Symposium on Geometry Processing*, pages 101–110, 2005.

[15] H. Bohl and U. Reif. Degenerate bézier patches with continuous curvature. *Comput. Aided Geom. Des.*, 14(8):749–761, 1997.

[16] V. Borrelli, F. Cazals, and J.-M. Morvan. On the angular defect of triangulations and the pointwise approximation of curvatures. *Comput. Aided Geom. Des.*, 20(6):319–341, 2003.

[17] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. pages 62–83. 2005.

[18] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 630–634, New York, NY, USA, 2004. ACM.

[19] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[20] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[21] D. Braess. *Finite Elements: Theory, fast solvers and applications in solid mechanics, 2nd edn*, volume 13. 2002.

[22] S.C. Brenner and C.Carstensen. *Encyclopedia of Computational Mechanics*, volume 1, chapter 4. Wiley, 2004.

[23] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991.

[24] F. Brezzi and P.-A. Raviart. Mixed finite element methods for 4th order elliptic equations. In *Topics in numerical analysis, III (Proc. Roy. Irish Acad. Conf., Trinity Coll., Dublin, 1976)*, pages 33–56. Academic Press, London, 1977.

[25] B. M. Brown, P. K. Jimack, and M. D. Mihajlović. An efficient direct solver for a class of mixed finite element problems. *Appl. Numer. Math.*, 38(1-2):1–20, 2001.

[26] O. P. Bruno and L. A. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications. *J. Comput. Phys.*, 169(1):80–110, 2001.

[27] T. Bülow. Spherical diffusion for 3d surface smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1650–1654, December 2004.

[28] Z. Caiming and C. Fuhua. Removing local irregularities of nurbs surfaces by modifying highlight lines. *Computer Aided Design*, 30(12):923–930, 1998.

[29] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 257–266, New York, NY, USA, 1991. ACM Press.

[30] C. Chen, K. Cheng, and H. M. Liao. Fairing of polygon meshes via bayesian discriminant analysis. In *WSCG: Int. Conf. in Central Europe on Comp. Graph., Vis. and Comp. Vis.*, pages 175–182, 2004.

[31] I. Choi and K. Lee. Evaluation of surfaces for automobile body styling. In *Computer Graphics International*, pages 202–, 1996.

[32] D. L. Chopp and J. A. Sethian. Motion by intrinsic Laplacian of curvature. *Interfaces Free Bound.*, 1(1):107–123, 1999.

[33] P. G. Ciarlet. *The finite element method for elliptic problems.* North-Holland Publishing Co., Amsterdam, 1978. Studies in Mathematics and its Applications, Vol. 4.

[34] P. G. Ciarlet and P.-A. Raviart. A mixed finite element method for the biharmonic equation. In *Mathematical aspects of finite elements in partial differential equations (Proc. Sympos., Math. Res. Center, Univ. Wisconsin, Madison, Wis., 1974)*, pages 125–145. Publication No. 33. Math. Res. Center, Univ. of Wisconsin-Madison, Academic Press, New York, 1974.

[35] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu. A finite element method for surface restoration with smooth boundary conditions. *Computer Aided Geometric Design*, pages 427–445, 2004.

[36] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 397–405, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[37] U. Clarenz, M. Rumpf, and A. Telea. Fairing of point based surfaces. In *Computer Graphics International*, pages 600–603, 2004.

[38] D. Cohen-Steiner and J. Morvan. Restricted delaunay triangulations and normal cycles. *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 237–246, 2003.

[39] H. S. M. Coxeter and S. L. Greitzer. *Geometry Revisited.* Math. Assoc. Amer., 1967.

[40] P. Csákany and A. M. Wallace. Computation of local differential parameters on irregular meshes. In *Proceedings of the 9th IMA Conference on the Mathematics of Surfaces*, pages 19–33, London, UK, 2000. Springer-Verlag.

[41] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *SIGGRAPH Computer Graphics*, 33(Annual Conference Series):317–324, 1999.

[42] A. Deslandes and D. L. Bonner. Reflection line control. U.S. Patent 6717579, 2004.

[43] J. C. Dill. An application of color graphics to the display of surface curvature. *SIGGRAPH Comput. Graph.*, 15(3):153–161, 1981.

[44] V. Dragnea and E. Angelopoulou. Direct shape from isophotes. In *BenCOS05*, pages 45–50, 2005.

[45] H. Du and H. Qin. Dynamic pde-based surface design using geometric and physical constraints. *Graph. Models*, 67(1):43–71, 2005.

[46] H. Du and H. Qin. Direct manipulation and interactive sculpting of pde surfaces. *Computer Graphics Forum*, 19:261–270(10), September 2000.

[47] N. Dyn, K. Hormann, S. Kim, and D. Levin. Optimizing 3d triangulations using discrete curvature analysis. In *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 135–146, Nashville, TN, USA, 2001. Vanderbilt University.

[48] G. Farin. *Curves and surfaces for CAGD: a practical guide.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[49] G. Farin and H. Hagen. A local twist estimator. In *Topics in surface modeling*, pages 79–84. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[50] G. Farin and N. Sapidis. Curvature and the fairness of curves and surfaces. *IEEE Comput. Graph. Appl.*, 9(2):52–57, 1989.

[51] R. T. Farouki. Graphical methods for surface differential geometry. In *Proceedings on Mathematics of surfaces II*, pages 363–385, New York, NY, USA, 1988. Clarendon Press.

[52] G. E. Fasshauer and L. L. Schumaker. Minimal energy surfaces using parametric splines. *Comput. Aided Geom. Des.*, 13(1):45–79, 1996.

[53] A. R. Forrest. On the rendering of surfaces. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 253–259, New York, NY, USA, 1979. ACM Press.

[54] Y. Gingold, A. Secord, J. Y. Han, E. Grinspun, and D. Zorin. A Discrete Model for Inelastic Deformation of Thin Shells. Technical report, Aug 2004.

[55] R. Glowinski and O. Pironneau. Numerical methods for the first biharmonic equation and the two-dimensional Stokes problem. *SIAM Rev.*, 21(2):167–212, 1979.

[56] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, 2004.

[57] J. Gravesen and M. Ungstrup. Constructing invariant fairness measures for surfaces. *Adv. Comput. Math.*, 17(1-2):67–88, 2002. Advances in geometrical algorithms and representations.

[58] J. A. Gregory and J. M. Hahn. A C^2 polygonal surface patch. *Comput. Aided Geom. Design*, 6(1):69–75, 1989.

[59] G. Greiner. Variational design and fairing of spline surfaces. *Comput. Graph. Forum*, 13(3):143–154, 1994.

[60] G. Greiner. Blending surfaces with minimal curvature. In *Graphics and Robotics*, pages 163–174, London, UK, 1995. Springer-Verlag.

[61] G. Greiner. Modeling of curves and surfaces based on optimization techniques. In H. Nowacki and P.D.Kaklis, editors, *Creating Fair and Shape-Preserving Curves and Surfaces*, pages 11–27. BG Teubner, 1998.

[62] G. Greiner, J. Loos, and W. Wesselink. Surface modeling with data dependent energy functionals. *Comput. Graph. Forum*, 15:175–186, 1996.

[63] C. M. Grimm. Simple manifolds for surface modeling and parameterization. In *Shape Modeling International*, 2002.

[64] C. M. Grimm and J. F. Hughes. Modeling surfaces of arbitrary topology using manifolds. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 359–368, August 1995.

[65] C. M. Grimm and J. F. Hughes. Parameterizating n-holed tori. In *The Mathematics of Surfaces IX*, 2003.

[66] E. Grinspun, Y. Gingold, J. Reisman, and D. Zorin. Computing discrete shape operators on general meshes. *Eurographics (Computer Graphics Forum)*, 25(3), 2006.

[67] X. Gu, Y. He, M. Jin, F. Luo, H. Qin, and S.-T. Yau. Manifold splines with single extraordinary point. In *SPM '07: Proceedings of the 2007 ACM*

symposium on Solid and physical modeling, pages 61–72, New York, NY, USA, 2007. ACM.

[68] X. Gu, Y. He, and H. Qin. Manifold splines. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 27–38, New York, NY, USA, 2005. ACM Press.

[69] N. Guid, C. Oblonsek, and B. Zalik. Surface interrogation methods. *Computers and Graphics*, 19(4):557–574, 1995.

[70] H. Hagen. Variational principles in curve and surface design. In *IMA Conference on the Mathematics of Surfaces*, pages 169–190, 1992.

[71] H. Hagen and S. Hahmann. Generalized focal surfaces: A new method for surface interrogation. In *IEEE Visualization*, pages 70–76, 1992.

[72] H. Hagen, S. Hahmann, and G.-P. Bonneau. Variational surface design and surface interrogation. *Comput. Graph. Forum*, 12(3):447–459, 1993.

[73] H. Hagen, S. Hahmann, and T. Schreiber. Visualization and computation of curvature behaviour of freeform curves and surfaces. *Computer-Aided Design*, 27(7):545–552, 1995.

[74] H. Hagen, S. Hahmann, T. Schreiber, Y. Nakajima, B. Wordenweber, and P. Hollemann-Grundstedt. Surface interrogation algorithms. *IEEE Computer Graphics and Applications*, 12(5):53–60, September 1992.

[75] H. Hagen and A. Nawotki. Variational design and parameter optimized surface fitting. In *Geometric Modelling*, pages 121–134, 1996.

[76] H. Hagen and H. Pottmann. Curvature continuous triangular interpolants. In *Mathematical methods in computer aided geometric design (Oslo, 1988)*, pages 373–384. Academic Press, Boston, MA, 1989.

[77] H. Hagen, T. Schreiber, and E. Gschwind. Methods for surface interrogation. In *IEEE Visualization*, pages 187–193, 1990.

[78] S. Hahmann. Visualization techniques for surface analysis. In *Data Visualization Techniques (Trends in Software, 6).* John Wiley and Son Ltd, 1999.

[79] S. Hahmann, A. Belyaev, L. Buse, G. Elber, B. Mourrain, and C. Rössl. Shape interrogation. In L. de Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, Mathematics and Visualization, chapter 1, pages 1–52. Springer, Berlin, Germany, 2008.

[80] M. A. Halstead, B. A. Barsky, S. A. Klein, and R. B. Mandell. Reconstructing curved surfaces from specular reflection patterns using spline surface fitting of normals. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 1996. ACM Press.

[81] B. Hamann. Curvature approximation for triangulated surfaces. In *Geometric modelling*, pages 139–153, London, UK, 1993. Springer-Verlag.

[82] P. C. Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Rev.*, 34(4):561–580, 1992.

[83] P. C. Hansen. Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems. *Numer. Algorithms*, 6(1-2):1–35, 1994.

[84] P. C. Hansen. *Rank-deficient and discrete ill-posed problems.* SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Numerical aspects of linear inversion.

[85] P.C. Hansen. Regularization Tools version 4.0 for Matlab 7.3. *Numer. Algorithms*, 46(2):189–194, 2007.

[86] M. Higashi, K. Kohji, and M. Hosaka. On formulation and display for visualizing features and evaluating quality of free-form surfaces. In *EUROGRAPHICS '90*, pages 299–309, 1990.

[87] K. Hildebandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *Eurographics Symposium on Geometry Processing: SGP'05*, pages 85–90, 2005.

[88] S.-Y. Hong, C.-S. Hong, H.-C. Lee, and K. Park. Discrete local fairing of b-spline surfaces. In *ICCS '01: Proceedings of the International Conference on Computational Sciences-Part I*, pages 693–697, London, UK, 2001. Springer-Verlag.

[89] B. K. P. Horn. *Robot Vision.* The MIT Press, 1986.

[90] J. Hoschek. Detecting regions with undesirable curvature. *Computer Aided Geometric Design*, 1(2):183–192, 1984.

[91] J. Hoschek. Smoothing of curves and surfaces. *Computer Aided Geometric Design*, 2(1-3):97–105, 1985.

[92] L. Hsu, R. Kusner, and J. Sullivan. Minimizing the squared mean curvature integral for surfaces in space forms. *Experiment. Math.*, 1(3):191–207, 1992.

[93] A. Hubeli and M. Gross. Fairing of non-manifolds for visualization. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 407–414, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[94] F. Jin. Directional surface fairing of elongated shapes. In H. Nowacki and P.D.Kaklis, editors, *Creating Fair and Shape-Preserving Curves and Surfaces*, pages 164–178. BG Teubner, 1998.

[95] M. Kallay. Constrained optimization in surface design. In B. Falcidieno and T.L. Kunii, editors, *Modelling in Computer Graphics*, pages 85–93. Springer, 1993.

[96] M. Kallay and B. Ravani. Optimal twist vectors as a tool for interpolating a network of curves with a minimum energy surface. *Computer Aided Geometric Design*, 7(6):465–473, 1990.

[97] S. Karbacher, J. Babst, G. Husler, and X. Laboureux. Visualization and detection of small defects on car-bodies. In B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 1999*, pages 1–8, August 1999.

[98] K. Karčiauskas and J. Peters. Polynomial C^2 spline surfaces guided by rational multisided patches. In *Computational methods for algebraic spline surfaces*, pages 119–134. Springer, Berlin, 2005.

[99] E. Kaufmann and R. Klass. Smoothing surfaces using reflection lines for families of splines. *Computer Aided Design*, 20(6):312–316, 1988.

[100] R. Klass. Correction of local surface irregularities using reflection lines. *Computer-Aided Design*, 12(2):73–77, March 1980.

[101] L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13(8):743–761, 1996.

[102] L. Kobbelt. Discrete fairing. In *7th IMA Conf. on the Mathematics of Surfaces*, pages 101–130, 1997.

[103] L. Kobbelt. Variational design with parametric meshes of arbitrary topology. In H. Nowacki and P.D.Kaklis, editors, *Creating Fair and Shape-Preserving Curves and Surfaces*, pages 189–198. BG Teubner, 1998.

[104] L. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, 16(3-4):142–158, 2000.

[105] J. J. Koenderink and A. J. van Doorn. Surface shape and curvature scales. *Image Vision Comput.*, 10(8):557–564, 1992.

[106] A. Kolb, H. Pottmann, and H. P. Seidel. Surface reconstruction based upon minimum norm networks. In M. Daehlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, pages 293–304. Vanderbilt University Press, Nashville, TN, 1995.

[107] V.I. Korobov, G.M. Sklyar, and V.V. Florinskii. A Minimal Polynomial for Finding the Switching Instants and the Support Vector of the Controllability Domain. *Differential Equations*, 38(1):15–18, 2002.

[108] S. Kubiesa, H. Ugail, and M. Wilson. Interactive design using higher order pdes. *The Visual Computer*, 20:682–693(12), December 2004.

[109] T. Langer, A. Belyaev, and H.-P. Seidel. Exact and approximate quadratures for curvature tensor estimation. In Günther Greiner, Joachim Hornegger, Heinrich Niemann, and Marc Stamminger, editors, *Vision, Modeling, and*

Visualization 2005 (VMV'05), pages 421–428, Erlangen, Germany, November 2005. Aka.

[110] A. Levin. Modified subdivision surfaces with continuous curvature. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1035–1040, New York, NY, USA, 2006. ACM.

[111] G. Liden and A. A. Ball. Intersection techniques for assessing surface quality. In *Proceedings of the 5th IMA Conference on the Mathematics of Surfaces*, pages 191–202, New York, NY, USA, 1994. Clarendon Press.

[112] T. Lilienblum, B. Michaelis, P. Albrecht, and R. Calow. Dent detection in car bodies. In *ICPR*, pages 4775–4778, 2000.

[113] X. Liu, H. Bao, Q. Peng, P.-A. Heng, and T.-T. Wong. Constrained fairing for meshes. *Comput. Graph. Forum*, 20(2):115–123, 2001.

[114] C. Loop. Second order smoothness over extraordinary vertices. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 165–174, New York, NY, USA, 2004. ACM Press.

[115] J. Loos, G. Greiner, and H.-P. Seidel. Modeling of surfaces with fair reflection line pattern. In *Shape Modeling International*, pages 256–, 1999.

[116] T. Maekawa, Y. Nishimura, and T. Sasaki. Circular highlight/reflection lines. *Computer-Aided Design & Applications*, 2(1-4):291–300, 2005.

[117] J.-L. Maltret and M. Daniel. Discrete curvatures and applications : a survey. Rapport de recherche LSIS.RR.2002.002, Laboratoire des Sciences de l'Information et des Systèmes, 2002.

[118] P. Mamassian. Isophotes on a smooth surface related to scene geometry. In B. C. Vemuri, editor, *Proc. SPIE Vol. 2031, p. 124-133, Geometric Methods in Computer Vision II, Baba C. Vemuri; Ed.*, volume 2031 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 124–133, June 1993.

[119] D. S. Meek and D. J. Walton. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. *Comput. Aided Geom. Des.*, 17(6):521–543, 2000.

[120] E. Mehlum and C. Tarrou. Invariant smoothness measures for surfaces. *Adv. Comput. Math.*, 8(1-2):49–63, 1998.

[121] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

[122] P. Monk. A mixed finite element method for the biharmonic equation. *SIAM J. Numer. Anal.*, 24(4):737–749, 1987.

[123] H. P. Moreton. Simplified curve and surface interrogation via mathematical packages and graphics libraries and hardware. *Computer-Aided Design*, 27(7):523–543, 1995.

[124] H. P. Moreton and C. H. Séquin. Surface design with minimum energy networks. In *Symposium on Solid Modeling and Applications*, pages 291–301, 1991.

[125] H. P. Moreton and C. H. Sé;quin. Functional optimization for fair surface design. In *SIGGRAPH '92: Proceedings of the 19th annual conference on*

Computer graphics and interactive techniques, pages 167–176, New York, NY, USA, 1992. ACM Press.

[126] H. P. Moreton and C. H. Séquin. Scale-invariant minimum-cost curves: Fair and robust design implements. *Comput. Graph. Forum*, 12(3):473–484, 1993.

[127] F. Munchmeyer. On surface imperfections. In *Proceedings on Mathematics of surfaces II*, pages 459–474, New York, NY, USA, 1988. Clarendon Press.

[128] J. C. Navau and N. P. Garcia. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design*, 17(7):643–671, 2000.

[129] J. C. Navau, N. P. Garcia, and M. V. Anglada. A generic approach to free form surface generation. *J. Comput. Inf. Sci. Eng.*, 2(4):294–301, 2002.

[130] A. Nawotki and H. Hagen. Physically based modeling. In H. Nowacki and P.D.Kaklis, editors, *Creating Fair and Shape-Preserving Curves and Surfaces*, pages 133–139. BG Teubner, 1998.

[131] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3):41, 2007.

[132] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *International Conference on Computer Graphics and Interactive Techniques*, pages 1142–1147, 2005.

[133] A. Neumaier. Solving ill-conditioned and singular linear systems: a tutorial on regularization. *SIAM Rev.*, 40(3):636–666 (electronic), 1998.

[134] H. Nowacki and D.Reese. Design and fairing of ship surfaces. In *Surfaces in Computer Aided Geometric Design*, pages 121–134. North-Holland Publishing Company, Amsterdam, 1983.

[135] H. Nowacki, G. Westgaard, and J. Heimann. Creation of fair surfaces based on higher order fairness measures with interpolation constraints. In H. Nowacki and P.D.Kaklis, editors, *Creating Fair and Shape-Preserving Curves and Surfaces*, pages 141–161. BG Teubner, 1998.

[136] M. Ohsaki, T. Ogawa, and R. Tateishi. Shape optimization of curves and surfaces considering fairness metrics and elastic stiffness. *Structural and Multidisciplinary Optimization*, 27(4):250–258, June 2004.

[137] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. In *GMP '00: Proceedings of the Geometric Modeling and Processing 2000*, page 229, Washington, DC, USA, 2000. IEEE Computer Society.

[138] D. L. Page, Y. Sun, A. Koschan, J. Paik, and M. Abidi. Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Journal of Graphical Models*, 64:1–31, 2003.

[139] J. Peters. Curvature continuous spline surfaces over irregular meshes. *Computer-Aided Geometric Design*, 13(2):101–131, Feb 1996.

[140] J. Peters. C^2 free-form surfaces of degree $(3, 5)$. *Comput. Aided Geom. Design*, 19(2):113–126, 2002.

[141] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experiment. Math.*, 2(1):15–36, 1993. 1058-6458.

[142] T. Poeschl. Detecting surface irregularities using isophotes. *Computer Aided Geometric Design*, 1(2):163–168, 1984.

[143] H. Pottmann. Visualizing curvature discontinuities of free-form surfaces. In *EUROGRAPHICS '89*, pages 529–536, 1989.

[144] H. Prautzsch. Freeform splines. *Comput. Aided Geom. Design*, 14(3):201–206, 1997.

[145] H. Prautzsch and G. Umlauf. A G^2-subdivision algorithm. In G. Farin, H. Bieri, G. Brunnet, and T. DeRose, editors, *Geometric Modelling, Computing Suppl. 13*, pages 217–224. Springer-Verlag, 1998.

[146] C. Radoux. Addition formulas for polynomials built on classical combinatorial sequences. *Journal of Computational and Applied Mathematics*, 115(1-2):471–477, 2000.

[147] T. Rando and J. A. Roulier. Designing faired parametric surfaces. *Computer-Aided Design*, 23(7):492–497, 1991.

[148] A. Razdan and M. Bae. Curvature estimation scheme for triangle meshes using biquadratic bezier patches. *Computer Aided Design*, 37:1481–1491, December 2005.

[149] J. Reisman, E. Grinspun, and D. Zorin. A note on the triangle-centered quadratic interpolation discretization of the shape operator. Technical report, New York University, 2007.

[150] J. Roulier and T. Rando. *Measures of Fairness for Curves and Surfaces*, pages 75–123. In Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design.

[151] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, pages 486–493, Sept 2004.

[152] R. E. Rusu. An algorithm for the elastic flow of surfaces. *Interfaces and Free Boundaries*, 7:229–239, 2005.

[153] N. S. Sapidis and G. D. Koras. Visualization of curvature plots and evaluation of fairness: an analysis of the effect of 'scaling'. *Computer Aided Geometric Design*, 14(4):299–311, 1997.

[154] S. Schaefer, J. Warren, and D. Zorin. Lofting curve networks using subdivision surfaces. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 103–114, New York, NY, USA, 2004. ACM.

[155] R. Schneider and L. Kobbelt. Mesh fairing based on an intrinsic pde approach. *Computer-Aided Design*, 33:767–777(11), 14 September 2001.

[156] R. Schneider and L. Kobbelt. Discrete fairing of curves and surfaces based on linear curvature distribution. In Pierre-Jean Laurent, Paul Sablonniere, and Larry Schumaker, editors, *Curve and Surface Design, Saint-Malo 1999*, Innovations in Applied Mathematics, pages 371–380, Saint-Malo, France, 2000. Vanderbilt University Press.

[157] R. Schneider and L. Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design*, 18(4):359–379, 2001.

[158] R. Schneider, L. Kobbelt, and H.-P. Seidel. Improved bi-laplacian mesh fairing. In Tom Lyche and Larry L. Schumaker, editors, *Mathematical Methods*

for Curves and Surfaces: Oslo 2000, Innovations in Applied Mathematics, pages 445–454, Oslo, Norway, 2001. Vanderbilt University.

[159] R. Scholz. A mixed method for 4th order problems using linear finite elements. *RAIRO Anal. Numér.*, 12(1):85–90, iii, 1978.

[160] D. Schweitzer. Artificial texturing: An aid to surface visualization. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 23–29, New York, NY, USA, 1983. ACM Press.

[161] L. R. Seidenberg, R. B. Jerard, and J. Magewick. Surface curvature analysis using color. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 260–267, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[162] C. H. Sequin. Cad tools for aesthetic engineering. *Computer Aided Design*, 37(7):737–750, June 2005.

[163] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.

[164] G. Strang and G. J. Fix. *An analysis of the finite element method.* Prentice-Hall Inc., Englewood Cliffs, N. J., 1973. Prentice-Hall Series in Automatic Computation.

[165] J. M. Sullivan. Curvature measures for discrete surfaces. Preprint, 2002.

[166] T. Surazhsky, E. Magid, O. Soldea, G. Elber, and E. Rivlin. A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *ICRA: International Conference on Robotics and Automation*, pages 1021–1026, 2003.

[167] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 125–132, Washington, DC, USA, 2002. IEEE Computer Society.

[168] G. Taubin. Curve and surface smoothing without shrinkage. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 852, Washington, DC, USA, 1995. IEEE Computer Society.

[169] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902, Washington, DC, USA, 1995. IEEE Computer Society.

[170] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1995. ACM Press.

[171] G. Taubin. Geometric signal processing on polygonal meshes. In *EUROGRAPHICS 2000 - STARs*, 2000.

[172] H. Theisel. Are isophotes and reflection lines the same? *Computer Aided Geometric Design*, 18(7):711–722, 2001.

[173] H. Theisel, C. Rössl, R. Zayer, and H.-P. Seidel. Normal based estimation of the curvature tensor for triangular meshes. In *Pacific Conference on Computer Graphics and Applications*, pages 288–297, 2004.

[174] H. Ugail, M. I. G. Bloor, and M. J. Wilson. Techniques for interactive design using the pde method. *ACM Trans. Graph.*, 18(2):195–212, 1999.

[175] H. Ugail, M.I.G. Bloor, and M.J. Wilson. Manipulation of pde surfaces using an interactively defined parameterisation. *Computers and Graphics*, 23:525–534(10), August 1999.

[176] D.-E. Ulmet. Reflection curves-new computation and rendering techniques. *International Journal of Mathematics and Mathematical Sciences*, 21:1121–1132, 2004.

[177] D.-E. Ulmet. Customized reflection lines for surface interrogation in car body design. In *SYNASC '07: Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 124–129, Washington, DC, USA, 2007. IEEE Computer Society.

[178] R.M.J. van Damme and L. Alboul. Tight triangulations. In T. Lyche M. Daehlen and L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, pages 517–526, 1995.

[179] T. I. Vassilev. Fair interpolation and approximation of b-splines by energy minimization and points insertion. *Computer-Aided Design*, 28(9):753–760, 1996.

[180] K. Watanabe and A. G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3), 2001.

[181] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. *Computer Graphics*, 28(Annual Conference Series):247–256, 1994.

[182] W. Welch and A. P. Witkin. Variational surface modeling. In *SIGGRAPH*, pages 157–166, 1992.

[183] J. W. Wesselink. *Variational Modeling of Curves and Surfaces*. PhD thesis, Technische Universiteit Eindhoven, 1996.

[184] G. Westgaard. *Construction of Fair Curves and Surfaces*. PhD thesis, Technische Universitat Berlin, Germany, 2000.

[185] G. Westgaard and H. Nowacki. Construction of fair surfaces over irregular meshes. In *Symposium on Solid Modeling and Applications*, pages 88–98, 2001.

[186] G. Xu. Consistent approximations of some geometric differential operators. Technical Report ICM-06-001, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, 2000.

[187] G. Xu. Surface fairing and featuring by mean curvature motions. *J. Comput. Appl. Math.*, 163(1):295–309, 2004.

[188] G. Xu, Q. Pan, and C. L. Bajaj. Discrete surface modeling using geometric flows. ICES Technical Report 03-38, University of Texas at Austin, 2003.

[189] G. Xu, Q. Pan, and C. L. Bajaj. Discrete surface modelling using partial differential equations. *Comput. Aided Geom. Design*, 23(2):125–145, 2006.

[190] G. Xu and Q. Zhang. G2 surface modeling using minimal mean-curvature-variation flow. *Computer-Aided Design*, 39:342–351, 2007.

[191] G. Xu and Q. Zhang. A general framework for surface modeling using geometric partial differential equations. *Comput. Aided Geom. Design*, 25(3):181–202, 2008.

[192] H. Yagou, Y. Ohtake, and A. G. Belyaev. Mesh smoothing via mean and median filtering applied to face normals. In *Geometric Modeling and Processing, Theory and Applications*, pages 124–131, 2002.

[193] Y.D. Yang and J.B. Gao. Remarks on the ciarlet-raviart mixed finite element. *Electronic Transactions on Numerical Analysis*, 4:158–164, 1996.

[194] C. Yifan, K. P. Beier, and D. Papageorgiou. Direct highlight line modification on nurbs surfaces. *Computer-Aided Geometric Design*, 14(6):583–601, 1997.

[195] L. Ying and D. Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Trans. Graph.*, 23(3):271–275, 2004.

[196] S. Yoshizawa and A. G. Belyaev. Fair triangle mesh generation with discrete elastica. In *GMP '02: Proceedings of the Geometric Modeling and Processing Theory and Applications (GMP'02)*, page 119, Washington, DC, USA, 2002. IEEE Computer Society.

[197] L. You, P. Comninos, and J. J. Zhang. Pde blending surfaces with c2 continuity. *Computers and Graphics*, 28(6):895–906, 2004.

[198] J. Yu, X. Yin, X. Gu, L. McMillan, and S. Gortler. Focal surfaces of discrete geometry. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 23–32, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[199] J. Yu, X. Yin, X. Gu, L. McMillan, and S. Gortler. Geometric modeling using focal surfaces. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, page 70, New York, NY, USA, 2007. ACM.

[200] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):644–651, August 2004.

[201] C. Zhang, P. Zhang, and F. Cheng. Fairing spline curves and surfaces by minimizing energy. *Computer-Aided Design*, 33(13):913–923, 2001.

[202] H. Zhang and E. Fiume. Mesh smoothing with shape or feature preservation. In *Advances in Modeling, Animation and Rendering*, pages 167–182. Springer, 2002.

[203] J. J. Zhang and L. You. Surface representation using second, fourth and mixed order partial differential equations. In *Shape Modeling International*, pages 250–256. IEEE Computer Society, 2001.

[204] J. J. Zhang and L. You. Rapid generation of c2 continuous blending surfaces. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part II*, pages 92–101, London, UK, 2002. Springer-Verlag.

[205] J. J. Zhang and L.H. You. Fast surface modelling using a 6th order pde. *Computer Graphics Forum*, 23:311–320(10), September 2004.

[206] Y. Zhang, C. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In *Communications in Numerical Methods in Engineering*. 2007.

[207] D. Zorin. Curvature-based energy for simulation and variational modeling. In *Int. Conf. on Shape Modeling and Applications(SMI)*, pages 198–206, 2005.

[208] D. Zorin. Constructing curvature-continuous surfaces by blending. *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 31–40, 2006.