Data-Dependent Fairing of Subdivision Surfaces

Ilja Friedel[1] Caltech

Patrick Mullen[2] Microsoft

Peter Schröder[1] Caltech

Abstract

In this paper we present a new algorithm for solving the data dependent fairing problem for subdivision surfaces, using Catmull-Clark surfaces as an example. Earlier approaches to subdivision surface fairing encountered problems with singularities in the parametrization of the surface. We address these issues through the use of the characteristic map parametrization, leading to well defined membrane and bending energies even at irregular vertices. Combining this approach with ideas from data-dependent energy operators we are able to express the associated nonlinear stiffness matrices for Catmull-Clark surfaces as linear combinations of precomputed energy matrices. This machinery also provides exact, inexpensive gradients and Hessians of the new energy operators. With these the nonlinear minimization problem can be solved in a stable and efficient way using Steihaug's Newton/CG trust-region method. We compare properties of linear and nonlinear methods through a number of examples and report on the performance of the algorithm.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations. G.1.6 [Numerical Analysis]: Optimization—Unconstrained optimization

Keywords: Geometric Modeling, CAD, Fairing, Bicubic B-Splines, Subdivision Surfaces, Catmull-Clark, Thinplate Energy, Nonlinear Minimization

1 Introduction

The construction of fair surfaces is important in many areas of geometric modeling and has a long tradition in areas such as ship-hull, airplane and automotive design. The goals of fairing typically fall into one of three categories: denoising or removal of spurious fine scale features; ab initio definition of a shape; and physical simulation of shells, *i.e.*, materials from which a given shape is to be manufactured. Typically fairing is achieved by defining a functional of the surface and then seeking a minimum of this functional subject to appropriate constraints. In some cases the functional arises naturally from material properties. More difficult is the definition of functionals which are meant to capture æsthetic notions of fairness. In practice the fairness functional is most often expressed in terms of integrals of surface properties. Examples include derivative and curvature information, though higher order invariants have also been used. In the present paper we will restrict ourselves to membrane and bending energies of the surface as these are central to the most often employed approaches.

Previous fairing methods can be classified by surface primitive (*e.g.*, spline patches, piecewise linear meshes, etc.), fairness functional (*e.g.*, membrane, bending, variation of curvature, etc.), and numerical approach, *i.e.*, the use of possibly simplified energies and the particular minimization scheme employed.

A popular approach [6, 10, 11] for the fairing of spline surfaces is based on computing the fairness integral using numerical quadrature. Splines such as bicubic patches are very smooth (C^2; in fact C^∞ in the interior of each integration domain) and the curvatures are well behaved. Hence high order integration rules such as Gauss [6, 10] and Lobatto [11] are well suited to compute good approximations of the integrals.

The resulting energy minimization problem is solved in different manners. Moreton and Sequin [11] compute a "good" initial approximation and improve it with an energy gradient descent scheme. The gradients of the fairing functional are computed using finite differences. This is both expensive and can lead to numerical difficulties. To avoid these difficulties we compute gradients and Hessians of our energies directly. Lott and Pullin [10] used a more sophisticated simplex method for the minimization of their functional. Greiner and co-workers [6] linearized their functional and numerically integrated the associated "stiffness" matrix entries. The solution of the resulting linear system of equations minimizes the linearized energy. This process can be iterated by solving a sequence of such linearized problems. However, such a sequence does not in general approach a global or even local minimum.

Most of the earlier work using parametrized smooth surfaces employed spline patches as the modeling primitive of choice. Subdivision surfaces are closely related in that they generalize B-splines to arbitrary topology control meshes. For example, the scheme of Catmull and Clark [2] generalizes bicubic splines. There are many other subdivision schemes (for a tutorial see [25]), but Catmull-Clark surfaces remain the most often employed in applications ranging from industrial design to computer games and animation (see for example the movies of Pixar). They are also available in the MPEG4 standard and are increasingly deployed in free-form surface design packages. Examples include Maya (Alias|wavefront), 3D Studio Max (Discreet), Softimage (Avid), and Catia (Dassault).

A possible approach to the fairing of subdivision surfaces could be devised by applying methods originally designed for mesh smoothing [9, 17, 19, 21] to the control polyhedron. However, minimizing some discrete energy of the control mesh does not necessarily lead to a minimization of the energy of the associated smooth surface. Achieving the latter requires computation of energies of the actual surface. Note that the *degrees of freedom* continue to be the control points, just as in the usual spline setting.

A special difficulty in the treatment of subdivision surfaces arises at irregular vertices. In the case of Catmull-Clark surfaces these are vertices of the control mesh where other than four patches meet. While the surface is globally C^2 it is only C^1 continuous at irregular vertices. This immediately raises the question if the corresponding squared curvature integrals exist at all. Reif and Schröder [16] showed that they do exist for a wide class of subdivision schemes including the scheme of Catmull and Clark. Nevertheless the numerical computation of these integrals is difficult since the curvatures generally diverge at the irregular vertices. Bakhvalov's theorem [1] connects the smoothness of integrands (here at most C^0)

1. {ilja,ps}@cs.caltech.edu
2. patmul@microsoft.com

The research of the second author was performed during his studies at Caltech.

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

SM'03, June 1620, 2003, Seattle, Washington, USA.
Copyright 2003 ACM 1-58113-706-0/03/0006...$5.00.

with the dimension of the integration problem (here $d = 2$) to estimate the best convergence rate for the worst case (C^0) integrand. The theorem states that the convergence rate of the integration error ϵ_n scales as $O(n^{-1/2})$, where n is the number of integrand samples. This result might be too pessimistic, but it reflects our experience that curvature integrals converge very slowly for patches incident to irregular vertices. This difficulty makes energy gradient computations, via recourse to finite differences, very inexact. We circumvent this difficulty by using exact expressions for the gradient and Hessian of the underlying energies. This observation might explain why not much literature is available on the fairing of subdivision surfaces [7, 12] and why the approaches taken differ from many standard spline methods.

Our treatment will use ideas first presented by Halstead and coworkers [7], who precomputed linearized energies. They used a standard spline parametrization which unfortunately leads to divergent bending energies. We remedy this problem through the use of the *characteristic map* [14, 24] parametrization. Exploiting the scaling relation of the subdivision operator eigen functions, all integrals can be computed with the same high accuracy as those arising in standard bicubic spline settings. This approach is still linear and does not depend on the input data. For data-dependent fairing functionals we follow the ideas of Greiner and co-workers [6]. To increase the efficiency of the resulting nonlinear minimization problems we decompose the energies into linear combinations of precomputed stiffness matrices. The weights of these linear combinations can be computed as straightforward functions of the current state of the control mesh. Since we have access to exact gradients and Hessians of our energies as well, we can employ very robust nonlinear minimization algorithms such as Steihaug's Newton/CG trust-region method [13].

Overview In the following sections we begin by fixing our notation and giving some of the basic identities (Section 2) before employing them to derive well defined stiffness matrices in Section 3. The resulting "simple" energies are independent of the input data, a serious limitation in applications. Data-dependent stiffness matrices are discussed next (Section 4) and we show how these can be decomposed into linear combinations of precomputed stiffness matrices. As a final step in the construction of all necessary components for the energy minimizer we derive exact derivatives of the data-dependent energies (Section 5) obviating the need for costly and inaccurate finite difference computations. Algorithms for energy minimization are discussed in Section 6 and fairing results presented in Section 7 before we conclude.

2 Surface Properties

In this section we establish the basic identities we need for the description of membrane and bending energies, show how stiffness matrices arise for surface given as linear combinations of basis functions, and state some useful facts about subdivision surfaces.

A subdivision surface is parametrized over its *control mesh*, a polyhedral manifold of two-dimensional, not necessarily planar faces embedded in $\mathbb{R}^3$. Each face can be parametrized over $\Omega \subset \mathbb{R}^2$. For simplicity we assume closed surfaces only, though the machinery we develop is not limited to these (some of our examples show surfaces with boundaries). To accommodate input polyhedra of arbitrary face and vertex valencies we assume two initial subdivision steps so that the surface can be treated as the union of patches parametrized over quadrilateral faces with at most one irregular vertex incident on each face. We denote the valence of a patch to be the valence of the irregular vertex or four otherwise. With this we may assume a local parametrization of a *surface patch* $S : \Omega = [0,1]^2 \to \mathbb{R}^3$

$$S(\bar{u}) = S(u_1,u_2) = (S_1(u_1,u_2), S_2(u_1,u_2), S_3(u_1,u_2))^T.$$

In most of what follows S will refer to a single Catmull-Clark subdivision surface patch of valence k. The symbol R is also used to denote a surface. In general it refers to a simpler surface near S.

The first fundamental form of the surface $R(\bar{u})$ is the 2×2 matrix $\mathbf{I}_R(\bar{u}) = \big(g_{ij}(\bar{u})\big)$ with

$$g_{ij}(u_1,u_2) = \langle \partial_i R(u_1,u_2),\, \partial_j R(u_1,u_2)\rangle,$$

and inverse $\mathbf{I}_R^{-1} = (g^{ij})$. The second fundamental form is defined as a 2×2 matrix $\mathbf{II}_R = (h_{ij})$ with entries

$$h_{ij} = \langle \partial_i\partial_j R, \nu_R\rangle,$$

where $\nu_R = \frac{\partial_1 R\times\partial_2 R}{\|\partial_1 R\times\partial_2 R\|}$ is the normal to the surface $R(u_1,u_2)$. Finally the Christoffel symbols of the surface R at parameter value $\bar{u}$ are defined as $\Gamma^k_{ij} = g^{kl}\langle\partial_i\partial_j R, \partial_l R\rangle$. Here and in some of the following expressions matching upper and lower indices which are not otherwise bound follow the Einstein summation convention. We often express the Christoffel symbols as two 2×2 matrices $\Gamma^1 = (\Gamma^1_{ij})$ and $\Gamma^2 = (\Gamma^2_{ij})$.

The gradient of a scalar function $f(\bar{u}) : \Omega \to \mathbb{R}$, which we take to be defined on the reference surface $R(\bar{u})$, is computed as

$$\nabla_R(f) = g^{jk}\partial_k f\,\partial_j R,$$

in matrix notation $\nabla_R(f) = (\partial_1 f, \partial_2 f)\cdot \mathbf{I}_R^{-1}\cdot(\partial_1 R, \partial_2 R)^T$ (we consider gradients to be row-vectors).

The Hessian of f with respect to the reference surface R is given as

$$\mathbf{H}_R(f) = \begin{pmatrix} g^{1l}(\partial_1\partial_l f - \partial_i f\Gamma^i_{1l}) & g^{1l}(\partial_2\partial_l f - \partial_i f\Gamma^i_{2l}) \\ g^{2l}(\partial_1\partial_l f - \partial_i f\Gamma^i_{1l}) & g^{2l}(\partial_2\partial_l h - \partial_i h\Gamma^i_{2l}) \end{pmatrix}$$

and in matrix notation $\mathbf{H}_R(f) = \mathbf{I}_R^{-1}\cdot\big(\mathbf{H}(f) - \big((\partial_1 f)\Gamma^1 + (\partial_2 f)\Gamma^2\big)\big)$.

All matrices in the last equation are symmetric. However the product of two symmetric matrices needs not be symmetric. While $\mathbf{H}_S(S_i)$ is symmetric[1], $\mathbf{H}_R(f)$ is in general not symmetric for an arbitrary reference surface R and function f.[2]

The derivatives of a parametrized surface are in general not very good indicators for the behavior of the surface. The main reason is the dependence of the derivatives on the chosen parametrization. Better suited are quantities that are independent of the parametrization, such as the *principal curvatures* κ_1, κ_2, the *mean curvature* $\frac{1}{2}(\kappa_1+\kappa_2)$ or the *Gaussian curvature* $\kappa_1\,\kappa_2$. The principal curvatures of $S(\bar{u})$ are the eigen values of the matrix $\mathbf{I}_S^{-1}\cdot\mathbf{II}_S$

Our main interest is focused on the functional $\int_\Omega \kappa_1^2 + \kappa_2^2\,d\omega$. The integrand can be expressed as [6]

$$\sum_{i=1}^{3}\mathbf{tr}\big(\mathbf{H}_S(S_i)\cdot\mathbf{H}_S(S_i)^T\big) = \kappa_1^2+\kappa_2^2. \qquad (1)$$

The data-dependent *membrane* or *stretching energy* of a surface S with respect to the reference surface R is defined as

$$E_R^{\mathbf{m}}(S) = \sum_{i=1}^{3}\int_\Omega \nabla_R(S_i)\cdot\nabla_R(S_i)^T\,d\omega_R. \qquad (2)$$

The data-dependent *bending energy*[3] of a patch S with respect to R is defined as

$$E_R^{\mathbf{b}}(S) = \sum_{i=1}^{3}\int_\Omega \mathbf{tr}(\mathbf{H}_R(S_i)\cdot\mathbf{H}_R(S_i)^T)\,d\omega_R. \qquad (3)$$

[1]The *Weingarten map* $\mathbf{I}_S^{-1}\cdot\mathbf{II}_S$ is self-adjoint [20, p.58].

[2]Example: $R(u_1,u_2) = (u_1, 2u_2, 0)$ and $f(u_1,u_2) = u_1\,u_2$.

[3]Our definition is slightly changed from [6] due to the possible asymmetry of $\mathbf{H}_R(f)$. This change is necessary to guarantee the positive semi-definiteness of J_S [6, Theorem 1(b)].

Note that both $E_R^{\mathbf{m}}(S)$ and $E_R^{\mathbf{b}}(S)$ are invariant under Euclidian motions since they only involve derivative quantities.

For $R(\bar{u}) = S(\bar{u})$, $\bar{u} \in \Omega$ Equation (1) implies

$$E_S^{\mathbf{b}}(S) = \int_\Omega \kappa_1^2 + \kappa_2^2 \; d\omega_S,$$

hence the energy functional $E_S^{\mathbf{b}}(S)$ is independent of the parametrization. Greiner and co-workers [6] argued that $E_R^{\mathbf{b}}(S) \approx E_S^{\mathbf{b}}(S)$ if the fundamental forms of S and R are approximately equal for $\bar{u} \in \Omega$. This observation is a major motivation for the work presented here.

2.1 The Stiffness Matrix K

So far we have only considered surfaces S and their energies with respect to some reference surface R generically. We now consider the case of surfaces which may be regarded as linear combinations of basis functions. In that case the energy integrals can be written as a bilinear function of the control points. The *stiffness* matrix collects the associated integrals of basis functions over a given patch.

From now on we assume that the surface S is a linear combination of finitely many basis functions

$$S(u_1, u_2) = \sum_{i=1}^{M} P_i \, N_i(u_1, u_2)$$

where the $P_i \in \mathbb{R}^3$ are the control points. Combining this representation of S with Equations (2) resp. (3) we note that $\mathbf{tr}$, ∇ and $\mathbf{H}$ are linear with respect to the control points P_i. Consequently

$$E_R(S) = P^T \cdot K \cdot P = \sum_{i,j=1}^{M} K_{ij} \cdot P_i^T \cdot P_j$$

where K is an $M \times M$ matrix with entries defined as

$$K_{ij}^{\mathbf{m}} = \int_\Omega \nabla_R(N_i) \cdot \nabla_R(N_j)^T \, d\omega_R,$$
$$K_{ij}^{\mathbf{b}} = \int_\Omega \mathbf{tr}(\mathbf{H}_R(N_i) \cdot \mathbf{H}_R(N_j)^T) \, d\omega_R,$$

respectively.

The concept of stiffness matrices partially separates the process of computing the energies of a given surface from the computation of the integrals in Equations (2) and (3). We will also see later that stiffness matrices carry important information regarding the derivatives of the data-dependent energies.

2.2 Catmull-Clark Subdivision Surfaces

In this paper we deal exclusively with Catmull-Clark subdivision surfaces. Nevertheless the ideas expressed here carry over to other subdivision surfaces as long as the derivative integrals over regular regions exist. This is, for example, the case for Doo-Sabin subdivision surfaces [4], which in general are only C^1. Stam [18] showed that one can represent a Catmull-Clark subdivision patch with a single irregular vertex of valence k (Figure 1) as an expansion in $M = 2k + 8$ functions

$$S(u_1, u_2) = \sum_{i=1}^{M} C_i \, \phi_i(u_1, u_2). \qquad (4)$$

The $\phi_i : \mathbb{R}^2 \to \mathbb{R}$ are the *eigen basis functions* of the local subdivision matrix and the C_i the control points P_i projected into the corresponding eigen space. For the case $k = 4$, *i.e.*, a regular bicubic patch, the eigen functions are simply the monomials $u_1^i u_2^j$,

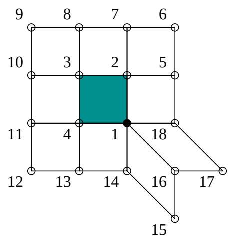


Figure 1: *Topological neighborhood of an irregular patch of valence* 5 *indicating all the degrees of freedom for this patch.*

$i, j = 0, \ldots, 3$. For general k Stam showed how to evaluate ϕ_i and its derivatives exactly at arbitrary parameter values (excepting the origin where certain derivatives do not exist). We will need his evaluation technique later for the numerical computation of all integrals.

Stam's technique is based on a remarkable property of the eigen basis functions called the *scaling relation*

$$\phi_i\left(\frac{1}{2}x\right) = \lambda_i \, \phi_i(x),$$

which we will also exploit. The $\lambda_i \in \mathbb{R}$ are the eigen values of the local subdivision operator. We assume sorted eigen values $\lambda_i \geq \lambda_{i+1}$. Catmull-Clark as well as most other subdivision schemes of practical interest have a non-degenerate set of eigen vectors with real eigen values $\lambda_1 = 1$ and $1 > \lambda := \lambda_2 = \lambda_3 > \lambda_4 =: \mu$ for all valences [25].

3 Simple Energies

So far we have not yet specified the reference surface R. To avoid diverging membrane and bending energy integrals the reference surface must be chosen carefully. The canonical choice is given by the *characteristic map* $\mathfrak{Ch} : \mathbb{R}^2 \to \mathbb{R}^2$ of a subdivision surface near an irregular vertex. In the smoothness analysis of subdivision surfaces near irregular vertices this map arises naturally as the one in which the parametrization of the subdivision surface is C^1 at the irregular vertex. The characteristic map is defined as

$$\mathfrak{Ch}(u_1, u_2) := \begin{bmatrix} \phi_2(u_1, u_2) \\ \phi_3(u_1, u_2) \end{bmatrix},$$

where the ϕ_i are the eigen functions in Equation (4), corresponding to the *sub-dominant* eigen value λ. For smooth subdivision schemes $\mathfrak{Ch}$ is known to be regular and injective [15, 23]. Note that $\mathfrak{Ch}$ does not depend on the input data, only on the subdivision scheme and the valencies.

A Catmull-Clark subdivision surface is only C^1 at irregular vertices. In most cases the second derivatives of the eigen functions ϕ_i do not exist at $\bar{u} = (0, 0)$. To simplify the evaluation of integrals over $[0, 1]^2$ and to make the integrands more numerically well behaved we rewrite it as follows. Let $L := [0, 1]^2 \setminus [0, \frac{1}{2})^2 \subset \mathbb{R}^2$ then

$$[0, 1]^2 = \bigcup_{i=0}^{\infty} 2^{-i} L.$$

Letting the image of L under the characteristic map be $L^{\mathfrak{Ch}} := \mathfrak{Ch}(L)$ we also have

$$\mathfrak{Ch}([0, 1]^2) = \bigcup_{i=0}^{\infty} \lambda^i \, L^{\mathfrak{Ch}}.$$

Before employing this decomposition of the integration domain we note that the partial derivatives of the eigen functions with respect to the characteristic map $\phi_i^{\mathfrak{Ch}} := \phi_i \circ \mathfrak{Ch}^{-1}$ satisfy the following scaling relationship

$$\begin{aligned}
\partial_k \phi_i^{\mathfrak{Ch}}(\lambda^m \bar{v}) &= \partial_k(\phi_i \circ \mathfrak{Ch}^{-1})(\lambda^m \bar{v}) \\
&= (\lambda_i/\lambda)^m \, \partial_k(\phi_i \circ \mathfrak{Ch}^{-1})(\bar{v}) \\
\partial_k \partial_l \phi_i^{\mathfrak{Ch}}(\lambda^m \bar{v}) &= \partial_k \partial_l(\phi_i \circ \mathfrak{Ch}^{-1})(\lambda^m \bar{v}) \\
&= (\lambda_i/\lambda^2)^m \, \partial_k \partial_l(\phi_i \circ \mathfrak{Ch}^{-1})(\bar{v}).
\end{aligned}$$

3.1 Simple Stiffness Matrices

In this section we show how the scaling relation observed by $\partial_k \phi_i^{\mathfrak{Ch}}$ and $\partial_k \partial_l \phi_i^{\mathfrak{Ch}}$ can be used to reduce the evaluation of the stiffness matrix entries to an integral over a single instance of $L^{\mathfrak{Ch}}$. Since the integrals are still independent of the input control points we denote the resulting stiffness matrices “simple.”

For example, to compute the membrane stiffness matrix $K^{\mathbf{m}}_{\mathfrak{Ch}}$ one has to compute integrals of the form

$$K^{\mathbf{m}}_{ij} = \int_{\mathfrak{Ch}([0,1]^2)} \partial_1 \phi_i^{\mathfrak{Ch}}(\bar{v}) \partial_1 \phi_j^{\mathfrak{Ch}}(\bar{v}) + \partial_2 \phi_i^{\mathfrak{Ch}}(\bar{v}) \partial_2 \phi_j^{\mathfrak{Ch}}(\bar{v}) \, d\bar{v}.$$

Since $\phi_1^{\mathfrak{Ch}}(\bar{v}) = const$, $K_{ij} = 0$, $i = 1$ or $j = 1$. For $i, j > 1$

$$\begin{aligned}
&\int_{\mathfrak{Ch}([0,1]^2)} \partial_k \phi_i^{\mathfrak{Ch}}(\bar{v}) \, \partial_l \phi_j^{\mathfrak{Ch}}(\bar{v}) \, d\bar{v} \\
&= \sum_{m=0}^{\infty} \int_{\lambda^m L^{\mathfrak{Ch}}} \partial_k \phi_i^{\mathfrak{Ch}}(\bar{v}) \, \partial_l \phi_j^{\mathfrak{Ch}}(\bar{v}) \, d\bar{v} \\
&= \sum_{m=0}^{\infty} \int_{L^{\mathfrak{Ch}}} (\lambda_i/\lambda)^m \, \partial_k \phi_i^{\mathfrak{Ch}}(\bar{v}) \, (\lambda_j/\lambda)^m \, \partial_l \phi_j^{\mathfrak{Ch}}(\bar{v}) \, \lambda^m \lambda^m \, d\bar{v} \\
&= (1 - \lambda_i \lambda_j)^{-1} \int_{L^{\mathfrak{Ch}}} \partial_k \phi_i^{\mathfrak{Ch}}(\bar{v}) \, \partial_l \phi_j^{\mathfrak{Ch}}(\bar{v}) \, d\bar{v}
\end{aligned}$$

With the same transformation one can show that the bending integrals for $i, j > 3$ simplify to sums of integrals of the form

$$\begin{aligned}
&\int_{\mathfrak{Ch}([0,1]^2)} \partial_k \partial_l \phi_i^{\mathfrak{Ch}}(\bar{v}) \, \partial_m \partial_n \phi_j^{\mathfrak{Ch}}(\bar{v}) \; d\bar{v} = \\
&\quad \left(1 - \frac{\lambda_i \lambda_j}{\lambda^2}\right)^{-1} \int_{L^{\mathfrak{Ch}}} \partial_k \partial_l \phi_i^{\mathfrak{Ch}}(\bar{v}) \, \partial_m \partial_n \phi_j^{\mathfrak{Ch}}(\bar{v}) \; d\bar{v}. \quad (5)
\end{aligned}$$

Since $\phi_2^{\mathfrak{Ch}}(\bar{v})$ and $\phi_3^{\mathfrak{Ch}}(\bar{v})$ are linear their second partial derivatives vanish with the result that the associated stiffness matrix entries for $i \leq 3$ or $j \leq 3$ vanish.

We have now managed to reduce the computation of the membrane and bending energy of a subdivision surface to the computation of integrals of derivatives of eigen functions over $L^{\mathfrak{Ch}}$. In a final step we will transform these integrals into integrals over L. The detailed steps to do so are exceedingly tedious and in the following Section we will only demonstrate the transformation for the case of membrane energies. The case of bending energies follows in the same footsteps, but results in rather long expressions. A demonstration of the steps can be found in Appendix A while complete, Maple derived source code is available from the authors[4].

3.2 Transformation of Membrane Integrals

The general form of the integrals we need to evaluate is given by

$$\int_{L^{\mathfrak{Ch}}} D\big(\phi_i^{\mathfrak{Ch}}\big)(\bar{v}) \cdot D\big(\phi_j^{\mathfrak{Ch}}\big)(\bar{v})^T \, d\bar{v}$$

where D denotes a matrix of partial derivative operators. Since $\phi_i^{\mathfrak{Ch}}(\bar{v}) = \phi_i \circ \mathfrak{Ch}^{-1}(\bar{v})$, $\phi_i^{\mathfrak{Ch}} : L^{\mathfrak{Ch}} \to \mathbb{R}$ we appear to be required to compute the inverse of the characteristic map, a highly non-trivial task. However this can be avoided for both membrane and bending integrals.

Since $D(\phi_i^{\mathfrak{Ch}})(\bar{v}) = D(\phi_i)(\mathfrak{Ch}^{-1}(\bar{v})) \cdot D(\mathfrak{Ch}^{-1})(\bar{v})$ we get

$$\begin{aligned}
&\int_{L^{\mathfrak{Ch}}} D\big(\phi_i\big)\big(\mathfrak{Ch}^{-1}(\bar{v})\big) \cdot D(\mathfrak{Ch}^{-1})(\bar{v}) \cdot \\
&\qquad \Big(D\big(\phi_j\big)\big(\mathfrak{Ch}^{-1}(\bar{v})\big) \cdot D(\mathfrak{Ch}^{-1})(\bar{v})\Big)^T \, d\bar{v}.
\end{aligned} \quad (6)$$

Performing the change of variables $\bar{v} = \mathfrak{Ch}(\bar{u})$ this in turn becomes

$$\begin{aligned}
&\int_L D\big(\phi_i\big)\Big(\mathfrak{Ch}^{-1}\big(\mathfrak{Ch}(\bar{u})\big)\Big) \cdot D(\mathfrak{Ch}^{-1})\big(\mathfrak{Ch}(\bar{u})\big) \cdot \\
&\Big(D\big(\phi_j\big)\Big(\mathfrak{Ch}^{-1}\big(\mathfrak{Ch}(\bar{u})\big)\Big) \cdot D(\mathfrak{Ch}^{-1})\big(\mathfrak{Ch}(\bar{u})\big)\Big)^T \, |J_{\mathfrak{Ch}}(\bar{u})| \, d\bar{u}.
\end{aligned}$$

Finally, using the identity $D(\mathfrak{Ch}^{-1})\big(\mathfrak{Ch}(\bar{u})\big) = \big(D(\mathfrak{Ch})(\bar{u})\big)^{-1}$ we arrive at a formula that does not involve $\mathfrak{Ch}^{-1}$

$$\begin{aligned}
&\int_L D(\phi_i)(\bar{u}) \cdot \big(D(\mathfrak{Ch})(\bar{u})\big)^{-1} \cdot \\
&\quad \Big(D(\phi_j)(\bar{u}) \cdot \big(D(\mathfrak{Ch})(\bar{u})\big)^{-1}\Big)^T \, \big|\mathbf{det}\big(D\big(\mathfrak{Ch}\big)(\bar{u})\big)\big| \, d\bar{u}.
\end{aligned}$$

While the operator notation hides the large number of terms involved, we note that the details of all the terms can be managed with a symbolic algebra system such as Maple. Automatically generated code can then be linked against numerical quadrature functions to compute all stiffness matrix entries once and for all in an offline process. We employed this process for our implementation.

4 Data-Dependent Energies

So far we have expressed energies with only the canonical reference surface induced by the characteristic map of a given valence. We now turn to data-dependent energies that take the input control points into account.

To illustrate that such data-dependence is required, consider the bending energy of a sphere $\mathbf{S}(r)$ with radius r

$$\int_S \kappa_1^2 + \kappa_2^2 \, d\omega_S = \int_S r^{-2} + r^{-2} \, d\omega_S = 2r^{-2} \, 4\pi r^2 = 8\pi.$$

The simple energy $E^{\mathbf{b}}_{\mathfrak{Ch}}(\mathbf{S}(r))$ computed with the data independent characteristic map parametrization of the previous section is proportional to r^2. This undesirable behavior was already observed by Wesselink [22] and is clearly wrong. For global uniform scaling it can be taken into account by a scalar factor. But if the scaling is non-uniform in different coordinate directions or patches vary significantly in size, the estimate $E^{\mathbf{b}}_{\mathfrak{Ch}}(S)$ can be made locally arbitrarily bad compared to $E^{\mathbf{b}}_S(S)$.

This observation motivates the scaling of the reference surface patch according to the dimensions of the original surface patch. This makes the energy $E_R(S)$ less dependent on the parametrization of S as already observed by Greiner *et al.* [6].

Let $\mathfrak{SCh}(\bar{u})$ be a linearly transformed version of the characteristic map $\mathfrak{Ch}(\bar{u})$. Since the energies are invariant under Euclidian motions we may assume that $\mathfrak{SCh}(\bar{u})$ is at the origin and parallel to the x/y-plane, leaving only 3 parameters to describe it

$$\mathfrak{SCh}(\bar{u}) = W \cdot \mathfrak{Ch}(\bar{u}) = \begin{bmatrix} s_x & s_{xy} \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} \phi_2(\bar{u}) \\ \phi_3(\bar{u}) \end{bmatrix},$$

with $s_x, s_{xy}, s_y \in \mathbb{R}$. In this form we can see that s_x, s_y are *scaling factors* of the characteristic map while s_{xy} is a *shearing term*.

[4] `http://www.multires.caltech.edu/pubs/`

4.1 Influence of the Map W

To simplify later expressions we note that $\partial_i \mathfrak{S}\mathfrak{Ch}(\bar u) = W \cdot \partial_i \mathfrak{Ch}(\bar u)$ and define $V := W^T \cdot W =: \left[\begin{smallmatrix} E & F \\ F & G \end{smallmatrix}\right]$ with $V^{-1} = \frac{1}{EG-F^2}\left[\begin{smallmatrix} G & -F \\ -F & E \end{smallmatrix}\right]$. This gives us

$$\begin{aligned} J_{\mathfrak{S}\mathfrak{Ch}} &= |\mathbf{det}(W)|\, J_{\mathfrak{Ch}} = |\mathbf{det}(V)|^{1/2}\, J_{\mathfrak{Ch}}, \\ \mathbf{I}_{\mathfrak{S}\mathfrak{Ch}} &= D(\mathfrak{Ch})(\bar u)^T \cdot W^T \cdot W \cdot D(\mathfrak{Ch})(\bar u). \end{aligned}$$

From the latter equation we see that the inverse of $\mathbf{I}_{\mathfrak{S}\mathfrak{Ch}}$ is given by

$$\mathbf{I}_{\mathfrak{S}\mathfrak{Ch}}^{-1} = \big(D(\mathfrak{Ch})(\bar u)\big)^{-1} \cdot (W^T \cdot W)^{-1} \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-T}.$$

It is somewhat tedious, but straightforward, to check that $\Gamma^i_{\mathfrak{S}\mathfrak{Ch}} = \Gamma^i_{\mathfrak{Ch}}$. Hence we have

$$\begin{aligned} \mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(f) &= \mathbf{I}_{\mathfrak{S}\mathfrak{Ch}}^{-1} \cdot \big(\mathbf{H}(f) - (\partial_1 f \Gamma^1 + \partial_2 f \Gamma^2)\big) \\ &= \mathbf{I}_{\mathfrak{S}\mathfrak{Ch}}^{-1} \cdot Q(f, \mathfrak{Ch}), \end{aligned}$$

where $Q(f, \mathfrak{Ch})$ is some symmetric matrix depending on $f(\bar u)$ and $\mathfrak{Ch}(\bar u)$, but not on W.

4.2 Precomputing the Membrane Energy Integrals

Let $\bar u \in [0,1]^2$ and $\bar v \in \mathfrak{S}\mathfrak{Ch}([0,1]^2)$. The first order data dependent membrane energy stiffness matrix integral

$$K^{\mathbf{m}}_{ij} = \int_{\mathfrak{S}\mathfrak{Ch}([0,1]^2)} D\big(\phi_i^{\mathfrak{S}\mathfrak{Ch}}\big)(\bar v) \cdot D\big(\phi_j^{\mathfrak{S}\mathfrak{Ch}}\big)(\bar v)^T \, d\bar v$$

can be transformed to

$$\begin{aligned} &\int_{[0,1]^2} D(\phi_i)(\bar u) \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-1} \cdot W^{-1} \cdot \\ &\qquad \Big(D(\phi_j)(\bar u) \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-1} \cdot W^{-1}\Big)^T |\mathbf{det}(W)|\, |J_{\mathfrak{Ch}}(\bar u)|\, d\bar u \\ &= |\mathbf{det}(W)| \int_{[0,1]^2} D(\phi_i)(\bar u) \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-1} \cdot W^{-1} \cdot W^{-T} \cdot \\ &\qquad \big(D(\mathfrak{Ch})(\bar u)\big)^{-T} \cdot D(\phi_j)(\bar u)^T\, |J_{\mathfrak{Ch}}(\bar u)|\, d\bar u \\ &= |\mathbf{det}(V)|^{1/2} \int_{[0,1]^2} D(\phi_i)(\bar u) \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-1} \cdot V^{-1} \cdot \\ &\qquad \big(D(\mathfrak{Ch})(\bar u)\big)^{-T} \cdot D(\phi_j)(\bar u)^T\, |J_{\mathfrak{Ch}}(\bar u)|\, d\bar u \\ &= \frac{|\mathbf{det}(V)|^{1/2}}{\mathbf{det}(V)} \int_{[0,1]^2} D(\phi_i)(\bar u) \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-1} \cdot \left[\begin{smallmatrix} G & -F \\ -F & E \end{smallmatrix}\right] \\ &\qquad \cdot \big(D(\mathfrak{Ch})(\bar u)\big)^{-T} \cdot D(\phi_j)(\bar u)^T\, |J_{\mathfrak{Ch}}(\bar u)|\, d\bar u \\ &= |(EG - F^2)|^{-1/2} \cdot \Big(E \int_{[0,1]^2} f^E_{ij}(\bar u)\, d\bar u + \\ &\qquad F \int_{[0,1]^2} f^F_{ij}(\bar u)\, d\bar u + \\ &\qquad G \int_{[0,1]^2} f^G_{ij}(\bar u)\, d\bar u\Big) \\ &= (EG - F^2)^{-1/2} \cdot \big(E\, K^E + F\, K^F + G\, K^G\big)_{ij}, \end{aligned}$$

where the functions $f^E_{ij}(\bar u)$, $f^F_{ij}(\bar u)$ and $f^G_{ij}(\bar u)$ do not depend on the choice of the parameters E, F, G (or equivalently on the choice of s_x, s_{xy} and s_y). This implies that $K^{\mathbf{m}}_{\mathfrak{S}\mathfrak{Ch}}$ can be evaluated by precomputing three matrices K^E, K^F and K^G and scaling them later as needed.

Comparing the first of the equations with Equation (6) it is not hard to see that the integrals above can be computed exactly the same way as in the data independent case as sums of integrals over $L-$regions.

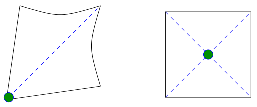


Figure 2: *Selecting sampling locations for E, F and G with respect to axial symmetries of the irregular (left) and the regular patch (right).*

4.3 Bending Energy

We show in Appendix (B) that it is possible to decompose the first order data-dependent bending energy as

$$\begin{aligned} E^{\mathbf{b}}_{\mathbf{dd}}(P) = P^T \cdot \Big(& c_{EE}(P) K^{EE} + c_{EF}(P) K^{EF} + \\ & c_{EG}(P) K^{EG} + c_{FF}(P) K^{FF} + \\ & c_{FG}(P) K^{FG} + c_{GG}(P) K^{GG}\Big) \cdot P, \end{aligned}$$

where $c_{EE} = EE\,(EG - F^2)^{-3/2}$, $c_{EF} = EF\,(EG - F^2)^{-3/2}$ and so on.

4.4 Choosing the Parameters E, F and G

We have already found an interpretation for s_x, s_{xy} and s_y as scaling and shearing terms of the characteristic map. To compute E, F and G one could estimate these parameters from the subdivision patch S using geometric properties.

There is another approach to obtain, hopefully good, values for E, F and G. It is based on the observation, that one can fix a parameter value $\bar u_s \in [0,1]^2$ and use the map W to translate and rotate the reference patch $\mathfrak{S}\mathfrak{Ch}(\bar u_s)$ into the tangential plane of the subdivision patch $S(\bar u_s)$. The map W can furthermore be used to match the first partial derivatives $\partial_i S(\bar u_s) = \partial_i \mathfrak{S}\mathfrak{Ch}(\bar u_s)$, and hence the first fundamental forms $\mathbf{I}_S(\bar u_s) = \mathbf{I}_{\mathfrak{S}\mathfrak{Ch}}(\bar u_s)$ of both patches at this particular point.

When choosing $\bar u_s$ we notice that partial derivatives $\partial_i S$ and the first fundamental form $\mathbf{I}_S$ are continuous on $[0,1]^2$. Hence any point $\bar u_s \in [0,1]^2$ is a candidate for estimating E, F and G.

The image of the characteristic map $\mathfrak{Ch}(\bar u)$ has one axis of symmetry (compare with Figure 2). It is natural to restrict the choice of $\bar u_s$ such that its image is on this axis. Hence $\bar u_s = (t, t)$ with $t \in [0,1]$.

In the regular case the choice of $\bar u_s$ should be symmetric with respect to all four corner vertices. The center $\bar u_s = (0.5, 0.5)$ of the regular patch is the only location to fulfill this requirement. Irregular patches do not have such a unique point and it is not clear which additional requirement would lead to an optimal choice of t.

Recalling Equation (4) we find

$$(\partial_i S)(0,0) = C_2(\partial_i \phi_2)(0,0) + C_3(\partial_i \phi_3)(0,0).$$

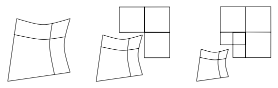
This is a convenient and easily evaluated formula and favors the point $\bar u = (0,0)$ for irregular patches.

Remarks All matrices are precomputed in an offline process. The scaling factors are easily evaluated online. Hence the evaluation of the energies is dominated by memory fetch operations accessing the matrices. The run-times for an efficient implementation will be at most $3\times$ resp. $6\times$ the time for evaluating the simple data independent energies.

For the simple operators of Section 3 we can assume $E = 1$, $F = 0$ and $G = 1$. As a numerical test of our implementation we verified, that

$$\begin{aligned} K^{\mathbf{m}}_{\mathfrak{Ch}} &= K^E + K^G \qquad \text{and} \\ K^{\mathbf{b}}_{\mathfrak{Ch}} &= K^{EE} + K^{EG} + K^{GG}. \end{aligned}$$

The approximation quality of the new operators was analyzed in [5]. Numerical experiments showed the first order data dependent membrane energy $E^{\mathbf{m}}_{\mathbf{dd}}(S^n(P))$ to approach the exact data-dependent membrane energy under repeated refinement S^n. On the other hand we cannot expect $E^{\mathbf{b}}_{\mathbf{dd}}(S^n(P))$ to converge to the exact bending energy. The second fundamental form never enters the computation of the first order data-dependent bending energy. Another consequence of refinement is the replacement, piece by piece, of the carefully crafted characteristic map parametrization with the standard spline parametrization:



This in effect recreates the problems encountered by Halstead and co-workers [7]. As a result $E^{\mathbf{b}}_{\mathbf{dd}}(S^n)$ tends to grow in each subdivision step. This indicates that the first order data-dependent bending energy is best used on coarse meshes.

5 Derivatives of Energies

At this point we have all the machinery to compute data independent as well as data-dependent energies. To change surfaces towards a minimum of these energies we need to evaluate the gradients of the energy with respect to the control points. For the data-dependent energies we furthermore need to compute Hessians of the energies with respect to the control points to invoke a robust nonlinear minimizer. Computing these derivatives is the subject of this section.

We want to regard the control points P of the subdivision patch as variables

$$E(\bar{X}) = \bar{X}^T \cdot \bar{K} \cdot \bar{X} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}^T \cdot \begin{bmatrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & K \end{bmatrix} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}.$$

The matrix $\bar{K}$ has dimension $3M \times 3M$, where M is the number of basis functions needed to represent the patch. When the matrix $\bar{K}$ does not depend on the control points, as is the case for the simple energies $E_{\mathfrak{e}\mathfrak{h}}(\bar{X})$, the gradient follows as

$$\nabla(E_{\mathfrak{e}\mathfrak{h}}(\bar{X})) = 2\,\bar{X}^T \cdot \bar{K}_{\mathfrak{e}\mathfrak{h}}$$

and the Hessian as

$$\mathbf{H}(E_{\mathfrak{e}\mathfrak{h}}(\bar{X})) = 2\,\bar{K}_{\mathfrak{e}\mathfrak{h}}.$$

In the data-dependent setting the stiffness matrices $\bar{K}(\bar{X})$ depend on the data $\bar{X}$. Fortunately the data-dependence is in the scalar factors, *e.g.*, in the bending case $\bar{K}(\bar{X}) = c_{EE}(\bar{X})\bar{K}_{EE} + c_{EF}(\bar{X})\bar{K}_{EF} + \ldots$.

For simplicity we will focus on the term $c_{EF}(\bar{X})\bar{X}^T\bar{K}_{EF}\bar{X}$. The gradient of this expression is computed as

$$\begin{aligned}\nabla(E^{EF}_{\mathbf{dd}}(\bar{X})) &= \nabla(c_{EF}(\bar{X})) \cdot \bar{X}^T \cdot \bar{K}_{EF} \cdot \bar{X} + \\ &\quad 2\,c_{EF}(\bar{X})\,\bar{X}^T \cdot \bar{K}_{EF}\end{aligned}$$

For the Hessian of the data-dependent energy we get

$$\begin{aligned}\mathbf{H}(E^{EF}_{\mathbf{dd}}(\bar{X})) &= \mathbf{H}(c_{EF}(\bar{X})) \cdot \bar{X}^T \cdot \bar{K}_{EF} \cdot \bar{X} + \\ &\quad 2\,\nabla(c_{EF}(\bar{X})) \cdot \bar{X}^T \cdot \bar{K}_{EF} + \\ &\quad 2\,\bar{K}_{EF} \cdot \bar{X} \cdot \nabla(c_{EF}(\bar{X})) + \\ &\quad 2\,c_{EF}(\bar{X})\,\bar{K}_{EF}\end{aligned}$$

All that remains to do is to evaluate $\nabla(c_{EF}(\bar{X}))$, $\mathbf{H}(c_{EF}(\bar{X}))$ and so on. This can be done by hand using the chain rule or by automatic differentiation [13]. We decided to go the first route. Experiments show that the evaluation of the gradient $\nabla(E_{\mathbf{dd}}(\bar{X}))$ with its $3M$ entries is on average only 1.5 times as expensive as a single function evaluation $E_{\mathbf{dd}}(\bar{X})$. The evaluation of the complete Hessian with $3M \times 3M$ entries is only $10 - 15$ times as expensive as a single function evaluation.

Moreton and Sequin [11] used finite differences to compute the gradient of the exact functional. Finite differences are very expensive requiring $O(M)$ function evaluations for the gradient and $O(M^2)$ for the Hessian. At the same time they achieve less accuracy. Using double precision a finite difference gradient typically only achieves $7 - 8$ digits (out of 16), while the Hessian computation deterioates to $3 - 5$ good digits.

In the next section we show that fast and accurate derivatives make nonlinear minimization very fast and effective. Compared to the linear system solver needed for the simple energies the full nonlinear minimization has a runtime which is worse by only a small factor.

6 Algorithms

Membrane energies model surfaces under tension. To produce "fair" surfaces this often is not what is desired. Imagine stretching a thin rubber sheet between a few tent poles (interpolation constraints). More natural is the use of bending energies, which we will focus on.

Energy functionals can have multiple, well seperated local minima (for the one-dimensional case see [8]). We regard any local minimum of the energy a solution to the fairing problem. This is mostly for practical reasons: the global minimum (in case it is unique) is hard to find. Even if found it could be far away from the input configuration. This might be undesirable for modeling, where a local minimum that is as close as possible to the initial configuration is more meaningful.

A neccessary condition for a local minimum X is $\nabla(E)(X) = 0$. This is not sufficient, since saddle points also satisfy this condition.

6.1 Minimizing the Simple Energy

The simple energy operator is a quadratic form $X^T \cdot K \cdot X$ with constant positive semi-definite matrix K. Consequently one can find a solution of the minimization problem by solving the linear equation system $K \cdot X = 0$. Without any further constraints $X = 0$ is a solution as are all configurations in the (often non-trivial) kernel of K [6, 7].

6.2 Minimizing the Data-Dependent Energy

A suggestion for obtaining a solution for the data-dependent problem was given by Greiner *et al.* [6]. They linearized the data-dependent functional by fixing $K_{\mathbf{dd}}(X)$ for given input data X. This leads to a linear equation system whose solution often reduces the data-dependent energy. But this solution rarely represents a local minimum of the data-dependent energy. Sometimes (especially if iterated) the data-dependent energy of the solution can become larger than the data-dependent energy of the input data.

We experimented with a number of nonlinear solvers to attack the data dependent energies. For example, we implemented a Newton solver which uses exact gradient and Hessian information, but is otherwise very similar to the approach suggested by Greiner and co-workers. The Newton method converged very fast to a local minimum when given an input vector X close to the solution. But it failed in many general settings. This observation is consistent with the observation that Newton iteration is not globally convergent—mostly because it lacks control over the length of the step taken during an iteration.

Trust region methods minimize functions iteratively. At each iteration k the input function $f(x)$ is modeled around x_k by the

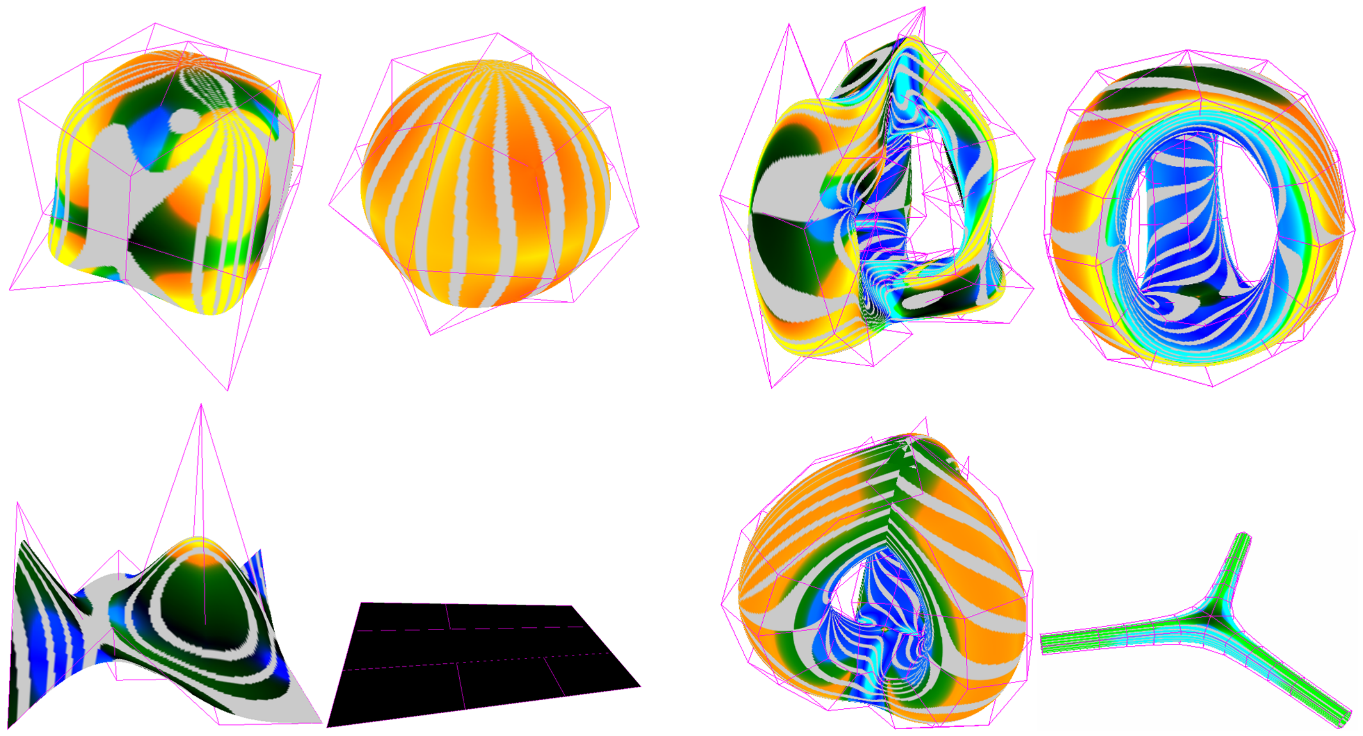


Figure 3: *The images illustrate the effect of unconstrained minimization of the first order data-dependent bending energy using the nonlinear minimizer. The top row shows the fairing of a distorted cube (78 degrees of freedom,* $0.23s$ *computation time) and of a distorted TetraThing (*636 *dofs,* $8.6s$*). The bottom left shows a bent plate flattened out (*48 *dofs,* $0.12s$*). Finally, a TetraThing with three open boundaries, unfolds under the influence of the bending forces (*696 *dofs,* $12.6s$*). Note that even grossly distorted initial configurations move towards symmetry for the cube and TetraThing. The control meshes are shown as pink lines. The limit surfaces are colored using reflection lines (grey) overlayed on curvature plots. The colors where chosen in RGB space. Red and blue denote positive/negative Gaussian curvature while green indicates non-zero mean curvature.*

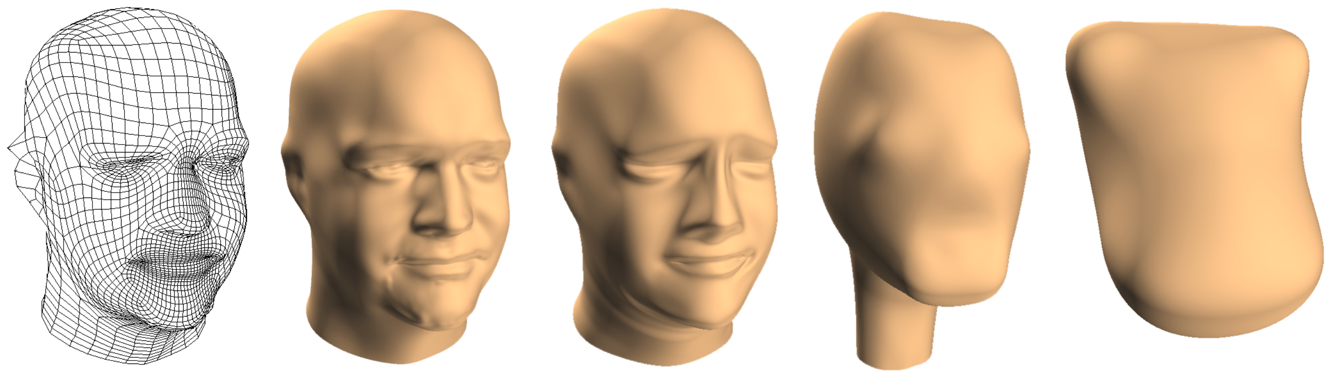


Figure 4: *Unconstrained fairing of a complex subdivision surface (*6975 *dofs). Solution until convergence took 152 iterations (*$20m5s$*) using the nonlinear minimizer. We show the original model, intermediate meshes at* 20 *and* 100 *iterations and the final mesh. Even though the model is topologically equivalent to a disk, the gradient* $\nabla(E^{\mathbf{b}}_{\mathbf{dd}})$ *disappears for the right-most mesh.*

first terms of the Taylor-series expansion.

$$m(x_k + \Delta x) = f(x_k) + \nabla(f(x_k)) \cdot \Delta x + \frac{1}{2}\Delta x^T \mathbf{H}(f(x_k))\Delta x$$

Since $f(x_k + \Delta x) = m(x_k + \Delta x) + O(\|\Delta x\|^3)$ this model is very accurate in a small neighborhood of x_k. The size of the neighborhood for which $m(x_k + \Delta x)$ is indeed a good model is estimated by the trust-region radius Δ_k. Any trust-region algorithm will adaptively keep track of this variable and minimize $m(x_k)$ to obtain $x_{k+1} = x_k + \Delta x$ subject to $\|\Delta x\| \leq \Delta_k$.

Many trust-region methods are known and all converge to local minima for arbitrary start vectors x_0. In our situation the problem size is fairly large. Gradients and Hessians are known exactly and are inexpensive to compute. Furthermore the Hessian is sparse but only guaranteed to be positive semi-definite in a neighborhood of the minimizer. For these reasons the Steihaug Newton/CG variant ([13, pp. 75ff and 156f]) was implemented.

Since we use the exact Hessian of the energy we can expect

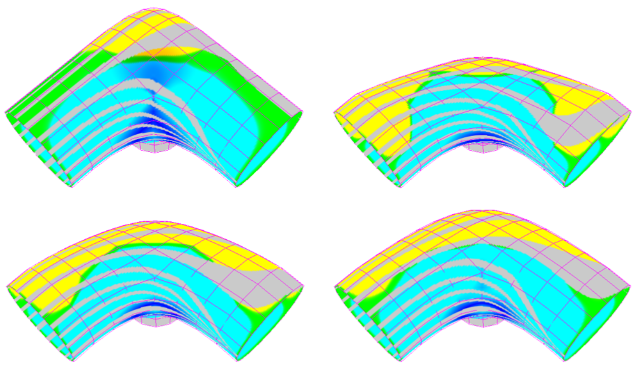


Figure 5: *In this example three pipes meet at a common joint. The two rings of control points near the boundary are fixed to effect interpolation and tangency constraints at the boundary of each pipe. The original is in the top left. The top right shows the surface resulting from minimizing $E^{\mathbf{b}}_{\mathbf{simple}}$. Note the sharp transition in the tangents and the flattening out in the elbow region. This effect is still present but much less pronounced for $E^{\mathbf{b}}_{\mathbf{dd}}$ solved linearly (bottom left). The best result is achieved with $E^{\mathbf{b}}_{\mathbf{dd}}$ minimized with the non-linear solver.*

super-linear convergence close to the minimum (similar to Newton iteration).

The implementation of the Steihaug Newton/CG method is very similar to a standard linear conjugate gradient solver. It takes about 200 lines of code to write. An important difference to the linear solver from a practical point of view is the need to recompute gradient and Hessian at each iteration. Recomputing the Hessian dominates the run-time in practice, even though we can construct it fairly cheaply.

The Steihaug Newton/CG method is exceedingly robust. Figure 3 shows a number of examples with very perturbed initial meshes converging to the expected shapes observing the symmetries in the mesh structure itself. Each of these examples has no constraints placed on the configuration. Figure 4 shows another example of unconstrained minimization.

6.3 Minimizing with Constraints

Finding unconstrained minimal energy surfaces is entertaining but of limited practical use. We need to give the designer of a surface the option to specify constraints. Typical constraints include constraining control or limit points to a given position, defining boundaries or creases, specifying tangent planes or restricting the movement of points within planes or lines. All these constraint types can be expressed as linear combinations of control points. In practice there is no difference on how these constraints are enforced within the linear and nonlinear solvers. We do this by defining the appropriate linear map $C : \mathbb{R}^{3M} \to \mathbb{R}^n$ with $n \leq 3M$ mapping control points to degrees of freedom $Y = C \cdot X$. The energy computation changes to $E(Y) = Y^T \cdot C \cdot K \cdot C^T \cdot Y$. Computing the matrix product $C \cdot K \cdot C^T$ can be impractical for general constraint matrices C. In these cases the matrix-vector product, which is the inner most loop of the sparse linear solver as well as the trust region solver, can be replaced by three consecutive matrix-vector products. In our current implementation we support control point and boundary constraints.

Figures 5, 6 and 7 show examples of constrained minimization.

7 Implementation and Results

The linear minimizer was implemented using the IML++ library [3] using a CG method with diagonal preconditioning. As a starting

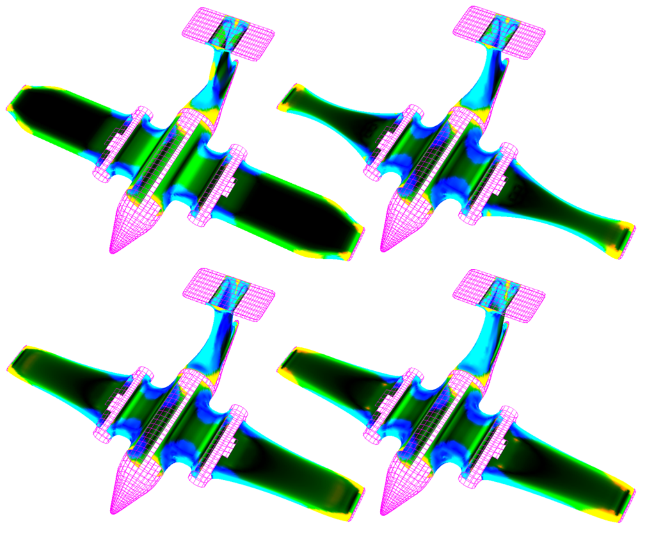


Figure 6: *The top left shows an original subdivision model of a toy airplane assembled from some simple primitives. The shape of the wings and tail fin was not as desired and consequently subjected to fairing. The magenta lines indicate constrained parts of the model while all other section were free to move. The upper right airplane is the result of minimizing the simple energy. The lower left airplane was created using the linear solver with the data-dependent energy and the lower right airplane using the nonlinear minimizer. Most noticable are the differences in the shape of the wings. Interestingly the simple bending energy results in significant pinching of the center part of the wings, a phenomenon usually associated with membrane energies. The wings created by linear and nonlinear solvers of the data-dependent bending energy look very similar but the curvature plots reveal fine differences (see Figure 7). The section between fuselage and engines reveal no significant differences between the faired shapes.*

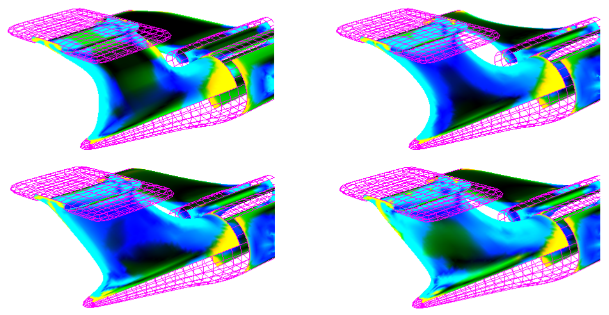


Figure 7: *A close-up of the tail section of the airplanes in Figure 6 using the same ordering. The "roundness" of the fin varies. This is also evidenced by the varying curvature distributions.*

guess for the solution vector we chose the input mesh. We are using the standard BLAS provided with IML++. The largest fraction $(50-80\%)$ of the run-time was spent in the linear equation solver. The rest of the time was used for setup and output of the solution.

Steihaug's Newton/CG trust-region method was implemented directly without using the functionality provided by IML++. Preconditioning was not implemented. Slightly more than half of the run-time accounts for recomputing $\mathbf{H}(E^{\mathbf{b}}_{\mathbf{dd}})$ at each iteration. Another quarter of the run-time was spent on matrix-vector multiplications.

While difficult to compare, we tried to achieve the same accuracy (10^{-6}) with both linear and nonlinear solvers. Figure 7 gives sizes and run-times for several sample problems. We observe that the linearized solution of the data-dependent energy takes often slightly more time than the linear solution of the simple energy. This appears to be due to worse conditioning of the stiffness matrices. The run-times of the nonlinear solver of the data-dependent problem are 3 to 6 times as big as the run-times of the corresponding linearized solver, a moderate increase given that a full blown nonlinear solver is used.

From the table we conclude that in most cases (6 out of 7) nonlinear minimization of the data-dependent bending energy achieves the surface with the lowest exact bending energy.

All run times reported in this paper where obtained on a 2GHz Intel Xeon processor under Linux using gcc with optimization.

8 Conclusions and Future Work

Subdivision surfaces are an increasingly important primitive used in geometric modeling applications. Motivated by this we have explored a number of mathematical approaches and algorithms for the fairing of such surfaces. In particular we addressed the issue of diverging energies in earlier approaches by using the characteristic map parametrization. First order data-dependent energies can be constructed by considering appropriately transformed characteristic maps as reference surfaces. This has the pleasant consequence that the resulting nonlinear minimization problem is not that much more costly than a solution to the simple linear problem. This is accomplished by observing that the final energies can be written as a linear combination of precomputed stiffness matrices. Furthermore, exact gradients and Hessians are also accessible allowing for the use of very robust nonlinear solvers such as the Steihaug Newton/CG metthod. We performed a number of comparative numerical experiments demonstrating both the performance and robustness of the algorithm.

Comparing the three different methods used, simple (*i.e.*, data independent) and first order data-dependent energies (with a linear or full nonlinear solver) we find that the data-dependent methods generally work better, but it is difficult to make stronger statements. All methods remove high frequencies from the original shapes. It is less clear which produces the best shapes. The first order data-dependence works well for coarse meshes (a few hundred to a few thousand patches) but in the bending case does not stay constant under refinement.

The method has definite limitations. The connectivity of meshes needs to be chosen carefully. For example, symmetries that are supposed to show up in the final surface need to be observed by the control mesh connectivity.

In future work we hope to understand the influence of constraints better. What are the best constraint setups to get “desirable” results? This is both a mathematical and user interface question. Further performance gains can be achieved by using BLAS libraries optimized for the Intel architecture (using SIMD instructions) as well as fine tuning the use of Hessians. These do not need to be recomputed at every step as we currently do. Finally it would be interesting to see whether a simple and efficient second order data-dependent energy can be devised such that the bending energies refine correctly.

Acknowledgment This work was supported in part by NSF (DMS 0220905, DMS 0138458, ACI 0219979) the DOE (W-7405-ENG-48/B341492), Intel, Alias|wavefront, nVidia, Pixar, and the Packard Foundation. Special thanks to Cici Koenig, Nathan Litke and Igor Guskov.

References

[1] BAKHVALOV, N. S. On approximate calculation of integrals. *Vestnik MGU, Ser. Mat. Mekh. Astron. Fiz. Khim. 4* (1959), 3–18. In Russian.

[2] CATMULL, E., AND CLARK, J. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (September 1978), 350–355.

[3] DONGARRA, J., LUMSDAINE, A., POZO, R., AND REMINGTON, K. A sparse matrix library in c++ for high performance architectures. In *Proceedings of the Second Object Oriented Numerics Conference* (1992), pp. 214–218. http://math.nist.gov/iml++/.

[4] DOO, D., AND SABIN, M. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design 10*, 6 (September 1978), 356–360.

[5] FRIEDEL, I. Data dependent energy operators for subdivision surfaces. Master’s thesis, Caltech, Department of Computer Science, 2002.

[6] GREINER, G., LOOS, J., AND WESSELINK, W. Data dependent thin plate energy and its use in interactive surface modeling. In *Computer Graphics Forum (Proc. EUROGRAPHICS ’96), 15(3)* (1996), pp. 175–186.

[7] HALSTEAD, M., KASS, M., AND DEROSE, T. Efficient, fair interpolation using Catmull-Clark surfaces. *Proceedings of SIGGRAPH 93* (1993), 35–44.

[8] HORN, B. K. P. The curve of least energy. *ACM Transactions on Mathematical Software (TOMS) 9*, 4 (1983), 441–460.

[9] HSU, L., KUSNER, R., AND SULLIVAN, J. M. Minimizing the squared mean curvature integral for surfaces in space forms. *Experimental Mathematics 1* (1992), 191–207.

[10] LOTT, N. J., AND PULLIN, D. I. Method for fairing b-spline surfaces. *Computer-Aided Design 20* (1988), 597–604.

[11] MORETON, H., AND SEQUIN, C. Functional optimization for fair surface design. *Computer Graphics 26*, Annual Conference Series (1992), 167–176.

[12] NASRI, A. H., WAN KIM, T., AND LEE, K. Fairing recursive subdivision surfaces with curve interpolation constraints. In *International Conference on Shape Modeling & Applications* (2001), pp. 49–61.

[13] NOCEDAL, J., AND WRIGHT, S. J. *Numerical Optimization*. Springer, 1999.

[14] REIF, U. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design 12*, 2 (1995), 153–174.

[15] REIF, U. *Analyse und Konstruktion von Subdivisionsalgorithmen für Freiformflächen beliebiger Topologie*. Department of Mathematics, Stuttgart University, 1998.

[16] REIF, U., AND SCHRÖDER, P. Curvature integrability of subdivision surfaces. *Advances in Computational Mathematics 2* (2001).

[17] SCHNEIDER, R., AND KOBBELT, L. Mesh fairing based on an intrinsic pde approach. *COMPUTER-AIDED DESIGN 33*, 11 (2001), 767–777.

linear $E^{\mathbf{b}}_{\mathfrak{Ch}}$	dof	run time	$E^{\mathbf{b}}_{\mathfrak{Ch}}$ before	$E^{\mathbf{b}}_{\mathfrak{Ch}}$ after	$E^{\mathbf{b}}_{\mathbf{exact}}$ after
ThreePipes	504	0.55s	$1.75 \cdot 10^{0}$	$1.52 \cdot 10^{0}$	$5.08 \cdot 10^{1}$
distortedY	993	2.21s	$1.77 \cdot 10^{-3}$	$4.27 \cdot 10^{0}$	$3.02 \cdot 10^{2}$
ThreeCylinders	1707	2.65s	$1.04 \cdot 10^{2}$	$3.16 \cdot 10^{1}$	$4.72 \cdot 10^{3}$
airplane	4575	11.41s	$1.66 \cdot 10^{1}$	$9.12 \cdot 10^{0}$	$1.97 \cdot 10^{3}$
cube2	57	0.01s	$1.40 \cdot 10^{1}$	$1.07 \cdot 10^{1}$	$3.75 \cdot 10^{1}$
cube32	18417	46.69s	$5.23 \cdot 10^{-1}$	$5.59 \cdot 10^{-2}$	$4.27 \cdot 10^{1}$
head.const	3834	15.96s	$1.58 \cdot 10^{-2}$	$1.07 \cdot 10^{-2}$	$6.42 \cdot 10^{1}$
linearized $E^{\mathbf{b}}_{\mathbf{dd}}$	dof	run time	$E^{\mathbf{b}}_{\mathbf{dd}}$ before	$E^{\mathbf{b}}_{\mathbf{dd}}$ after	$E^{\mathbf{b}}_{\mathbf{exact}}$ after
ThreePipes	504	0.58s	$3.16 \cdot 10^{1}$	$3.13 \cdot 10^{1}$	$4.95 \cdot 10^{1}$
distortedY	993	5.07s	$6.06 \cdot 10^{6}$	$9.14 \cdot 10^{2}$	$2.49 \cdot 10^{2}$
ThreeCylinders	1707	3.81s	$7.46 \cdot 10^{2}$	$3.68 \cdot 10^{5}$	$1.53 \cdot 10^{4}$
airplane	4575	18.74s	$9.19 \cdot 10^{2}$	$2.98 \cdot 10^{2}$	$9.91 \cdot 10^{2}$
cube2	57	0.01s	$4.64 \cdot 10^{1}$	$4.59 \cdot 10^{1}$	$3.75 \cdot 10^{1}$
cube32	18417	47.85s	$4.70 \cdot 10^{2}$	$6.07 \cdot 10^{1}$	$4.41 \cdot 10^{1}$
head.const	3834	22.90s	$1.19 \cdot 10^{3}$	$6.91 \cdot 10^{2}$	$8.71 \cdot 10^{1}$
nonlinear $E^{\mathbf{b}}_{\mathbf{dd}}$	dof	run time	$E^{\mathbf{b}}_{\mathbf{dd}}$ before	$E^{\mathbf{b}}_{\mathbf{dd}}$ after	$E^{\mathbf{b}}_{\mathbf{exact}}$ after
ThreePipes	504	2.16s	$3.16 \cdot 10^{1}$	$2.99 \cdot 10^{1}$	$4.82 \cdot 10^{1}$
distortedY	993	25.03s	$6.06 \cdot 10^{6}$	$3.86 \cdot 10^{2}$	$9.91 \cdot 10^{1}$
ThreeCylinders	1707	12.88s	$7.46 \cdot 10^{2}$	$1.92 \cdot 10^{2}$	$4.01 \cdot 10^{2}$
airplane	4575	66.20s	$9.19 \cdot 10^{2}$	$2.56 \cdot 10^{2}$	$7.27 \cdot 10^{2}$
cube2	57	0.14s	$4.64 \cdot 10^{1}$	$2.23 \cdot 10^{1}$	$2.67 \cdot 10^{1}$
cube32	18417	268.00s	$4.70 \cdot 10^{2}$	$5.53 \cdot 10^{1}$	$2.64 \cdot 10^{1}$
head.const	3834	140.00s	$1.19 \cdot 10^{3}$	$3.65 \cdot 10^{2}$	$1.02 \cdot 10^{2}$

Figure 8: *Sizes and run-times for different test problems. The fourth column shows the simple resp. data-dependent energy of the surface before minimization. The fifth column shows the same energy type of the surface after minimization. The final column displays the exact bending energy of the surface after minimization. The computation of the exact bending energy was performed using numerical quadrature. Due to the curvature singularities at extraordinary points a single evaluation of the exact energy was several times as expensive as the complete data-dependent minimization.*

[18] STAM, J. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Computer Graphics 32*, Annual Conference Series (1998), 395–404.

[19] TAUBIN, G. A signal processing approach to fair surface design. *Computer Graphics 29*, Annual Conference Series (1995), 351–358.

[20] THORPE, J. A. *Elementary topics in differential geometry*. Springer, 1979.

[21] WELCH, W., AND WITKIN, A. Free-form shape design using triangulated surfaces. *Computer Graphics 28*, Annual Conference Series (1994), 247–256.

[22] WESSELINK, W. *Variational Modeling of Curves and Surfaces*. Eindhoven University of Technology, 1996.

[23] ZORIN, D. *Subdivision and Multiresolution Surfaces*. PhD thesis, Caltech, 1998.

[24] ZORIN, D. Smoothness of Subdivision on Irregular Meshes. *Constructive Approximation 16*, 3 (2000), 359–397.

[25] ZORIN, D., AND SCHRÖDER, P., Eds. *Subdivision for Modeling and Animation*. Course Notes. ACM Siggraph, 2000.

A Transformation of Bending Integrals

In this section we demonstrate that the bending integrals over $L^{\mathfrak{Ch}}$ can be transformed into integrals of L just as in the case of the membrane integrals. Because we must now deal with second derivatives this excercise is somewhat messy. To keep the expressions manageable we introduce the following shorthand notations. The variables $\bar{u} = (u_1, u_2) \in L$ and $\bar{v} = (v_1, v_2) \in L^{\mathfrak{Ch}}$; the characteristic map and its inverse $\mathfrak{v}(\bar{u}) = \mathfrak{Ch}(\bar{u}) : L \to L^{\mathfrak{Ch}}$ and $\mathfrak{u}(\bar{v}) = \mathfrak{Ch}^{-1}(\bar{v}) : L^{\mathfrak{Ch}} \to L$. Indices will be used equivalently as sub- and superscripts while partial differention is indicated with the comma notation, *e.g.*, $\partial_k \partial_l f^i_j(\bar{u}) = f^i_{j,kl}(\bar{u})$.

With this notation we have $D\mathfrak{u} = (D\mathfrak{v})^{-1}$, which is equivalent to $\mathfrak{u}^k_{,i}\,\mathfrak{v}^j_{,k} = \delta^j_i$ where δ denotes the Kronecker symbol. We can compute the second partial derivatives $\mathfrak{u}^t_{,im}$ in terms of $\mathfrak{v}$ by differentiating this expression

$$0 = (\mathfrak{u}^k_{,i}\,\mathfrak{v}^j_{,k})_{,m} = \mathfrak{u}^k_{,im}\,\mathfrak{v}^j_{,k} + \mathfrak{u}^k_{,i}\,\mathfrak{v}^j_{,kq}\,\mathfrak{u}^q_{,m}.$$

Multiplying by $\mathfrak{u}^t_{,j}$ and reordering first product gives

$$\begin{aligned} 0 &= \mathfrak{u}^t_{,j}\,\mathfrak{v}^j_{,k}\,\mathfrak{u}^k_{,im} + \mathfrak{u}^t_{,j}\,\mathfrak{u}^k_{,i}\,\mathfrak{v}^j_{,kq}\,\mathfrak{u}^q_{,m} \\ &= \delta^t_k\,\mathfrak{u}^k_{,im} + \mathfrak{u}^t_{,j}\,\mathfrak{u}^k_{,i}\,\mathfrak{v}^j_{,kq}\,\mathfrak{u}^q_{,m} \\ &= \mathfrak{u}^t_{,im} + \mathfrak{u}^t_{,j}\,\mathfrak{u}^k_{,i}\,\mathfrak{v}^j_{,kq}\,\mathfrak{u}^q_{,m}. \end{aligned}$$

After appropriate renaming of the symbols we arrive at

$$\mathfrak{u}^t_{,im} = -\mathfrak{v}^c_{,ab}\,\mathfrak{u}^a_{,i}\,\mathfrak{u}^b_{,m}\,\mathfrak{u}^t_{,c}. \quad (7)$$

The right hand side involves only $\mathfrak{v}^c_{,ab}$ and first partial derivatives of $\mathfrak{u}$. The latter can be expressed in terms of $\mathfrak{v}$ through the identity $D\mathfrak{u} = (D\mathfrak{v})^{-1}$.

The entries of the stiffness matrix are integrals of the form

$$\int_{L^{\mathfrak{Ch}}} \phi^{\mathfrak{Ch}}_{k,ij}(\bar v)\,\phi^{\mathfrak{Ch}}_{l,ij}(\bar v)\; d\bar v. \tag{8}$$

Because of the growth of the terms involved we restrict our attention to the transformations of one of these factors

$$\begin{aligned}
&\int_{L^{\mathfrak{Ch}}} \phi^{\mathfrak{Ch}}_{,ij}(\bar v)\, d\bar v \\
&\quad= \int_{L^{\mathfrak{v}}} (\phi\circ\mathfrak{v})_{,ij}(\bar v)\; d\bar v \\
&\quad= \int_{L^{\mathfrak{v}}} (\phi_{,k}(\mathfrak{u}(\bar v))\,\mathfrak{u}^k_{,j}(\bar v))_{,i}\, d\bar v \\
&\quad= \int_{L^{\mathfrak{v}}} \phi_{,ks}(\mathfrak{u}(\bar v))\,\mathfrak{u}^s_{,i}(\bar v)\,\mathfrak{u}^k_{,j}(\bar v) + \phi_{,k}(\mathfrak{u}(\bar v))\,\mathfrak{u}^k_{,ji}(\bar v)\, d\bar v.
\end{aligned}$$

After substituting $\bar v = \mathfrak{v}(\bar u)$ we get

$$\int_L \Big(\phi_{,ks}(\bar u)\mathfrak{u}^s_{,i}\big(\mathfrak{v}(\bar u)\big)\mathfrak{u}^k_{,j}\big(\mathfrak{v}(\bar u)\big) + \phi_{,k}(\bar u)\mathfrak{u}^k_{,ji}\big(\mathfrak{v}(\bar u)\big)\Big)|J_{\mathfrak{v}}(\bar u)|\, d\bar u. \tag{9}$$

The final formula, which depends only on derivatives of ϕ and $\mathfrak{v}$ but not its inverses, is obtained by replacing $\mathfrak{u}^s_{,i}$ and $\mathfrak{u}^k_{,ji}$ with help of the identity $D\mathfrak{u} = (D\mathfrak{v})^{-1}$ and Equation (7).

All functions involved in this final formula are eigen functions of the subdivision patch parametrized over L. The eigen functions are C^2 on an open neighborhood including L. Because of the regularity of $\mathfrak{Ch}$ [14] the determinant of $\mathfrak{Ch}$ does not vanish. As a consequence the integrand of Equation 9 (and also the integrand of Equation 8) is smooth and bounded on L. Hence it can be computed accurately with high order quadrature rules.

B Data-Dependent Bending Energy

As in the unscaled case, we try to avoid numerical integration close to the origin, where the integrand has poles. By definition of $\mathfrak{S}\mathfrak{Ch}$ we have $\mathfrak{S}\mathfrak{Ch}(2^{-n}\bar u) = W\cdot\mathfrak{Ch}(2^{-n}\bar u) = \big(W\cdot\left[\begin{smallmatrix}\lambda & 0\\ 0 & \lambda\end{smallmatrix}\right]^n\big)\cdot\mathfrak{Ch}(\bar u)$. This leads to the idea that the necessary change of the integration domain from $2^{-n}L$ to L is governed by the rules derived for the change of $\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}$ under application of a linear map and the scaling relations of the eigen functions. Recall, that

$$\begin{aligned}
&K_{ij} \\
&\quad= \int_{[0,1]^2} \mathbf{tr}\Big(\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}\big(\phi_i(\bar v)\big)\cdot\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}\big(\phi_j(\bar v)\big)^T\Big)\,|J_{\mathfrak{S}\mathfrak{Ch}}(\bar v)|\, d\bar v \\
&\quad= \sum_{n=0}^{\infty}\int_{2^{-n}L} \mathbf{tr}\Big(\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}\big(\phi_i(\bar v)\big)\cdot\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}\big(\phi_j(\bar v)\big)^T\Big)\,|J_{\mathfrak{S}\mathfrak{Ch}}(\bar v)|\, d\bar v.
\end{aligned}$$

We pick one of the integrals in the sum and fix n. Now let us examine the effects of the substitution $\bar v = 2^{-n}\,\bar u$ on the terms in

$$\begin{aligned}
&\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_j(\bar v)) \\
&\quad= \mathbf{I}^{-1}_{\mathfrak{S}\mathfrak{Ch}}(\bar v)\cdot\big(\mathbf{H}(\phi_j(\bar v)) - (\partial_1\phi_j(\bar v)\Gamma^1(\bar v) + \partial_2\phi_j(\bar v)\Gamma^2(\bar v))\big).
\end{aligned}$$

The derivatives of the eigen functions scale as

$$\begin{aligned}
\partial_k\phi_i(2^{-n}\bar u) &= 2^n\lambda_i^n\,\partial_k\phi_i(\bar u)\\
\partial_k\partial_l\phi_i(2^{-n}\bar u) &= 2^{2n}\lambda_i^n\,\partial_k\partial_l\phi_i(\bar u),
\end{aligned}$$

which implies the identities

$$\begin{aligned}
\mathbf{H}(\phi_j)(2^{-n}\bar u) &= 2^{2n}\lambda_j^n\,\mathbf{H}(\phi_j)(\bar u)\\
\mathbf{I}^{-1}_{\mathfrak{S}\mathfrak{Ch}}(2^{-n}\bar u) &= (2\lambda)^{-2n}\,\mathbf{I}^{-1}_{\mathfrak{S}\mathfrak{Ch}}(\bar u)\\
\Gamma^k_{ij}(2^{-n}\bar u) &= g^{kl}\langle\partial_i\partial_j\mathfrak{Ch},\partial_l\mathfrak{Ch}\rangle = 2^n\,\Gamma^k_{ij}(\bar u).
\end{aligned}$$

These finally result in

$$\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_j)(2^{-n}\bar u) = \big(\frac{\lambda_j}{\lambda^2}\big)^n\,\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_j)(\bar u).$$

The Jacobian scales as before $J_{\mathfrak{S}\mathfrak{Ch}}(2^{-n}\bar u) = \lambda^{2n}\,J_{\mathfrak{S}\mathfrak{Ch}}(\bar u)$.

With these identities we can reduce the entire integral to one over a single instance of L

$$\begin{aligned}
&\int_{2^{-n}L} \mathbf{tr}\Big(\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}\big(\phi_i(\bar v)\big)\cdot \\
&\qquad\qquad \mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}\big(\phi_j(\bar v)\big)^T\Big)\,|J_{\mathfrak{S}\mathfrak{Ch}}(\bar v)|\, d\bar v \\
&= \int_L \mathbf{tr}\Big(\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_i(\bar u))\Big(\frac{\lambda_i}{\lambda^2}\Big)^n\Big(\frac{\lambda_j}{\lambda^2}\Big)^n \\
&\qquad\qquad \mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_j(\bar u))^T\Big)\,\lambda^{2n}\,|J_{\mathfrak{S}\mathfrak{Ch}}(\bar u)|\, d\bar u \\
&= \Big(\frac{\lambda_i\lambda_j}{\lambda^2}\Big)^n\int_L \mathbf{tr}\Big(\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_i(\bar u))\cdot \\
&\qquad\qquad \mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_j(\bar u))^T\Big)\,|J_{\mathfrak{S}\mathfrak{Ch}}(\bar u)|\, d\bar u.
\end{aligned}$$

The last formula shows the same scaling relation as the simple bending integral (Equation 5).

B.1 Precomputing the Bending Energy Integrals

Now we want to find an efficient way to assemble the matrix $K_{\mathfrak{S}\mathfrak{Ch}}$. We have

$$\begin{aligned}
&K^{\mathfrak{S}\mathfrak{Ch}}_{ij}\Big(1-\frac{\lambda_i\lambda_j}{\lambda^2}\Big) \\
&\quad= \int_L \mathbf{tr}\Big(\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_i)^T\cdot\mathbf{H}_{\mathfrak{S}\mathfrak{Ch}}(\phi_j)\Big)\, d\omega_{\mathfrak{S}\mathfrak{Ch}} \\
&\quad= \int_L \mathbf{tr}\Big(Q(\phi_i,\mathfrak{Ch})\cdot\mathbf{I}^{-1}_{\mathfrak{S}\mathfrak{Ch}}\cdot\mathbf{I}^{-1}_{\mathfrak{S}\mathfrak{Ch}}\cdot Q(\phi_j,\mathfrak{Ch})\Big)|\mathbf{det}(V)|^{1/2}\, d\omega_{\mathfrak{Ch}} \\
&\quad= |\mathbf{det}(V)|^{1/2}\int_L \mathbf{tr}\Big(Q(\phi_i,\mathfrak{Ch})\cdot D\mathfrak{Ch}^{-1}\cdot V^{-1}\cdot D\mathfrak{Ch}^{-1}\cdot \\
&\qquad\qquad D\mathfrak{Ch}^{-1}\cdot V^{-1}\cdot D\mathfrak{Ch}^{-1}\cdot Q(\phi_j,\mathfrak{Ch})\Big)\, d\omega_{\mathfrak{Ch}},
\end{aligned}$$

where we exploited the fact that $\mathbf{tr}(A\cdot A^T) = \mathbf{tr}(A^T\cdot A)$. Since $V^{-1} = \frac{1}{EG-F^2}\cdot\left[\begin{smallmatrix}G & -F\\ -F & E\end{smallmatrix}\right]$ this may be rewritten as

$$\begin{aligned}
K^{\mathfrak{S}\mathfrak{Ch}}_{ij} &= \frac{(EG-F^2)^{\frac12}}{(EG-F^2)^2}\Big(1-\frac{\lambda_i\lambda_j}{\lambda^2}\Big)^{-1} \\
&\int_L E^2\,f^{EE}_{ij}(\bar u) + EF\,f^{EF}_{ij}(\bar u) + EG\,f^{EG}_{ij}(\bar u) + \\
&\qquad F^2\,f^{FF}_{ij}(\bar u) + FG\,f^{FG}_{ij}(\bar u) + G^2\,f^{GG}_{ij}(\bar u)\, d\bar u,
\end{aligned}$$

where each of the functions f^{xx} is independent of the parameters E, F and G. With $xx\in\{EE, EF, EG, FF, FG, GG\}$ we get

$$K^{\mathfrak{S}\mathfrak{Ch}}_{ij} = \sum_{xx} xx\,(EG-F^2)^{-3/2}\Big(\big(1-\frac{\lambda_i\lambda_j}{\lambda^2}\big)^{-1}\int_{L^u} f^{xx}_{ij}(\bar u)\, d\bar u\Big),$$

which can be used to define 6 ancillary matrices

$$K^{xx}_{ij} = \Big(1-\frac{\lambda_i\lambda_j}{\lambda^2}\Big)^{-1}\int_{L^u} f^{xx}_{ij}(\bar u)\, d\bar u.$$