Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

DIPLOMARBEIT

Quadrilateral Meshes Generated by Fairing Structure Lines

Ausgeführt am Institut für
Diskrete Mathematik und Geometrie
der Technischen Universität Wien

unter Anleitung von
o.Univ-Prof. Dr. Helmut Pottmann

durch
Marlies Grüner
Laurenzgasse 3/8
1050 Wien

Datum

Unterschrift

Abstract

One of the main problems in architectural geometry is the generation of aesthetically appealing surfaces. This process occasionally conditions the application of appropriate smoothing methods. In this thesis a particular case of 3-dimensional planar quadrilateral meshes is observed. The smoothness of a quadrilateral mesh is defined, not through differentiability like in the continuous case, but based on the trend of a structure line; the more rectilineal a polygon becomes, the smoother it gets. This thesis also sets out to preliminary introduce well-known smoothing methods; such as the minimization of energy functionals, the Laplacian smoothing method or Taubin's smoothing method. Subsequently an alternative fairing concept is introduced, which 'smoothes' the undesirable zigzag lines with methods of optimization. This approach examines the smoothness of each point in advance and flattens it if necessary. The distance between the smoothed points can be directed towards the initial point via a penalty function. With reference to examples, the results of the smoothing through the alternative fairing method and that of the well-known methods of Laplace and Taubin can be compared.

Kurzfassung

Das Generieren ästhetisch ansprechender Flächen ist in der architektonischen Geometrie von besonderem Interesse. Dieser Prozess bedingt mitunter das Anwenden geeigneter Glättungsverfahren. In der vorliegenden Diplomarbeit wird der spezielle Fall von diskreten 3-dimensionalen planaren Vierecksflächen betrachtet. Die Glätte eines Netzes wird - nicht wie im kontinuierlichen Fall durch die Differenzierbarkeit, sondern - anhand des Verlaufs einer Strukturlinie bestimmt; je geradliniger ein Polygon wird, umso glatter ist es. Einleitend werden dazu in dieser Arbeit bekannte Glättungsverfahren vorgestellt; wie zum Beispiel das Minimieren von Energiefunktionalen, das Verfahren von Laplace oder Taubin. Anschließend wird ein Fairing-Konzept präsentiert, welches mit Hilfe von Optimierungsmethoden die unerwünschten Zickzacklinien aus dem Netz „glättet“. Das Verfahren untersucht vorab jeden Punkt auf dessen Glattheit und fixiert ihn gegebenenfalls. Mit Hilfe einer Gewichtsfunktion kann der Abstand der geglätteten Punkte zu den Ausgangspunkten gesteuert werden. Anhand von Beispielen werden die Ergebnisse der Glättung dieser Fairing-Methode mit denen der bekannten Verfahren von Laplace und Taubin verglichen.

Danksagung

Ich danke meinem Betreuer Prof. Helmut Pottmann für die hilfreiche und wegweisende Unterstützung während der Konzeption meiner Diplomarbeit. Zu großem Dank bin ich auch den Assistenten Simon Flöry und Bernhard Blaschitz des Instituts "Geometric Modeling and Industrial Geometry" an der Technischen Universität Wien verpflichtet, die sich die Zeit nahmen meine Fragen zu beantworten und mir dabei behilflich waren, konzeptionelle Probleme zu lösen bzw. Heinz Schmiedhofer, der mir geholfen hat die Beispiele des letzten Kapitels in der hier angeführten Form präsentieren zu können.

Meinen Freundinnen Claudia Weidl, Cornelia Derdak, Annie Marsh und Michaela Lutz verdanke ich die korrigierte Version dieser Diplomarbeit.

Für die jahrelange Unterstützung und das Verständis, welches mir während meiner Studienzeit entgegen gebracht wurde, bedanke ich mich bei meiner Familie und meinen Freunden. Im Speziellen danke ich meinem Bruder, sowie meinem Vater, der mich auf die Idee gebracht hat, dass auch "Technische" Mathematik Schwerpunkt meines Interesses sein kann und meiner Mutter, die mich ebenso zeitlebens motivierte, obwohl ich keine Chirurgin geworden bin.

Contents

Abstract i

Kurzfassung ii

Danksagung iii

1 Introduction 1

2 Differential Geometry 3
- 2.1 Basics of Differential Geometry 3
  - 2.1.1 (Surface) Curves and Curve Networks 3
  - 2.1.2 Regular Surfaces 5
  - 2.1.3 Fundamental Form 6
  - 2.1.4 Curvatures 7
  - 2.1.5 Laplace-Operator 9
- 2.2 Basics of Discrete Differential Geometry 10
  - 2.2.1 Laplace Operator 11
  - 2.2.2 Laplace-Beltrami Operator 12
  - 2.2.3 Discrete Curvature 13

3 Presentation of the well-known Smoothing and Fairing Methods 17
- 3.1 Minimization of the Bending Energy 17
  - 3.1.1 Energy of Polygons and Polygon Networks 18
  - 3.1.2 Energy of the Curvature 21
  - 3.1.3 Energy of the Variation of the Curvature 23
- 3.2 Fourier Analysis on Meshes 24
  - 3.2.1 The Method of Fourier Descriptors 25
  - 3.2.2 The Laplacian Smoothing Method 26
  - 3.2.3 Taubin's Smoothing Method 28
  - 3.2.4 Constraints 30
  - 3.2.5 Different Ways of choosing Weights 32
- 3.3 Spring Model 33
- 3.4 Algorithm of Schneider-Kobbelt 37

4 Explanation of the alternative Fairing Method 42
- 4.1 The Algorithm 46
- 4.2 The Influence of the Parameters 47

5 Examples **50**
5.1 Example 1 50
5.2 Example 2 53
5.3 Example 3 56
5.4 Example 4 59

6 Conclusion **62**

List of Figures **63**

Bibliography **64**

1 Introduction

The application of aesthetically appealing (meshes approximating) surfaces is becoming more and more important in architecture. While a surface may be smooth in the mathematical - differentiable - sense, it does not have to be satisfying in an aesthetic view. Therefore the measure of the 'well-shape-ness' - especially for meshes - is more difficult to define in technical terms. The process of designing such surfaces or meshes is called *fairing.*

This thesis gives an overview of well-established smoothing and fairing methods and a variety of techniques to obtain the desired effect. The methods change the vertex positions either iteratively or by calculate them by solving a large sparse linear system. The methods were chosen in a way that a variety for smoothing and fairing a mesh is presented. Additionally some constraints - such as deviation for each point, planarity[1] of the resulting mesh or boundary constraints - are mentioned.

After giving an overview, this thesis provides another concept for fairing quadrilateral meshes. The new approach is linear in the unknowns and minimizes the variation of consecutive direction vectors of a structure line. Certain constraints, which will be described in the associated section, are imposed for the new as well as for the well-known methods. On the basis of examples, the efficiency of this method is underlined by the comparison of the new method and the ones mentioned before.

The remainder of this thesis is organized as follows:

- Chapter 2 gives a short introduction to differential geometry, providing a consistent notation. The section is divided into differential geometry and the discrete counterpart with the focus on what will be needed in the following sections.
- In Chapter 3, a review of existing smoothing and fairing methods is given. The first subsection deals with the minimization of well-known energy functionals and the energy (of the variation) of the curvature in the continuous as well as in the discrete case, cf. [11], [30], [10] and [29]. The second subsection presents the well-established Laplacian smoothing method and Taubin's smoothing method described in [27], [26] and [28] based on Fourier Analysis. The third method is an iteration-based algorithm which moves vertices to suitable positions while minimizing the curvature variation, cf. [33]. The last algorithm of Schneider-Kobbelt deals with solving a differential equation to create fair discrete surfaces, see [20] and [21].
- Chapter 4 provides a new fairing method which eliminates zigzag lines. The functionality and the constraints are explained and the the unfluence of the modifiable

---

[1] The input meshes of the examples are indeed planar, but the resulting meshes do not have planar faces anymore. For the sake of completeness the planarity of meshes is mentioned but the requirement of planar faces in the resulting mesh goes beyond the scope of this thesis.

parameters are presented via some examples.

- Chapter 5 compares the new fairing concept with the Laplacian smoothing method and the Taubin smoothing method. The presentation of four examples will show the assets and drawbacks of each method.
- Chapter 6 concludes the thesis.

2 Differential Geometry

This section introduces the basic terms from Differential Geomtry and Discrete Differential Geometry. A consistent notation which will be needed in the following sections is presented and some technicalities that later on allow us to describe theoretical properties of fair polygon networks are resolved. The first subsection is about the basics of Differential Geometry and the second about the discretization.

2.1 Basics of Differential Geometry

Before discussing discrete analogs, we briefly review the used theory of Differential Geometry for curves and surfaces in this thesis and define a consistent notation. For further details we refer to [12] and [6].

2.1.1 (Surface) Curves and Curve Networks

Definition 2.1 (Parametric differentiable Curve)**.** A *parametric differentiable curve* is a continuous differentiable mapping $c : I \to \mathbb{R}^n$, where $I \subseteq \mathbb{R}$ is an open interval.

Definition 2.2 (Regular Curve)**.** A parametric differentiable curve $c : I \to \mathbb{R}^n$ is called *regular*, if $c'(t) \neq 0$ for all $t \in I$. A point t with $c(t) = 0$ is called *singularity*.

The differentiability in the definition above means that the function c maps each $t \in I$ onto a point $c(t) = (x_1(t), \ldots, x_n(t)) \in \mathbb{R}^n$ such that the functions $x_1(t), \ldots, x_n(t)$ are differentiable. The variable t is called the *parameter* of the curve.

Definition 2.3 (Frenet Frame)**.** Let $c(s)$ be a curve in $\mathbb{R}^3$ in arc length parameterization and we assume $c''(t) \neq 0$. At each point of the curve, we define three pairwise orthogonal unit vectors $(\mathbf{t}, \mathbf{n}, \mathbf{b})$ as follows:

$$\begin{aligned} \mathbf{t} &= c' && \ldots \textit{Unit Tangent Vector} \\ \mathbf{n} &= \frac{c''}{\|c''\|} && \ldots \textit{Principal Normal Vector} \\ \mathbf{b} &= \mathbf{t} \times \mathbf{n} && \ldots \textit{Binormal Vector} \end{aligned}$$

By differentiation of the identity $(c')^2 = 1$ we find $c' \cdot c'' = 0$ and thus we see that the two vectors $\mathbf{t}$ and $\mathbf{n}$ are orthogonal. Of course, they span the osculating plane. The orthonormal frame $(\mathbf{t}, \mathbf{n}, \mathbf{b})$ is called *Frenet frame.*

Remark 2.1. It will be useful to use the convention that derivatives with respect to arc length s are indicated by primes $(c', c'', \ldots)$, whereas derivatives with respect to another parameter t are written with dots $(\dot{c}, \ddot{c}, \ldots)$.

Lemma 2.1. *The Frenet frame of a C^3 curve $c(s)$ in arc length parameterization fulfills the equations*

$$\begin{aligned}\mathbf{t}' &= \kappa\,\mathbf{n},\\ \mathbf{n}' &= -\kappa\,\mathbf{t} + \tau\,\mathbf{b},\\ \mathbf{b}' &= -\tau\,\mathbf{n},\end{aligned}$$

with $\kappa = \|\mathbf{t}'\| = \|\mathbf{c}''\|$ and $\tau = \det(c', c'', c''')/c''^2$.

The derivation of the Frenet formula is demonstrated in [1]. By construction, the coefficients $\kappa(s)$ and $\tau(s)$ are Euclidean invariants of the curve.

Definition 2.4 (Curvature)**.** κ is called *curvature* and τ *torsion* at the point $c(s)$.

For a curve given by an arbitrary parameterization $c(t)$, one can show the following expressions for curvature and torsion

$$\kappa = \frac{\|\dot{c} \times \ddot{c}\|}{\|\dot{c}\|^3},$$

$$\tau = \frac{\det(\dot{c}, \ddot{c}, c^{(3)})}{(\dot{c} \times \ddot{c})^2}.$$

Definition 2.5 (Curve Network)**.** A *curve network* is a finite set of curves $\mathcal{C} = \{c\}$, each defined in its parameter interval $[a_c, b_c]$.

Definition 2.6 (Knot)**.** We call points which are common to more than one curve *knots*. Each knot k of the curve network has a collection C_k^s of curves starting there and another set C_k^e of curve segments ending there. The location of the knot in space is some point $\mathbf{p}_k$. So if a curve c, defined in the interval $[a_c, b_c]$ starts in the knot $\mathbf{p}_k$, i.e., $c \in C_k^s$, then $c(a_c) = \mathbf{p}_k$, and analogously, if a curve c ends in the knot $\mathbf{p}_k$, i.e., $c \in C_k^e$, then $c(b_c) = \mathbf{p}_k$.

Definition 2.7 (Connectivity)**.** The knots $\mathbf{p}_k$ together with the sets C_k^s, C_k^e define the *connectivity* of the network.

Definition 2.8 (Outgoing and Incoming Curves)**.** The curves in the set C_k^s are called *outgoing curves* of the knot $\mathbf{p}_k$, and the curves in the set C_k^e are called *incoming curves* of $\mathbf{p}_k$.

To obtain nice polygon networks we often want two curve segments joining in a knot (an incoming curve and an outgoing one) to actually belong to one larger curve. More formally, a curve $c_e \in C_k^e$ ending in a knot $\mathbf{p}_k$ together with a curve $c_s \in C_k^s$ starting in $\mathbf{p}_k$ may be required to form a single smooth curve.

Definition 2.9 (Structure Line)**.** Two curve segments (c_e, c_s) with the above properties are part of a *structure line* of the curve network.

Note that structure lines can also consist of more than two curve segments [30].

Definition 2.10 (Geodesic Line)**.** A *geodesic line* has the property that its main normal vector is parallel to the surface normal vector at every point of the line [5].

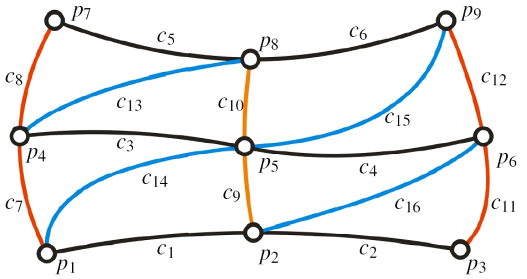


Figure 2.1: Example of a curve network given by curves $c_1, \dots, c_{16}$. The connectivity is defined by the knots $p_1, \dots, p_9$. The structure lines are: (c_1, c_2), (c_3, c_4), (c_5, c_6), (c_7, c_8), (c_9, c_{10}), (c_{11}, c_{12}), (c_{14}, c_{15}).

2.1.2 Regular Surfaces

Definition 2.11 (Parametric Surface)**.** A subset $S \subset \mathbb{R}^3$ is called *parametric surface*, if for each point $\mathbf{p} \in S$, a neighborhood $V \in \mathbb{R}^3$ exists and a continuous mapping $\mathbf{x} : U \to V \cap S$ from an open set $U \subset \mathbb{R}^2$ onto $V \cap S \subset \mathbb{R}^3$, so that

$$\mathbf{x} : (u, v) \in U \mapsto \mathbf{x}(u, v) = (x_1(u, v), x_2(u, v), x_3(u, v)) \in S.$$

$\mathbf{x}$ is called *parameterization.*

Definition 2.12 (Regular Surface)**.** The parameterized surface is called *regular*, if the partial derivative vectors $\frac{\partial \mathbf{x}}{\partial u}$, $\frac{\partial \mathbf{x}}{\partial v}$ are linearly independent for all $(u, v) \in S$.

Definition 2.13 (Tangent Plane)**.** A curve $(u(t), v(t))$ in $\mathbf{x}$'s parameter domain defines a surface curve $\mathbf{c}(t) = \mathbf{x}(u(t), v(t))$. The tangent of this curve has the direction vector

$$\dot{\mathbf{c}}(t) = \dot{u}\frac{\partial \mathbf{x}}{\partial u} + \dot{v}\frac{\partial \mathbf{x}}{\partial v}.$$

This vector is a linear combination of the two vectors $\frac{\partial \mathbf{x}}{\partial u}$, $\frac{\partial \mathbf{x}}{\partial v}$. Therefore, the tangent of an arbitrary surface curve at a given surface point lies in the plane spanned by $\frac{\partial \mathbf{x}}{\partial u}$, $\frac{\partial \mathbf{x}}{\partial v}$, the so-called *tangent plane* $T_{\mathbf{p}}(S)$.

Definition 2.14 (Unit Normal Vector Field)**.** Given a parameterization $\mathbf{x} : U \subset \mathbb{R}^2 \to S$ at $\mathbf{p} \in S$, the *unit normal vector* at each surface point $\mathbf{q} \in \mathbf{x}(U)$ is defined by

$$\mathbf{n}(\mathbf{q}) = \frac{\partial \mathbf{x}/\partial u \times \partial \mathbf{x}/\partial v}{\|\partial \mathbf{x}/\partial u \times \partial \mathbf{x}/\partial v\|}(\mathbf{q}).$$

The differentiable mapping $\mathbf{n} : \mathbf{x}(U) \to \mathbb{R}^3$ assigns each $\mathbf{q} \in \mathbf{x}(U)$ a unit normal vector $\mathbf{n}(\mathbf{q})$. If $V \subset S$ is an open subset of S and $\mathbf{n} : V \to \mathbb{R}^3$ is a differentiable mapping, $\mathbf{n}$ is called a *differentiable unit normal vector field.*

Definition 2.15 (Orientation)**.** A regular surface is called *orientable*, if a globally differentiable unit normal vector field can be defined on the surface. The choice of such a vector field $\mathbf{n}$ is called *orientation* of the surface.

2.1.3 Fundamental Form

Definition 2.16 (First Fundamental Form)**.** Given a regular surface S. Let $\langle .,. \rangle$ denote the Euclidean scalar product on $\mathbb{R}^3$ (and as tangent spaces are subsets of $\mathbb{R}^3$ on each tangent space $T_{\mathbf{p}}\mathbb{R}^3$). The inner product restricted to each tangent space $T_{\mathbf{p}}(S)$ is denoted by $\langle .,. \rangle_{\mathbf{p}}$. The bilinear form $\mathbf{I}(.) = \langle .,. \rangle_{\mathbf{p}}$ is called *first fundamental form* of the surface S in $\mathbf{p} \in S$.

Let $\mathbf{x}(u,v)$ be a parametric surface. Then the inner product of a tangent vector[2] $\dot{\mathbf{c}}(t) = \dot{u}\mathbf{x}_u + \dot{v}\mathbf{x}_v$ of an parametric curve $\mathbf{c}(t)$ is

$$\begin{aligned}\mathbf{I}(\dot{u}\mathbf{x}_u + \dot{v}\mathbf{x}_v) &= \langle \dot{u}\mathbf{x}_u + \dot{v}\mathbf{x}_v, \dot{u}\mathbf{x}_u + \dot{v}\mathbf{x}_v \rangle_{\mathbf{p}} \\ &= \langle \mathbf{x}_u, \mathbf{x}_u \rangle_{\mathbf{p}}\, \dot{u}^2 + 2\langle \mathbf{x}_u, \mathbf{x}_v \rangle_{\mathbf{p}}\, \dot{u}\,\dot{v} + \langle \mathbf{x}_v, \mathbf{x}_v \rangle_{\mathbf{p}}\, \dot{v}^2 \\ &= E(u,v)\,\dot{u}^2 + 2F(u,v)\,\dot{u}\,\dot{v} + G(u,v)\,\dot{v}^2.\end{aligned}$$

The bivariate functions

$$\begin{aligned}E(u,v) &= \langle \mathbf{x}_u, \mathbf{x}_u \rangle_{\mathbf{p}} \\ F(u,v) &= \langle \mathbf{x}_u, \mathbf{x}_v \rangle_{\mathbf{p}} \\ G(u,v) &= \langle \mathbf{x}_v, \mathbf{x}_v \rangle_{\mathbf{p}}\end{aligned}$$

are called the *coefficients of the first fundamental form.*

In differential geometry, properties that only depend on the first fundamental form are called *intrinsic.* Intuitively, the intrinsic geometry of a surface can be perceived by $2D$ creatures that live on the surface without knowledge of the third dimension. Examples include length and angles of curves on the surface [3]. The coefficients of the first fundamental form can be used to measure the angle of two surface curves intersecting at a point (if these curves possess tangent vectors), to calculate the lengths of curves on the surface and the areas of regions on the surface.

Definition 2.17 (Gaussian Map)**.** Let $S \subset \mathbb{R}^3$ be a surface with orientation $\mathbf{n}$. The mapping $\mathbf{n} : S \to S^2 : \mathbf{p} \mapsto \mathbf{n}(\mathbf{p})$, where S^2 denotes the unit sphere is called *Gaussian map.*

The Gaussian map is differentiable and its differential $d\mathbf{n}_{\mathbf{p}}$ is a linear mapping from $T_{\mathbf{p}}(S)$ to $T_{\mathbf{n}(\mathbf{p})}(S^2)$. Since $T_{\mathbf{p}}(S)$ and $T_{\mathbf{n}(\mathbf{p})}(S^2)$ are parallel planes, $d\mathbf{n}_{\mathbf{p}}$ can be considered as a linear mapping of $T_{\mathbf{p}}(S)$ onto itself. Since $d\mathbf{n}_{\mathbf{p}} : T_{\mathbf{p}}(S) \to T_{\mathbf{p}}(S)$ is a self-adjoint linear mapping[4], a quadratic form Q on $T_{\mathbf{p}}(S)$ given by $Q(\mathbf{v}) = \langle d\mathbf{n}_{\mathbf{p}}(\mathbf{v}), \mathbf{v} \rangle$, $\mathbf{v} \in T_{\mathbf{p}}(S)$ can be assigned to $d\mathbf{n}_{\mathbf{p}}$.

---

[2] To obtain more compact formulae, we write $\mathbf{x}_u$ instead of $\frac{\partial \mathbf{x}}{\partial u}$.

[3] Note that the term "intrinsic" is also often used to denote independence of a particular parametrization.

[4] For further information we refer to [6].

Definition 2.18 (Second Fundamental Form)**.** The quadratic form $\mathbf{II_p}$ defined on $T_{\mathbf{p}}(S)$

$$\mathbf{II_p}(\mathbf{v}) = -\langle\, d\mathbf{n_p}(\mathbf{v}), \mathbf{v}\,\rangle$$

is called *second fundamental form* of S at $\mathbf{p}$.

On the surface S we consider a curve $\mathbf{c}(s) = \mathbf{x}(u(s), v(s))$, with s as arc length of $\mathbf{c}$. Then, the second derivative vector is

$$\mathbf{c}'' = u''^2\,\mathbf{x}_{uu} + 2u'\,v'\,\mathbf{x}_{uv} + v''^2\,\mathbf{x}_{vv} + u''\,\mathbf{x}_u + v''\,\mathbf{x}_v.$$

We are interested in the normal component of the vector $\mathbf{c}''$. Thus, we compute its inner product with the unit normal vector $\mathbf{n}$,

$$\mathbf{c}''\cdot\mathbf{n} = (\mathbf{x}_{uu}\cdot\mathbf{n})\,u'^2 + 2\,(\mathbf{x}_{uv}\cdot\mathbf{n})\,u'\,v' + (\mathbf{x}_{vv}\cdot\mathbf{n})\,v'^2.$$

The coefficients of $\mathbf{c}''\cdot\mathbf{n}$

$$\begin{aligned} L &= \mathbf{x}_{uu}\cdot\mathbf{n} \\ M &= \mathbf{x}_{uv}\cdot\mathbf{n} \\ N &= \mathbf{x}_{vv}\cdot\mathbf{n} \end{aligned}$$

are called *coefficients of the second fundamental form.*

2.1.4 Curvatures

The curvatures of a smooth curve $\mathbf{c}(s)$ are the local properties of its shape, invariant under euclidean motions. The only first-order information is the tangent line; since all lines in space are equivalent, there are no first-order invariants. Second-order information (again, independent of parameterization) is given by the osculating circle; the corresponding invariant is its curvature $\kappa = \frac{1}{r}$.

Definition 2.19 (Center of Curvature)**.** The *center of curvature* of a curve is found at a point that is at a distance equal to the radius of curvature lying on the normal vector. It is the point at infinity if the curvature is zero. The osculating circle to the curve is centered at the center of curvature.

A plane curve is completely determined (up to rigid motion) by its (signed) curvature $\kappa(s)$ as a function of arc length s. For a space curve, however, we need to look at the third-order invariants; these are the torsion τ and the derivative κ', but the latter of course gives no new information. Curvature and torsion now form a complete set of invariants: a space curve is determined by $\kappa(s)$ and $\tau(s)$ [24].

Definition 2.20 (Normal Curvature)**.** The *normal curvature* κ_n is the curvature of the planar curve that results from intersecting the surface S with the plane through $\mathbf{p}$ spanned by **n** and **t**. We distinguish between the normal curvature of a curve with arc length parameterization

$$\kappa_n = \mathbf{c}'' \cdot \mathbf{n} = (\mathbf{x}_{uu} \cdot \mathbf{n})\, u'^2 + 2\,(\mathbf{x}_{uv} \cdot \mathbf{n})\, u'\, v' + (\mathbf{x}_{vv} \cdot \mathbf{n})\, v'^2,$$

and the normal curvature of a curve with arbitrary parametrization

$$\kappa_n = \frac{L\,\dot{u}^2 + 2M\,\dot{u}\,\dot{v} + N\,\dot{v}^2}{E\,\dot{u}^2 + 2F\,\dot{u}\,\dot{v} + G\,\dot{v}^2},$$

where the coefficients of the fundamental forms are used.

Definition 2.21 (Principal Curvatures and Principal Direction)**.** The minimal normal curvature κ_1 and the maximal normal curvature κ_2 are called *principal curvatures*. The associated tangent vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are called *principal directions* and are always perpendicular to each other.

Theorem 2.1 (Euler's Theorem)**.** *The normal curvature can also be written as*

$$\kappa_n = \kappa_1\, \cos^2\phi + \kappa_2\, \sin^2\phi,$$

where ϕ is the angle between the unit tangent vector $\mathbf{t}$ *and the principal direction* $\mathbf{e}_1$.

Proof. Since $\mathbf{e}_1, \mathbf{e}_2$ form an orthonormal basis of the tangent space, we may write

$$\mathbf{t} = \langle \mathbf{t}, \mathbf{e}_1 \rangle \mathbf{e}_1 + \langle \mathbf{t}, \mathbf{e}_2 \rangle \mathbf{e}_2 = \mathbf{e}_1\, \cos\phi + \mathbf{e}_2 \sin\phi.$$

The normal curvature along **t** is given by

$$\begin{aligned}
\kappa_n &= \mathbf{II}_{\mathbf{p}}(\mathbf{t}) = -\langle d\mathbf{n}_{\mathbf{p}}(\mathbf{t}), \mathbf{t} \rangle \\
&= -\langle d\mathbf{n}_{\mathbf{p}}(\mathbf{e}_1\, \cos\phi + \mathbf{e}_2\, \sin\phi), \mathbf{e}_1\, \cos\phi + \mathbf{e}_2 \sin\phi \rangle \\
&= \langle \mathbf{e}_1\, \kappa_1\, \cos\phi + \mathbf{e}_2\, \kappa_2\, \sin\phi, \mathbf{e}_1\, \cos\phi + \mathbf{e}_2\, \sin\phi \rangle \\
&= \kappa_1\, \cos^2\phi + \kappa_2\, \sin^2\phi.
\end{aligned}$$

□

Using Euler's theorem, we can express the normal curvature κ_n for a direction **t** by the principal curvatures κ_1 and κ_2 and the principal curvature directions $\mathbf{e}_1$ and $\mathbf{e}_2$:

$$\kappa_n = \langle \mathbf{e}_1\, \kappa_1\, \cos\phi + \mathbf{e}_2\, \kappa_2 \sin\phi, \mathbf{e}_1 \cos\phi + \mathbf{e}_2 \sin\phi \rangle.$$

Definition 2.22 (Mean Curvature)**.** The *mean curvature* H is the average of the principal curvatures, i.e.,

$$H = \frac{\kappa_1 + \kappa_2}{2} = \frac{E\,N - 2\,F\,M + G\,L}{2(E\,G - F^2)}$$

The mean curvature is certainly not intrinsic [23]. Note that the sign of H depends on the choice of the unit normal $\mathbf{n}$, so often it is more natural to work with the *vector mean curvature* (or *mean curvature vector*) $\mathbf{H} := H \cdot \mathbf{n}$.

Definition 2.23 (Gaussian Curvature)**.** The *Gaussian curvature* K is defined as the product of the principal curvatures, i.e.,

$$K = \kappa_1 \kappa_2 = \frac{L\,N - M^2}{E\,G - F^2}.$$

The Gaussian curvature is invariant under local isometries and as such also intrinsic to the surface.

Definition 2.24 (Total Curvature)**.** The *total curvature* of a curve is $\int \kappa\, ds$ and the total curvature of a surface is $\int \kappa_1^2 + \kappa_2^2$.

For plane curves, we can consider instead the signed curvature, and find that $\int \kappa\, ds$ is always an integral multiple of 2π.

2.1.5 Laplace-Operator

Definition 2.25 (Laplace Operator)**.** The *Laplace operator* is a second-order differential operator in the n-dimensional Euclidean space $\mathbb{E}^n$, defined as the divergence (∇) of the gradient (∇f). If f is a twice-differentiable real-valued function, then the Laplacian of f is defined by

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f.$$

Equivalently, the Laplacian of f is the sum of all the unmixed second partial derivatives in the Cartesian coordinates $\mathbf{x}_i$:

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial \mathbf{x}_i^2}.$$

As a second-order differential operator, the Laplace operator maps C^k-functions to C^{k-2}-functions for $k \geq 2$.

Definition 2.26 (Laplace-Beltrami Operator)**.** The Laplace operator can be generalized to operate on functions defined on surfaces (or more generally on Riemannian manifolds). This more general operator goes by the name *Laplace-Beltrami operator.* As the Laplacian, the Laplace-Beltrami operator is defined as the divergence of the gradient.

Just as the curvature κ is the geometric version of the second derivative for curves, mean curvature H is the geometric version of the Laplacian Δ. Indeed, if a surface S is written locally near a point $\mathbf{p}$ as the graph of a height function f over its tangent plane $T_{\mathbf{p}}S$, then $H(\mathbf{p}) = \Delta f$. Alternatively, we can write $\mathbf{H} = \Delta_B\, \mathbf{x}$, where $\mathbf{x}$ is the position vector in $\mathbb{E}^3$ and Δ_B is the Laplace-Beltrami operator, the intrinsic surface Laplacian.

2.2 Basics of Discrete Differential Geometry

Since polygonal meshes are piecewise linear surfaces, the concepts introduced before cannot be applied directly. The following definitions of discrete differential operators are thus based on the assumption that meshes can be interpreted as piecewise linear approximations of smooth surfaces. The goal is to compute approximations of the differential properties of the underlying surface directly from the mesh data.

Definition 2.27 (Polygon)**.** A polygon $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ with vertices $\mathbf{p}_i \in \mathbb{R}^d$ can be seen as a discrete curve. The *length* (number of vertices) of a polygon with vertices $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ is n. A polygon is either thought to be *closed* with $\mathbf{p}_{n+1} = \mathbf{p}_1$, $\mathbf{p}_{n+2} = \mathbf{p}_2, \ldots$ or *open*.

Definition 2.28 (Knot, Connectivity)**.** A *knot* is a vertex that is shared by more than one polygon. The *connectivity* of a polygon network is given by the polygons and the knots. Note that in contrast to the curve network case, we do not consider incoming and outcoming curves at a knot, and each polygon is a structure line.

Definition 2.29 (Neighborhood Structure)**.** Both for polygons and polygon networks, a *neighborhood* $N(\mathbf{v}_i)$ of a vertex $\mathbf{v}_i$ is a set of vertices. If the vertex $\mathbf{v}_j$ belongs to the neighborhood $\mathbf{v}_i$, we say that $\mathbf{v}_j$ is a *neighbor* of $\mathbf{v}_i$. The *neighborhood structure* of a polygonal curve or polyhedral surface is the family of all its neighborhoods $\{N(\mathbf{v}_i) : i = 1, \ldots, n\}$. A particularly important neighborhood structure is the *first-order neighborhood structure* $N_1(\mathbf{v}_i)$, where for each pair of vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ that share a face (edge for a curve), we make $\mathbf{v}_j$ a neighbor of $\mathbf{v}_i$, and $\mathbf{v}_i$ a neighbor of $\mathbf{v}_j$. For example, for a polygonal curve represented as a list of consecutive vertices, the first-order neighborhood of a vertex $\mathbf{v}_i$ is $N(\mathbf{v}_i) = \{\mathbf{v}_{i-1}, \mathbf{v}_{i+1}\}$.

Definition 2.30 (Symmetric Neighborhood Structure)**.** A neighborhood structure is *symmetric* if every time that a vertex $\mathbf{v}_j$ is a neighbor of vertex $\mathbf{v}_i$, also $\mathbf{v}_i$ is a neighbor of $\mathbf{v}_j$.

With non-symmetric neighborhoods certain constraints can be imposed[5].

Definition 2.31 (Difference Operator)**.** The *difference polygon* $\Delta\mathbf{p}$ consists of the vectors

$$\Delta\mathbf{p}_k = \mathbf{p}_{k+1} - \mathbf{p}_k. \tag{1}$$

By iteration, we construct higher difference polygons $\Delta^2\mathbf{p}, \Delta^3\mathbf{p}$ that consist of vectors

$$\begin{aligned}\Delta^2\mathbf{p}_k &= \Delta(\Delta\mathbf{p}_k) = \mathbf{p}_{k+2} - 2\mathbf{p}_{k+1} + \mathbf{p}_k \\ \Delta^3\mathbf{p}_k &= \Delta(\Delta^2\mathbf{p}_k) = \mathbf{p}_{k+3} - 3\mathbf{p}_{k+2} + 3\mathbf{p}_{k+1} - \mathbf{p}_k,\end{aligned}$$

---

[5] An example for non-symmetric neighborhood can be the fixing of certain vertices. If the neighborhood of $\mathbf{v}_i$ is empty, then the vertex $\mathbf{v}_i$ does not change during the fairing process, because the discrete Laplacian $\Delta\mathbf{v}_i$ is equal to zero by definition of empty sum.

and so on. We use the difference of successive points as a discrete first derivative, and the difference of such differences as a discrete second derivative. If the polygon $\mathbf{p}$ is open, the length of the difference polygon $\Delta\mathbf{p} = (\Delta\mathbf{p}_1, \dots, \Delta\mathbf{p}_{n-1})$ is $n-1$. If $\mathbf{p}$ is closed, indices are taken $mod\, n$ and the length of the difference polygon $\Delta\mathbf{p} = (\Delta\mathbf{p}_1, \dots, \Delta\mathbf{p}_n)$ is n.

2.2.1 Laplace Operator

Definition 2.32 (Discrete Laplace Operator)**.** The *discrete Laplace operator* on a piecewise linear surface, i.e., a triangle mesh, is expressed as

$$\Delta\mathbf{p}_i = \sum_{\mathbf{p}_j \in N_1(\mathbf{p}_i)} \omega_{ij}(\mathbf{p}_j - \mathbf{p}_i),$$

where for all vertices $\mathbf{p}_i$ weights are normalized such that

$$\sum_{\mathbf{p}_j \in N_1(\mathbf{p}_i)} \omega_{ij} = 1.$$

(Note that normalization and symmetry are not generally necessary for smoothing. In contrast, possibly required area terms destroy these properties.) We can now write the discrete Laplace operator as a matrix $\mathbf{L}$ with non-zero entries

$$\mathbf{L} = \begin{cases} -1, & i = j \\ \omega_{ij}, & \mathbf{p}_j \in N_1(\mathbf{p}_i) \end{cases}$$

$\mathbf{L}$ is generally sparse, the number of non-zeros in each row is one plus the valence of the associated vertex. For the uniform discretization Δ_{uni} we choose weights $\omega_{ij} = \frac{1}{|N_1(\mathbf{p}_i)|}$, i.e., the Laplacian depends only on the mesh connectivity. Then $\mathbf{L}$ is symmetric and has real eigenvalues and eigenvectors.

The eigenvectors of $\mathbf{L}$ form an orthogonal basis of $\mathbb{R}^n$, where n denotes the number of vertices, and the associated eigenvalues are commonly interpreted as frequencies. The projections of the coordinates $p_x, p_y, p_z \in \mathbb{R}^n$ into this basis is called spectrum of the geometry. Given eigenvectors $\mathbf{e}_i$, the x-components p_x of the mesh geometry can now be expressed as

$$p_x = \sum_{i=1}^{n} \alpha_i^x \mathbf{e}_i,$$

where the coefficients $\alpha_i^x = \mathbf{e}_i^T \mathbf{p}$, and similar for p_y, p_z. It shows that the eigenvectors associated with the first eigenvalues $0 \leq \lambda_1 \leq \ldots \leq \lambda_n$ correspond to low-frequency components: in other words, cancelling coefficients α_i associated with high-frequency components yields a smoothed version of the shape. This is well-known from spectral graph theory: the projection into the linear space spanned by the eigenvectors provides a generalization of the Discrete Fourier Transform.

This can also be seen immediately for the discrete univariate setting. Here, the decomposition is equivalent to the discrete cosine transform. For general surface meshes, their spectral decomposition defines a natural frequency domain. Taubin [27,28] uses this fact to motivate geometric signal processing and to define low-pass filters for smoothing meshes [3].

The discrete Laplace operator can be seen as a discrete approximation of the following curvilinear integral

$$\frac{1}{|\gamma|}\int_{\mathbf{v}\in\gamma}(\mathbf{v}-\mathbf{v}_i)dl(\mathbf{v}),$$

where γ is a closed curve embedded in the surface which encircles the vertex $\mathbf{v}_i$, and $|\gamma|$ is the length of the curve. It is known that, for a curvature continuous surface, if the curve γ is let to shrink to the point $\mathbf{v}_i$, the integral converges to the mean curvature H of the surface at the point $\mathbf{v}_i$ times the surface normal vector $\mathbf{n}_i$ at the same point

$$\lim_{\epsilon\to 0}\frac{1}{|\gamma_\epsilon|}\int_{\mathbf{v}\in\gamma_\epsilon}(\mathbf{v}-\mathbf{v}_i)dl(\mathbf{v}) = H\,\mathbf{n}_i.$$

The expression on the right hand side is the *curvature normal.* It follows that the length of the Laplacian vector is equal to the product of the average edge length times the mean curvature

$$\Delta\mathbf{v}_i = \Bigg(\sum_{\mathbf{v}_j\in N_1(\mathbf{v}_i)}\omega_{ij}\|(\mathbf{v}_j-\mathbf{v}_i)\|\Bigg)H\mathbf{n}_i,$$

which can be used as a definition of discrete mean curvature [28].

2.2.2 Laplace-Beltrami Operator

In general, the Laplace operator is defined as the divergence of the gradient, i.e. $\Delta = \nabla^2 = \nabla\cdot\nabla$. In Euclidean space this second-order differential operator can be written as the sum of second partial derivatives

$$\Delta f = div\nabla f = \sum_i \frac{\partial^2 f}{\partial x_i^2}$$

with Cartesian coordinates x_i. The Laplace-Beltrami operator Δ_B extends this concept to functions defined on surfaces. For a given function f defined on a manifold surface S the Laplace-Beltrami operator is defined as

$$\Delta_B f = div\nabla f,$$

which requires a suitable definition of the divergence and gradient operators on manifolds. A discretization of the Laplace-Beltrami operator is

$$\Delta_B\,\mathbf{v} = \frac{2}{A_{Voronoi}(\mathbf{v})}\sum_{\mathbf{v}_i\in N_1(\mathbf{v})}(\cot\alpha_i+\cot\beta_i)(\mathbf{v}_i-\mathbf{v}),$$

where $\alpha_i = \angle(\mathbf{v}, \mathbf{v}_{i-1}, \mathbf{v}_i)$, $\beta_i = \angle(\mathbf{v}, \mathbf{v}_{i+1}, \mathbf{v}_i)$ and $A_{Voronoi}(\mathbf{v})$ denotes the Voronoi area

$$A_{Voronoi}(\mathbf{v}) = \frac{1}{8} \sum_{\mathbf{v}_i \in N(\mathbf{v})} (\cot \alpha_i + \cot \beta_i) \|\mathbf{v} - \mathbf{v}_i\|^2,$$

around the vertex $\mathbf{v}$ as shown in Figure 2.2.

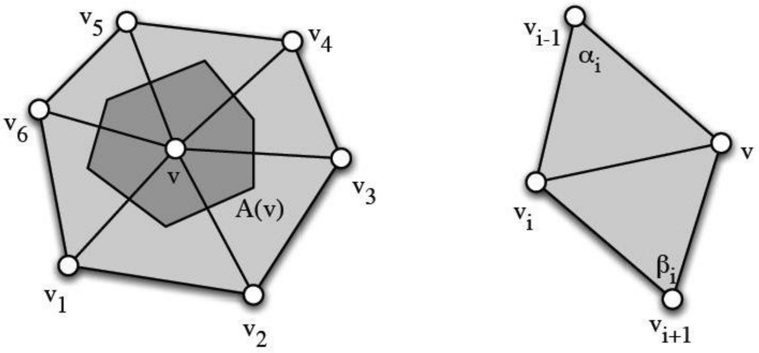


Figure 2.2: The Laplace-Beltrami operator Δ_B is computed by a linear combination of its vertex $\mathbf{v}$ and those of its one-ring neighbors $\mathbf{v}_i$. The corresponding weights are given by the cotangent values of α_i and β_i and the Voronoi area $A(\mathbf{v})$.

Applied to the coordinate function $\mathbf{x}$ of the surface the Laplace-Beltrami operator evaluates to the mean curvature normal

$$\Delta_B \mathbf{x} = -2H\mathbf{n}.$$

Note that the Laplace-Beltrami operator is an intrinsic property that only depends on the metric tensor of the surface and is thus independent of a specific parameterization.

2.2.3 Discrete Curvature

Definition 2.33 (Discrete Curvature)**.** The *discrete curvature* at a point $\mathbf{p}_i$ is defined to be the reciprocal value of the circle radius interpolating the points $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$ or 0 if the points lie on a straight line, which leads to the well-known formula

$$\kappa_i = 2 \frac{det(\mathbf{p}_i - \mathbf{p}_{i-1}, \mathbf{p}_{i+1} - \mathbf{p}_i)}{\|\mathbf{p}_i - \mathbf{p}_{i-1}\| \, \|\mathbf{p}_{i+1} - \mathbf{p}_i\| \, \|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|}.$$

Definition 2.34 (Discrete Normal Curvature)**.** Given an edge $(\mathbf{v}_i, \mathbf{v}_j)$, vertex positions $\mathbf{p}_i, \mathbf{p}_j$ and the normal $\mathbf{n}_i$,

$$\kappa_n(\mathbf{p}_i) = 2 \, \frac{(\mathbf{p}_j - \mathbf{p}_i)\mathbf{n}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}$$

provides an approximation of the *normal curvature* at $\mathbf{p}_i$ in the tangent direction that results from projecting $\mathbf{p}_i$ and $\mathbf{p}_j$ into the tangent plane defined by $\mathbf{n}_i$.

The Discrete Mean Curvature

The mean curvature of a discrete surface is supported along the edges. A common model for integrated *discrete mean curvature* associated with an edge e is given by the product of dihedral angle and edge length,

$$H(e) = (\pi - \theta_e) \cdot \|e\|.$$

A similar model was used by replacing $(\pi - \theta_e)$ by $\cos(\theta_e/2)$. Obviously, these two versions agree (up to a factor of two) in the limit of small angles between normals ($\theta_e \to \pi$) [2]. A linear model for the *mean curvature normal* is

$$\mathbf{H}(e) = -2\cos\frac{\theta_e}{2} \cdot \|e\| \cdot \mathbf{n}_e,$$

where $\mathbf{n}_e$ is the (angle-bisecting) normal vector to e of unit length. Note that unlike the scalar-valued mean curvature H, $\mathbf{H}$ is a vector, and the norm of $\mathbf{H}$ corresponds to the (scalar) *mean curvature.* This version is linear in vertex positions of the mesh (cf. [2]).

Definition 2.35 (Discrete Mean Curvature)**.** Vertex-centered discrete mean curvatures are obtained by choosing one of the above edge-based models and summing the contributions from all edge-based curvatures of incident edges,

$$\mathbf{H}(\mathbf{p}) = \frac{1}{2} \sum_{e:\mathbf{p}\in e} \mathbf{H}(e).$$

The factor $\frac{1}{2}$ is due to the fact that we treat integrated quantities here, and each triangle in the vertex star of a vertex $\mathbf{p}$ is seen by exactly two edges.

In [16] the mean curvature H is computed as

$$H = \frac{1}{2}\|\mathbf{H}(\mathbf{x}_i)\|,$$

where $\mathbf{H}(\mathbf{x}_i)$ is defined as

$$\mathbf{H}(\mathbf{x}_i) = \frac{1}{2\,A_{Mixed}} \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} (\cot\alpha_{ij} + \cot\beta_{ij})(\mathbf{x}_i - \mathbf{x}_j).$$

The surface area A_{Mixed} for each vertex $\mathbf{x}$ is defined as the sum of each triangle T from the 1-ring neighborhood of $\mathbf{x}$, whereas the areas of the triangle T are computed as follows: if

T is non-obtuse, the area is the Voronoi area, or either the area $\frac{area(T)}{4}$ for an obtuse angle of T at $\mathbf{x}$ is added or $\frac{area(T)}{2}$ for an non-obtuse angle of T at $\mathbf{x}$.

The Discrete Gaussian Curvature

Let M be a triangulation of smooth surface S in $\mathbb{R}^3$. For a vertex $\mathbf{p}$ of M, suppose $N(\mathbf{p}_i)$ is the set of the 1-ring neighbor vertices of $\mathbf{p}$. The set $\{\mathbf{p_i p p_{i+1}}\}$, $(i = 1, \dots, n)$ of n Euclidean triangles forms a piecewise linear approximation of S around $\mathbf{p}$. Let γ_i denote the angle $\angle \mathbf{p_i p p_{i+1}}$ and the angular defect at $\mathbf{p}$ be $2\pi - \sum_i \gamma_i$.

A popular discrete scheme[6] for computing a discrete Gaussian curvature G is in the form of $\frac{2\pi - \sum_i \gamma_i}{E}$, where E is a geometry quantity. In general, one selects E as $A(\mathbf{p})$ and obtains the following approximation

$$G^{(1)} = \frac{3\,(2\pi - \sum_i \gamma_i)}{A(\mathbf{p})},$$

where $A(\mathbf{p})$ is the sum of the areas of triangles $\mathbf{p_i p p_{i+1}}$.
It is impossible to construct a discrete scheme which is convergent for any discrete mesh, but we know that the discrete scheme $G^{(1)}$ has quadratic convergence rate if the mesh satisfies the so-called *parallelogram criterion*, which requires valence 6 and for non-uniform data, the discrete scheme $G^{(1)}$ is not always convergent to true Gaussian curvature.
Another possibility for discrete Gaussian Curvature is

$$G^{(2)} = \frac{2\pi - \sum_i \gamma_i}{\frac{1}{2}A(\mathbf{p}) - \frac{1}{8}\sum_i \cot(\gamma_i) d_i^2},$$

where d_i is the length of edges $\mathbf{p}_i\mathbf{p}_{i+1}$. As stated before, the previous discrete schemes, including $G^{(1)}$ and $G^{(2)}$, only converge at the regular vertex with valence 6.

It should be pointed out that there is another discrete scheme

$$G^{(3)} = \frac{2\pi - \sum_i \gamma_i}{A_{Voronoi}(\mathbf{p})},$$

where $A_{Voronoi}(\mathbf{p})$ is the area of Voronoi region. Since $A(\mathbf{p})$ could be approximated by $3\,A_{Voronoi}(\mathbf{p})$ (under some conditions) $G^{(3)}$ is easily derived from $G^{(1)}$. In [16] the Gaussian curvature is calculated as

$$G^{(4)} = \frac{2\pi - \sum_i \gamma_i}{A_{Mixed}(\mathbf{p})}.$$

For further information about the discrete Gaussian curvature schemes and their convergence we refer to [32].

---

[6] For further information we refer to [32].

The Discrete Principle Curvature

We have seen in Section 2.1.4 that the mean and Gaussian curvatures are easy to express in terms of the two principal curvatures κ_1 and κ_2. Therefore, since both H and G have been derived for triangulated surfaces, we can define the discrete principal curvatures as:

$$\begin{aligned}\kappa_1 &= H(\mathbf{x}_i) + \sqrt{H(\mathbf{x}_i)^2 - G(\mathbf{x}_i)} \\ \kappa_2 &= H(\mathbf{x}_i) - \sqrt{H(\mathbf{x}_i)^2 - G(\mathbf{x}_i)}.\end{aligned}$$

Unlike the continuous case, where $H^2 - G$ is always positive, we must make sure that H^2 is always larger than G to avoid any numerical problems, and threshold $H^2 - G$ to zero if it is not the case (an extremely rare occurrence).

3 Presentation of the well-known Smoothing and Fairing Methods

One of the main problems in geometric modeling is the generation of aesthetically appealing surfaces. Usually these surfaces are subject to technical requirements like smoothing or interpolation constraints. While the constraints can easily be expressed in mathematical terms (and thus are compatible to the mathematical description of the surface itself), the explicit formulation of 'well-shaped-ness' causes some difficulties. A surface may be smooth in a mathematical sense but still unsatisfactory from an aesthetical point of view. While the former denotes the continuous differentiability (C^k) of a surface, the latter is an aesthetic measure of 'well-shaped-ness' and therefore in technical terms more difficult to define than smoothness. The process of designing a high-quality surface is called *fairing*.

Motivated by physical models of elastic membranes or thin plates, the variational approach to surface design measures the 'bad-shapedness' of a surface by the value of some (bending) energy functionals. In the referring literature, most techniques use constrained energy minimization. The functionals are typically formulated in terms of intrinsic shape properties, i.e. quantities that do not depend on the particular parameterization (or triangulation in the discrete setting), such as curvatures [3].

This section provides an overview of fairing methods used to evaluate the quality of meshes. First of all, a variety of energy functionals and their possible constraints are presented. Secondly, the methods of Laplace and Taubin, which use Fourier Analysis for their approaches, are described. Furthermore, a Spring Model approach which moves the points of an undesired shape along the normal line is presented. Finally, the algorithm of Schneider and Kobbelt to create a fair discrete surface by solving a differential equation is provided.

3.1 Minimization of the Bending Energy

The first smoothing procedures are based on minimizing an energy functional of a polygon network, described in [11], [30], [10] and [29]. The classical energy functionals are quadratic, and after discretization we obtain quadratic functions, which makes minimization easy [10].
At first we define the energy of a single curve, and then the energy of the curve network as the sum of energies of all the curves that contribute to the curve network. Since polygons and polygon networks are the discrete analogues of curves and curve networks, their energies are the discretized ones of the curves and curve networks.

3.1.1 Energy of Polygons and Polygon Networks

Derivation of Curves and Curve Networks

For a smooth curve $c(t)$, defined in some parameter interval $[a, b]$, the energy which minimizes the L^2-norm of the first derivative is defined by

$$E_1(c) = \int_a^b \|c'(t)\|^2 dt. \tag{2}$$

Minimizing the L^1- or L^2-norm of the first derivative of curves is a classical problem of Differential Geometry and leads to the geodesic lines of surfaces [10]. The *linearized bending energy* or *cubic spline energy* is defined by

$$E_2(c) = \int_a^b \|c''(t)\|^2 dt. \tag{3}$$

Minimizing a linear combination of energies (2) and (3) leads to the well-known *tension energy*

$$E_\tau(c) = E_2 + \tau E_1, \tag{4}$$

where τ is a tension parameter. The parameter τ controls the tension of the curve. In the case where a curve is restricted to a surface, increasing the tension parameter τ moves the curve closer to the surface [10]. As mentioned above, the energy of the entire curve network $\mathcal{C}$ is the sum of energies of all single curves of the curve network:

$$E(\mathcal{C}) = \sum_{c \in \mathcal{C}} E(c).$$

Here, E is either E_1, E_2 or E_τ. We refer to energy minimizing curve networks also as *fair curve network* or short: *fair webs* [30].
An unconstrained E_2-minimizing curve network turns out to consist of C^2 cubic splines. The usually unique E-minimal curve network in the unconstrained case is found as a solution of a system of linear equations [29].

Constraints on Fair Webs

Constraints imposed on the smoothing process include the sustainment of linear and nonlinear surface features such as sharp edges, corners, or non-planar curves. The hard error (or accuracy) bounds are represented as tolerance cylinders, where the surface is permitted to move. A constraint is hard, when the error is not guaranteed to be below the allowed bounds. The smoothing is accomplished by means of an energy minimization technique constrained to keep the surface inside the tolerance bounds [11].

Fair webs constrained to surfaces have a number of nice properties [30]. The most important one from the fairness point of view is, that for E_2- and E_τ-minimizing fair webs, a structure line is actually C^2. A discretized network has analogous properties. We consider fair webs whose curves are constrained to a given surface F. Although we assume that the connectivity of the fair web is maintained, we have the freedom to choose which knot points shall be fixed, and which knot points shall be free[7].

Consideration of Polygons and Polygon Networks

A polygon $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ with vertices $\mathbf{p}_i \in \mathbb{R}^d$ can be seen as discrete curve. The *length* (number of vertices) of a polygon with vertices $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ is n.

By replacing integration by summation, any of the functionals E_1, E_2 and E_τ is thus converted into a quadratic function, and we define the discrete counterparts of the energies from equation (2)-(4) by

$$\begin{aligned} E_1(\mathbf{p}) &= \sum_{i=1}^{n} \|\Delta \mathbf{p}_i\|^2, & (5)\\ E_2(\mathbf{p}) &= \sum_{i=1}^{n} \|\Delta^2 \mathbf{p}_i\|^2, & (6)\\ E_\tau(\mathbf{p}) &= \sum_{i=1}^{n} (\|\Delta^2 \mathbf{p}_i\|^2 + \tau \|\Delta \mathbf{p}_i\|^2), & (7) \end{aligned}$$

where $\Delta \mathbf{p}_i$ is the difference polygon defined in (1).
The gradients ∇E_1, ∇E_2 and ∇E_τ of these energy functions are given by

$$\begin{aligned} \nabla E_1(\mathbf{p}) &= -2\Delta^2 \mathbf{p}, & (8)\\ \nabla E_2(\mathbf{p}) &= 2\Delta^4 \mathbf{p}, & (9)\\ \nabla E_\tau(\mathbf{p}) &= 2(\Delta^4 \mathbf{p} - \tau \Delta^2 \mathbf{p}). & (10) \end{aligned}$$

An energy gradient $\nabla E(\mathbf{p})$ of equations (8), (9) and (10) may be visualized as a sequence of vectors $(\nabla E(\mathbf{p}))_k$ attached to the vertices $\mathbf{p}_k$ of the polygon.

If we replace the set of curves $\mathcal{C}$ by a set of polygons $\mathcal{P}$ we get a *polygon network* [29]. Polygon networks have energies which are defined as the sum of the energies of the polygons they are made of. Therefore, we can define the energy of a polygon network to be the sum of energies of the n different polygons defined by $j = const$ and the m different polygons defined by $i = const$:

$$E(\mathcal{P}) = \sum_{j=1}^{n} \sum_{i=2}^{m-1} \|\mathbf{p}_{i-1,j} - 2\mathbf{p}_{i,j} + \mathbf{p}_{i+1,j}\|^2 + \sum_{i=1}^{m} \sum_{j=2}^{m-1} \|\mathbf{p}_{i,j-1} - 2\mathbf{p}_{i,j} + \mathbf{p}_{i,j+1}\|^2. \qquad (11)$$

---

[7] Knots are called *fixed*, if their position is chosen, eg. by the user. And knots are called *free*, if the location is not fixed as a side condition. Their position will be determined by the energy minimization procedure.

Constraints on Polygon Networks

A *fair polygon network* is one which has minimal energy among all networks which fulfill a fixed set of constraints. Fair polygon networks are well suited to compute visually pleasing meshes in the sense that they are formed by sequences of fair discretized curves. By fixing certain knots, the designer can influence the creation of a fair mesh. They may also be applied to surface parameterization by mapping chosen lines of the parameter domain to fair curves on a surface. For instance, it is easy to wind a visually pleasing textured band around an object. Further, fair polygon networks are useful for the design of fair surfaces which avoid given obstacles [29].

In order to express planarity of a quad face $Q_{i,j}$, we consider the four angles $\phi^1_{i,j}, \dots, \phi^4_{i,j}$ enclosed by the edges of $Q_{i,j}$, measured in the interval $[0, \pi]$. It is known that $Q_{i,j}$ is planar and convex if and only if these angles sum up to 2π. We use the notation

$$c_{pq,i,j} := \phi^1_{i,j} + \ldots + \phi^4_{i,j} - 2\pi = 0.$$

In [11] another possible constraint to allow each point a magnitude of deviation was mentioned. For that approach, each point $\mathbf{p}_{ij}$ has to be equipped with its own *tolerance cylinder* Z_{ij}. That means that every point has a horizontal deviation bounded by r_{ij} and a vertical deviation bounded by h_{ij}, so that the point is within a cylinder Z_{ij} of diameter $2r_{ij}$ and height $2h_{ij}$, which is centered in the given point $\mathbf{p}_{ij} = (x_{ij}, y_{ij}, z_{ij})$. The ultimate goal is to move the points $\mathbf{p}_{ij}$ in a way that makes the energy (11) smaller, but the surface still passes through the cylinders Z_{ij} [8].

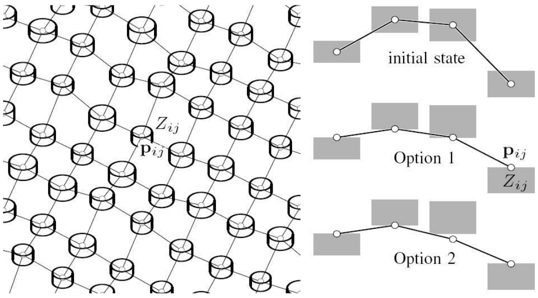


Figure 3.1: Left: Polyline network with the tolerance cylinders Z_{ij} associated with the vertices $\mathbf{p}_{ij}$. Right: At top, initial state before optimization. At center, Option 1 of optimization (all vertices stay inside the tolerance cylinders). At the bottom, Option 2 in which the vertices are allowed to leave the cylinders, provided the surface still passes through them.

---

[8] This condition is stricter than actually necessary. It is certainly sufficient that for each vertex $\mathbf{p}_{ij}$, one of the two polylines meeting there meets Z_{ij}.

Minimizing the quadratic function (11) of the variables x_{ij}, y_{ij}, z_{ij}, subject to the constraints mentioned above, is a quadratic optimization problem with convex condition - in case of the first condition, that $\mathbf{p}_{ij}$ remains inside Z_{ij} - and non-convex side condition otherwise. The convex optimization problem has a unique solution. The difference between the first and second option is not very big, as illustrated in [11], for which reason the second procedure is "convex enough" so that we do not expect a direct minimization procedure getting stuck in a local minimum. Minimizing the quadratic functions (11) of the variables (x_{ij}, y_{ij}, z_{ij}) contains too many variables for just submitting it to a generic optimization procedure. The quality, that the energy would be zero, if computed with x and y coordinates of the initial data alone, allows a more direct approach. Therefore we do not expect the points $\mathbf{p}_{ij}$ to move very much in x and y direction during optimization. Indeed, this is confirmed by numerical experiments. We only use the z coordinates of the data points as variables for minimization:

$$E = \sum_{j=1}^{n} \sum_{i=2}^{m-1} \|z_{i-1,j} - 2z_{i,j} + z_{i+1,j}\|^2 + \sum_{i=1}^{m} \sum_{j=2}^{m-1} \|z_{i,j-1} - 2z_{i,j} + z_{i,j+1}\|^2.$$

Because of the condition, that $\mathbf{p}_{ij}$ remaining inside Z_{ij} leads to a convex optimization problem with unique solution, we employ a gradient descent method with the original data as initial condition. It is elementary that the gradient of the energy is given by

$$(\nabla E)_{ij} = (z_{i-2,j} + z_{i+2,j} + z_{i,j-2} + z_{i,j+2}) - 4(z_{i,j-1} + z_{i+1,j} + z_{i,j-1} + z_{i,j+1}) + 12z_{i,j},$$

provided $i, j > 2$ and $i < m-1$, $j < n-1$. For implementation of the optimization see [11].

The algorithm is capable of smoothing the data with tolerance cylinders of different sizes. These flexible tolerances have two particular advantages: first, we can preserve features present in the data by reducing the size of the tolerance cylinders in feature areas, and secondly the algorithm can be used to fill holes present in the original data during the smoothing process.

3.1.2 Energy of the Curvature

A common method to guarantee excellent surface fairness is to minimize fairness functionals based on geometric invariants [20]. In the case of the fairness of a curve $c(t)$, the minimization of the arc length integral of the squared magnitude of curvature is traditionally:

$$E(c) = \int \kappa^2 \, dt.$$

One of the best-known functionals in the mentioned category and most frequent ones is the total curvature of a surface S [21]:

$$E(S) = \int_S \kappa_1^2 + \kappa_2^2 \, dS. \tag{12}$$

Here κ_1 and κ_2 are the principal curvature, which non-linearly depend on the surface S. E is a geometric quantity whose definition is independent of parametrization. Therefore, this formulation does not create shape artifacts related to an underlying fixed surface parameterization.

For closed surfaces, it turns out that minimizing bending energy is equivalent to minimizing mean curvature, since the area integral of Gaussian curvature, $G = \kappa_1\kappa_2$, is a topological constant that depends only on the genus of the surface. Using mean curvature H as the energy functional is also known as *Willmore energy* [22]:

$$E_W(S) = \int_S (H^2 - G)dA = \frac{1}{4}\int_S (\kappa_1 - \kappa_2)^2 dA.$$

Recall from the differential geometry of surfaces that information about the curvature of a surface at a point is given by the second fundamental form. The normal section curvature of a surface $S(u,v)$ in the direction of a parametric unit tangent $\mathbf{t}$ is given by $\kappa = II(\mathbf{t},\mathbf{t})$, where

$$II(\mathbf{t},\mathbf{t}) = \mathbf{t}^T \begin{bmatrix} \frac{\partial^2 S}{\partial u^2}\cdot\mathbf{n} & \frac{\partial^2 S}{\partial u \partial v}\cdot\mathbf{n} \\ \frac{\partial^2 S}{\partial v \partial u}\cdot\mathbf{n} & \frac{\partial^2 S}{\partial^2 v}\cdot\mathbf{n} \end{bmatrix} \mathbf{t}$$

and $\mathbf{n}$ is the surface unit normal. It is straightforward to show that the squared Frobenius norm of the matrix is equivalent to $\kappa_1^2 + \kappa_2^2$ [31].

A minimization process based on such functional leads to surfaces of extraordinary quality, but due to the demanding construction process, the required computation time can be enormous. A popular technique to simplify this approach is to give up the parameter independence and approximate the geometric invariants with higher-order derivatives. For some important fairness functionals this results in algorithms that enable the construction of a solution by solving a linear system. A representation of this category is the *thin-plate energy* of a mesh X, which can be used to create surfaces satisfying C^1 boundary conditions:

$$E_{thin\,plate}(X) \;=\; \frac{1}{2}\int_\omega \left(\frac{\partial^2 X}{\partial u^2}\right)^2 + 2\left(\frac{\partial^2 X}{\partial u \partial v}\right)^2 + \left(\frac{\partial^2 X}{\partial v^2}\right)^2 dudv. \tag{13}$$

Note that the thin-plate energy turns out to be equal to the total curvature only when the parametrization (u,v) is isometric.
Many practical fairing methods also prefer the *membrane functional* of a mesh X:

$$E_{membrane}(X) \;=\; \frac{1}{2}\int_\omega \left(\frac{\partial X}{\partial u}\right)^2 + \left(\frac{\partial X}{\partial v}\right)^2 dudv. \tag{14}$$

Celniker and Gossard [4] were able to improve the quality of interpolating surfaces by using a fairness norm based on a linear combination of the energy of a membrane and a

thin plate. The fairness norm is still quadratic, so it can be minimized by solving a linear system. As a result their method is fast enough for interactive design, but their technique is not capable of describing surfaces of arbitrary genus [9].
Instead of minimizing a functional, another approach first applies variational calculus to transform the minimization problem into a problem of solving a differential equation with constraints. For the functional (13) the optimal surfaces can be characterized by the partial differential equation $\Delta^2 X = 0$, where Δ is the Laplace operator, to create surfaces satisfying prescribed C^1 boundary conditions [20]. Similarly, minimizing the membrane energy functional (14) leads to solving $\Delta\, X = 0$ [3].

The Euler-Lagrange equations $\Delta^i X = 0$ associated with minimizers of various fairing functionals show their relation to steady state solutions of diffusion flow (and hence signal processing and low-pass filters). It follows that fairing indeed refers to designing fair surfaces that ideally depend only on the given boundary conditions: for surfaces derived from $\Delta^k X = 0$, boundary constraints of order $\mathcal{C}^{k-1}$ are interpolated. Note that for the solution of the arising linear systems appropriate boundary conditions have to be applied to guarantee the existence of solutions [3].

3.1.3 Energy of the Variation of the Curvature

The problem of creating surfaces with G^1 continuity is very difficult to solve satisfactorily. Most techniques use heuristics to set extra degrees of freedom and sufficient but not necessary constructions to guarantee G^1 continuity; however they typically produce unnecessary and undesirable "wrinkles". Henry P. Moreton and Carlo H. Séquin presented in [17] a method to minimize a fairness functional subject to given geometric constraint using nonlinear optimization techniques. Once the geometric constraints are satisfied by construction, the techniques described in [17] set the remaining surface parameters (degrees of freedom) to minimize the fairness functional while maintaining G^1 continuity using a penalty function. The curve and surface functional minimize the variation of curvature; thus we refer to the curves as *minimum variation curves* (MVC) and to the surfaces as *minimum variation surfaces* (MVS) (cf. [17]). In the case of curves, the integral of the squared magnitude of the derivative of curvature

$$E(c) = \int \frac{d\kappa^2}{dt} dt$$

is minimized. This new functional results in curves with noticeably smoother curvature plots [17]. For surfaces, the functional is the integral of the squared magnitude of the derivatives of normal curvature taken in the principle directions

$$E(S) = \int \left(\frac{d\kappa_n^2}{d\hat{\mathbf{e}}_1} + \frac{d\kappa_n^2}{d\hat{\mathbf{e}}_2} \right) dS.$$

Here κ_n is the normal curvature and $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$ are the corresponding principal curvature directions. The principal curvatures κ_1 and κ_2 are the normal curvatures in the principle directions. Thus the problem of computing $\frac{d\kappa_n}{d\hat{\mathbf{e}}_1}$ and $\frac{d\kappa_n}{d\hat{\mathbf{e}}_2}$ is transformed into a computation of $\frac{d\kappa_1}{d\hat{\mathbf{e}}_1}$ and $\frac{d\kappa_2}{d\hat{\mathbf{e}}_2}$:

$$E(S) = \int \left(\frac{d\kappa_1^2}{d\hat{\mathbf{e}}_1} + \frac{d\kappa_2^2}{d\hat{\mathbf{e}}_2} \right) dS. \tag{15}$$

Moreton and Séquin complete their objective function which is being minimized by adding a penalty for lack of G^1 continuity. They use the gradient descent scheme with an appropriate initial surface and refine that surface until the optimal surface is reached.

The choice of this functional has the advantage that it leads to regular shapes commonly used in geometric modeling. The minimization of the fairness functional also produces very fair free-form surfaces [17]. As for the functional (12) the required computation time for the minimization process of the fairing functional (15) can be enormous. In general, some surface energy is defined that quantifies surface fairness, and curvature is used to express these terms as it is independent of the special parameterization of a surface. A fair surface is then designed by minimizing these energies [3].

Analogously to the previous case of curvature, giving up parameter independence corresponds to solving the sixth-order partial differential equation $\Delta^3 \mathbf{X} = 0$ [3].

3.2 Fourier Analysis on Meshes

Fourier analysis is a natural tool to solve the problem of signal smoothing. By generalizing classical discrete Fourier analysis to two-dimensional discrete surface signals - functions defined on polyhedral surfaces of arbitrary topology - we reduce the problem of surface smoothing, or fairing, to low-pass filtering. The space of signals - functions defined on a certain domain - is decomposed into orthogonal subspaces associated with different frequencies, with the low-frequency content of a signal regarded as subjacent data, and the high-frequency content as noise [27].

The signal-processing approach was motivated by the problem of how to fair large polyhedral surfaces of arbitrary topology. Most existing algorithms based on fairness norm optimization [9, 17, 31] are prohibitively expensive for very large surfaces. Therefore a new algorithms with linear time and space complexity was required. However, the signal processing formulation results in much less expensive computations. In [9, 17, 31], after finite element discretization, the problem is often reduced to the solution of a large sparse linear system, or a more expensive global optimization problem. Large sparse linear systems are solved using iterative methods, and usually result in quadratic time complexity algorithms [27].

The simplest smoothing algorithm that satisfies the linear complexity requirement is Laplacian smoothing, described in detail in section 3.2.2. Laplacian smoothing is an iterative process, where in each step every vertex of the mesh is moved to the barycenter of its neighbors. The only problem with Laplacian smoothing is shrinkage. The algorithm introduced by Taubin solves this problem and imposed the signal processing machinery necessary to analyze the behavior of these smoothing processes [28].

3.2.1 The Method of Fourier Descriptors

The method of Fourier descriptors smoothes a closed curve by removing the noise from the coordinates, i.e. by projecting the coordinate signals onto the subspace of low frequencies. To denoise a signal it is sufficient to compute its discrete Fourier transformation, discard its high frequency coefficients, and compute the linear combination of remaining terms as the result. This is exactly what the method of Fourier descriptors does to smooth a closed curve.

The approach to extend Fourier analysis to signals defined on polyhedral surfaces of arbitrary topology is based on the observation that the classical Fourier transform of a signal can be seen as the decomposition of the signal into a linear combination of the eigenvectors of the Laplace operator. To extend Fourier analysis to surfaces of arbitrary topology we only have to define a new operator that takes the place of the Laplacian.

As an initiation for the Laplacian and Taubin smoothing methods, we consider the classical case of a discrete time n-periodic signal described in [27, 28]. The signal is a function defined on a regular polygon of n vertices, which is represented as a column vector $\mathbf{x} = (x_1, \dots, x_n)^T$. The components of this vector are the values of the signal at the vertices of the polygon [27]. The discrete Laplacian of $\mathbf{x}$ is defined as

$$\Delta \mathbf{x}_i = \sum_{\mathbf{x}_j \in N_1(\mathbf{x}_i)} \omega_{ij}(\mathbf{x}_j - \mathbf{x}_i),$$

where the indices are incremented and decremented $mod\, n$. The weights are non-negative numbers that add up to one for each vertex star

$$\sum_{\mathbf{x}_j \in N_1(\mathbf{x}_i)} \omega_{ij} = 1.$$

The weights can be chosen in many different ways, which is discussed at the end of this section. One particularly simple choice that produces good results is to set ω_{ij} equal to the inverse of the number of neighbors $\frac{1}{|N_1(\mathbf{x}_i)|}$ of vertex $\mathbf{x}_i$, for each element j of $N_1(\mathbf{x}_i)$ [27]. If $W = (\omega_{ij})$ is the square $n \times n$ matrix of weights, with the convention that the weight

ω_{ij} is equal to 0 if vertex $\mathbf{x}_j$ is not a neighbor of vertex $\mathbf{x}_i$. We also assume that once set, the weights are kept constant during the iterative smoothing process [27].
If we define the matrix $K = I - W$, with I the identity matrix, the Laplace operator applied to a graph signal $\mathbf{x}$ can be written in matrix form as follows:

$$\Delta\mathbf{x} = -K\mathbf{x}.$$

For a first-order neighborhood structure, and for the choice of weight described above, the matrix K has real eigenvalues $0 \leq k_1 \leq \ldots \leq k_n \leq 2$ with corresponding linearly independent real unit length right eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$.[9] In matrix form

$$KE = E\,diag(\mathbf{k}),$$

with $E = (\mathbf{e}_1, \ldots, \mathbf{e}_n)$, $\mathbf{k} = (k_1, \ldots, k_n)^T$, and $diag(\mathbf{k})$ the diagonal matrix with k_i in its i-th diagonal position. Seen as discrete surface signals, these eigenvectors should be considered as the *natural vibration modes* of the surface, and the corresponding eigenvalues as the associated *natural frequencies.*

Since $\mathbf{e}_1, \ldots, \mathbf{e}_n$ form a basis of n-dimensional space, every signal $\mathbf{x}$ can be written in a unique way as a linear combination

$$\mathbf{x} = \sum_{j=1}^{n} \hat{\mathbf{x}}_j \mathbf{e}_j = E\,\hat{\mathbf{x}}.$$

The vector of coefficients $\hat{\mathbf{x}}$ is the Discrete Fourier Transform of $\mathbf{x}$, and E is the Fourier Matrix.

Fourier descriptors have been widely used then in the computer vision literature as multi-resolution shape descriptors for object recognition [26]. The method of Fourier descriptors does not produce shrinkage, but nevertheless, it does have two significant problems. The first problem is that it does not extend to surfaces of arbitrary topological type [26]. Second, in general, the matrix K is large, and although sparse, it is almost impossible to reliably compute its eigenvalues and eigenvectors. Even using the Fast Fourier Transform algorithm, the number of arithmetic operations is of the order of $n\,log(n)$, where n is the number of vertices. This makes it impractical to smooth vertex positions of large meshes with the Fourier descriptors method [28].

3.2.2 The Laplacian Smoothing Method

Perhaps the most popular linear technique of geometric smoothing parametrized curves is the so-called *Laplacian Smoothing Method* or *Gaussian Smoothing Method.* In the continuous case, Laplacian smoothing is performed by convolving the vector function that

---

[9] The proof can be found in the Appendix of [27].

parameterizes the curve with a Gaussian kernel. Laplacian smoothing is defined on polyhedral surfaces of arbitrary topology. When Laplacian smoothing is applied to a noisy 3D polygonal mesh without constraints, noise is removed, but significant shape distortion may be introduced. The main problem is that Laplacian smoothing produces shrinkage, because in the limit, all the vertices of the mesh converge to their barycenter [28].

Except for the zero frequency, all the frequencies are attenuated. To prevent shrinkage, the smoothing algorithm must produce a low-pass filter effect [26].

The set of displacements $\Delta\mathbf{x}_i$ produced by the Laplacian smoothing step that moves each vertex to the barycenter of its neighbors can be described as the result of applying the Laplace operator to the vertices of the mesh. The Laplace operator is defined on a signal $\mathbf{x}$ by weighted averages over the neighborhoods

$$\Delta\mathbf{x}_i = \sum_{\mathbf{x}_j \in N_1(\mathbf{x}_i)} \omega_{ij}(\mathbf{x}_j - \mathbf{x}_i). \tag{16}$$

For each signal $\mathbf{x}_i$ the weights ω_{ij} are still normalized so that

$$\sum_{\mathbf{x}_j \in N_1(\mathbf{x}_i)} \omega_{ij} = 1, \tag{17}$$

but otherwise they can be chosen in many different ways taking into consideration the neighborhood structures. Some particular choices are mentioned below. Once the Laplace operators for all vertices are computed, the new positions $\mathbf{x}'_i$, are obtained by adding to each vertex current position $\mathbf{x}_i$ its corresponding *displacement vector* $\Delta\mathbf{x}_i$

$$\mathbf{x}'_i = \mathbf{x}_i + \lambda\Delta\mathbf{x}_i \tag{18}$$

where $0 < \lambda < 1$ is called the *scale factor*, which can be a common value for all the vertices or be vertex dependent [26]. The scaling factor is used to control the speed of the diffusion process [28]. For $\lambda < 0$ and $\lambda \geq 1$ the algorithm enhances high frequencies instead of attenuating them [27]. One step of the Laplacian smoothing algorithm can be described in matrix form as follows:

$$\mathbf{x}' = (\mathbf{I} - \lambda K)\mathbf{x} = f(K)\mathbf{x},$$

where $f(K)$ is a matrix obtained by evaluating the univariante polynomial $f(k)$ in the matrix K. The function of one variable $f(k)$ is the *transfer function* of the filter. To produce significant smoothing this process is iterated n times. In the case of Laplacian smoothing, where the transfer function is $f(k) = (1 - \lambda k)^n$, with $0 < \lambda < 1$, we see that for every $k \in (0, 2]$, we have $(1 - \lambda k)^n \to 0$ when $n \to \infty$ because $|1 - \lambda k| < 1$. This means that all the frequency components, other than the zero frequency component (the barycenter of all the vertices), are attenuated for large n. On the other hand, the neighborhood normalization constraint of equation (17) implies that the matrix K always has 0 as its first eigenvalue with associated eigenvector $(1, \ldots, 1)^T$, and the zero frequency component is preserved without changes because $f(0) = 1$ independently of the values of λ and n. In conclusion Laplacian smoothing filters out too many frequencies [28].

The Laplacian smoothing method has a number of advantages with respect to the existing method of Fourier descriptors. The first advantage is that it applies to piece-wise linear surfaces of arbitrary topological type, not only those that can be parameterized by functions defined on a rectangular domain. The second advantage is that, since first-order neighbors are defined implicitly in the list of edges or faces of curve or surface, no storage is required to encode the neighborhood structures. The third advantage is that the number of operations is a linear function of the total number of vertices, edges and faces [26].

However, by iterating the smoothing process a significant shrinkage effect is also imposed, because the convolution with a Gaussian kernel is not a low-pass filter operation [26]. To define a low-pass filter, the matrix $(\mathbf{I} - \lambda K)$ must be replaced by some other function $f(K)$ of the matrix K. Taubin's non-shrinking fairing algorithm, described in the next section, is one particularly efficient choice.

3.2.3 Taubin's Smoothing Method

Taubin presented in [26] a signal processing approach to the problem of fairing piece-wise linear shapes of arbitrary dimension and topology. Without changing the connectivity of the faces the faired surface has exactly the same number of vertices and faces as the original one. The main innovation is the modification of the Laplacian smoothing method to prevent shrinkage, obtaining a simple and general method to smooth general and arbitrary polygonal curves and polyhedral surfaces that has all the good properties of earlier methods, but none of their disadvantages [26].

This method produces a low pass filter effect, where curve or surface curvature takes the place of frequency. The original non-smooth curve or surface is modeled as an underlying smooth curve or surface, plus a normal perturbation vector field. The underlying curve or surface is bounded above in curvature, and the perturbation that needs to be filtered out is regarded as zero mean high curvature noise. The two scale factors determine the pass-band and stop-band curvatures. For higher attenuation in the stop-band, the two Laplacian smoothing steps must be repeated alternating the two scale factors λ and μ. The amount of attenuation is then determined by the number n of iterations.

The smoothing algorithm consists of two consecutive Laplacian smoothing steps. After a first Laplacian smoothing step with a positive scale factor λ is applied to all the vertices of the shape, a second Laplacian smoothing step is applied to all the vertices, but with a negative scale factor μ, greater in magnitude than the first scale factor $(0 < \lambda < -\mu)$. To produce a significant smoothing effect, these two steps must be repeated, alternating the

positive and negative scale factors, a number of times.
If the process is iterated n times, the output can still be expressed as $\mathbf{x}^n = f(K)^n\mathbf{x}$. It is well known that for any of these functions, the matrix $f(K)$ has the eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ of the matrix K, and eigenvalues the result $f(k_1), \ldots, f(k_n)$ of evaluating the function on the eigenvalues of K (see 3.2.1). Taubin based his approach on defining a suitable generalization of frequency to the case of arbitrary connectivity meshes. Using a discrete approximation to the Laplacian, its eigenvectors become the "frequencies" of a given mesh. Repeated application of the resulting linear operator to the mesh was then employed to tailor the frequency content of a given mesh [5]. Since for any polynomial transfer function

$$\mathbf{x}' = f(K)\mathbf{x} = \sum_{j=1}^{n} f(k_j)\hat{\mathbf{x}}_j\mathbf{e}_j,$$

because $K\mathbf{e}_j = k_j\mathbf{e}_j$, to define a low-pass filter a polynomial such that $f(k_i) \approx 1$ for low frequencies, and $f(k_i) \approx 0$ for high frequencies in the region of interest $k \in [0, 2]$ has to be found [27]. The so-called *Taubin smoothing algorithm* or *$\lambda|\mu$-algorithm* uses

$$f(k) = (1 - \lambda k)(1 - \mu k),$$

where $0 < \lambda$, and μ is a new negative scale factor such that $\mu < -\lambda$. After performing the Gaussian smoothing step of equation (18) with positive scale factor λ for all the vertices - the shrinking step - we then perform another similar step can be created

$$\mathbf{x}_i' = \mathbf{x}_i + \mu\,\Delta\mathbf{x}_i$$

for all the vertices, but with negative scale factor μ instead of λ - the un-shrinking step [27]. Since $f(0) = 1$ and $\mu + \lambda < 0$, there is a positive value of k, the *pass-band frequency* k_{PB}, such that $f(k_{PB}) = 1$. The value of k_{PB} is

$$k_{PB} = \frac{1}{\lambda} + \frac{1}{\mu}. \tag{19}$$

The graph of the transfer function $f(k)^n$ displays a typical *low-pass filter* shape in the region of interest $k \in [0, 2]$. The *pass-band region* extends from $k = 0$ to $k = k_{PB}$, where $f(k)^n \approx 1$. As k increases from $k = k_{PB}$ to $k = 2$, the transfer function decreases to zero. The faster the transfer function decreases in this region, the better. The rate of decrease is controlled by the number of iterations n.

Once k_{PB} has been chosen, we have to define λ and n (μ comes out of equation (19) afterwards). Of course we want to minimize n, the number of iterations. To do so, λ must be chosen as large as possible, while keeping $|f(k)| < 1$ for $k_{PB} < k \leq 2$ (if $|f(k)| \geq 1$ in $[k_{PB}, 2]$, the filter will enhance high frequencies instead of attenuating them). For $k_{PB} < 1$ the choice of λ so that $f(1) = -f(2)$ ensures a stable and fast filter.

In Taubin's case the problem of surface fairing is reduced to sparse matrix multiplication instead a linear time complexity operation [27]. The method is efficient both in terms of computational complexity, and in terms of storage requirements. The complexity is linear in the number of edges or faces of the shape, and the storage required is a linear function of the vertices.

Automatic Anti-Shrinking Fairing

Pure diffusion will, by nature, induce shrinkage. This is inconvenient as this shrinking may be significant for aggressive smoothing. Taubin proposed to use a linear combination of the Laplacian transfer function $f(k)$ to amplify low frequencies in order to balance the natural shrinking. Unfortunately, the linear combination depends heavily on the mesh in practice, and this requires fine tuning to ensure both stable and non-shrinking results.
Desbrun et al. apply in [5] a simple scale on the vertices to achieve exact volume preservation. By multiplying all the vertex positions by $\beta = (\frac{V_0}{V_n})^{\frac{1}{3}}$, the volume is guaranteed to go back to its original value. As this is a simple scaling, it is harmless in terms of frequencies. To put it differently, this scaling amplifies all the frequencies in the same way to change the volume back. The overall complexity for volume preservation is then linear [5].

3.2.4 Constraints

In [27] Taubin extended the analysis, and modified the algorithm accordingly, to accommodate different types of constraints. The ability to impose constraints to the smoothing process, such as specifying the positions of some vertices, or normal vectors, specifying ridge curves, or the behavior of the smoothing process along the boundaries of the mesh, is needed in the context of free-form interactive shape design. Taubin [27] shows that by modifying the neighborhood structure certain kinds of constraints can be imposed without any modification of the algorithm, while other constraints require minor modifications and the solution of small linear systems.

Interpolatory Constraints

A simple way to introduce interpolatory constraints in Taubin's fairing algorithm is to use non-symmetric neighborhood structures. If no other vertex is a neighbor of a certain vertex $\mathbf{v}_i$, i.e. the neighborhood of $\mathbf{v}_i$ is empty, then the value $\mathbf{x}_i$ of any discrete surface signal $\mathbf{x}$ does not change during the fairing process, because the discrete Laplacian $\Delta\mathbf{x}_i$ is equal to zero by definition of empty sum. Other vertices are allowed to have $\mathbf{v}_1$ as a neighbor, though.

Hierarchical Interpolation

This is another application of non-symmetric neighborhoods. We start by assigning a numeric label l_i to each vertex of the surface. The vertex $\mathbf{v}_j$ will be defined as a neighbor of

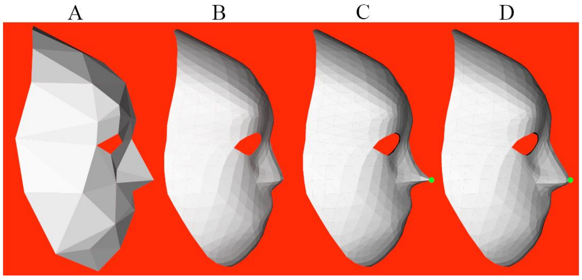


Figure 3.2: Example of surfaces designed using smoothing steps with one interpolartory constraint. (A): Skeleton. (B): Surface (A) after Taubin's smoothing without constraints. (C): Same as (B) but with non-smooth interpolatory constraint. (D) Same as (B) but with smooth interpolatory constraint.

vertex $\mathbf{v}_i$, if $\mathbf{v}_i$ and $\mathbf{v}_j$ share an edge (or face) and if $l_i \leq l_j$. Note that if $\mathbf{v}_j$ is a neighbor of $\mathbf{v}_i$ and $l_i < l_j$, then $\mathbf{v}_i$ is not a neighbor of $\mathbf{v}_j$. The symmetry applies only to vertices with the same label. For example, if we assign a label to all the boundary vertices of a surface with boundary greater than we assign a label to all the internal vertices, then the boundary is faired as a curve, independently of the interior vertices, but the interior vertices follow the boundary vertices. If we also assign a label to a closed curve composed of internal edges of the surface equal to the label of the boundary vertices, then the resulting surface will be smooth along, and on both sides of the curve, but not necessarily across the curve. If we also assign a label greater than the label of boundary and internal vertices to some isolated points along the curves, then those vertices will in fact not move, because they will have empty neighborhoods.

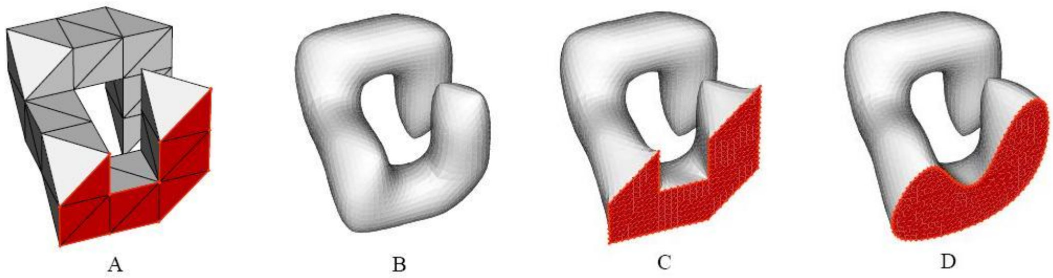


Figure 3.3: Example for different choices of boundary constraints. (A): Skeleton with marked vertices. (B): Surface (A) after three levels of Taubin's smoothing without constraints. (C): Same as (B) but with empty neighborhoods of marked vertices. (D): Same as (B) but with hierarchical neighborhoods, where marked vertices have label 1 and unmarked vertices have label 0.

Tangent Plane Constraints After section 2 it follows that imposing normal constraints at $\mathbf{v}_i$ is achieved by imposing linear constraints on $\Delta\mathbf{v}_i$. If $\mathbf{n}_i$ is the desired normal direction at vertex $\mathbf{v}_i$ after the smoothing process, and $\mathbf{s}_i$ and $\mathbf{t}_i$ are two linearly independent vectors tangent to $\mathbf{n}_i$, the surface after n iterations of the smoothing algorithm will satisfy the normal desired constraint at the vertex $\mathbf{v}_i$ in the following two linear constraints

$$\mathbf{s}_i^T \Delta\mathbf{v}_i^n = \mathbf{t}_i^T \Delta\mathbf{v}_i^n = 0$$

are satisfied. This leads us to the problem of smoothing with general linear constraints.

3.2.5 Different Ways of choosing Weights

The weights of the discrete Laplacian in equation (16) can be chosen in many different ways taking into consideration the neighborhood structures. The three weighting schemes described in this section can be applied to both Laplacian smoothing and Taubin smoothing, but Fujiwara weights and Desbrun weights must be recomputed after each iteration, or after a small number of iterations. This makes the whole smoothing process a nonlinear operation, and computationally more expensive.

By first choosing an edge cost $c_{ij} = c_{ji} \geq 0$ for each graph edge, and then setting $\omega_{ij} = \frac{c_{ij}}{c_i}$, where c_i is the average cost of edges incident to $\mathbf{x}_i$:

$$c_i = \sum_{j \in N_1(\mathbf{x}_i)} c_{ij} > 0.$$

If all the edges have unit cost $c_{ij} = 1$, then for each neighbor $\mathbf{x}_j$ of $\mathbf{x}_i$, the weight ω_{ij} is equal to the inverse of the number of neighbors $\frac{1}{|N_1(\mathbf{x}_i)|}$ of $\mathbf{x}_i$. This choice of weights - called *equal weights* - is independent of the vertex positions, or geometry, of the mesh, and only function of the connectivity of the mesh [28]. Very satisfactory results are obtained on meshes which display very small variation in edge length and face angles across the whole mesh. When these assumptions are not made, local distortions are introduced. The edge weights can be used to compensate for the irregularities of the tesselation, and produce results which are functions of the local geometry of the signal, rather than the local parameterization.

A more general way of choosing weights for a surface with a first-order neighborhood structure, is using a positive function $\phi(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_j, \mathbf{x}_i)$ defined on the edges of the surface

$$\omega_{ij} = \frac{\phi(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{h \in N_1(\mathbf{x}_i)} \phi(\mathbf{x}_i, \mathbf{x}_h)}.$$

For example, the function can be the surface area of the two faces that share the edge or some power of the length of the edge $\phi(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^\alpha$. When using a power of the length of the edges as weighting function, the exponent $\alpha = -1$ produces good results [27]. The so-called *Fujiwara weights* makes the Laplace operator independent of the edge lengths, and only dependent on the directions of the vectors pointing to the neighboring vertices. This weighting scheme does not solve the problems arising from unequal face angles.

Desbrun weights compensate not only for unequal edge lengths, but also for unequal face angles. Laplacian smoothing with equal edge costs tends to equalize the lengths of the edges, and so, tends to make the triangular faces equilateral. The vertex displacements produced by the Laplace operator can be decomposed into a normal and a tangential component. Based on a better approximation to the curvature normal, Desbrun proposes in [5] the following choice of edge costs

$$c_{ij} = cot\, \alpha_{ij} + cot\, \beta_{ij},$$

where α_{ij} and β_{ij} are the two angles opposite to the edge $e = (i, j)$ in the two triangles having e in common. This choice of weights produces no tangential drift when all the faces incident to the vertex are coplanar [28].

3.3 Spring Model

In the spring model, a linear spring whose length approximately represents a curvature radius, is attached along the normal line of each polygon node. Energy is assigned to the difference of the lengths, that is, the difference in curvature radius, of neighboring springs. In the paper [33], Yamada et al. presented an iteration-based algorithm for generating fair polygonal curves and surfaces that is based on a new discrete spring model. The algorithm moves the nodes to suitable positions while minimizing the curvature variation by an iterative approach under the given constraints. To update the node positions, two types of spring forces are applied to each node. First, a force acting in the normal direction, to optimize the curvature variation and second, a force in the direction perpendicular to the normal, to optimize the node distribution. It accepts various constraints, such as positional, normal and least-square constraints, so that useful surface models can be generated. The polygonal surfaces are not limited to triangular meshes; they also include quadrilateral meshes. Theoretically, n-sided faces such as pentagons or hexagons may also be included in the meshes.

Consider a planar curve and its normal lines at two neighboring sampling points $\mathbf{p}_i$ and $\mathbf{p}_j$. For a curvature-continuous curve, if $\mathbf{p}_i$ approaches $\mathbf{p}_j$, the intersection point $\mathbf{H}$ of the normal lines converges to a center of curvature at $\mathbf{p}_j$. Therefore, the idea is to attach a linear spring to each normal line of a node consisting in a polygonal curve or surface.

The first step of the algorithm is to calculate a unit normal vector $\mathbf{n}_i$ for each node $\mathbf{p}_i$. In the case of a polygonal surface the spring model algorithm calculates pseudo-normal vectors, because the polygonal surface is a discrete model and therefore, the unit normal vector must be calculated only approximately. In the implementation, the unit normal $\mathbf{n}_i$ is calculated by averaging the normals of polygonal faces adjoining the node. In the early phase of iterations, the normal vector is unreliable; however, as the iterations proceed, the normal vector converges to the reliable normal of the fair surface.

The linear spring works to keep equal the spring lengths $|\mathbf{p}_i - \mathbf{H}|$ and $|\mathbf{p}_j - \mathbf{H}|$ of a V-shape formed by $\mathbf{p}_i$, $\mathbf{H}$ and $\mathbf{p}_j$. The spring length approximately represents the curvature radius; therefore, keeping the spring length equal is equivalent to minimizing variation in the curvature radius. Suppose node $\mathbf{p}_j$ is fixed by a constraint. If $|\mathbf{p}_i - \mathbf{H}|$ is smaller than $|\mathbf{p}_j - \mathbf{H}|$, node $\mathbf{p}_i$ moves to a new position along the normal $\mathbf{n}_i$ in the direction that enlarges $|\mathbf{p}_i - \mathbf{H}|$ to the size of $|\mathbf{p}_j - \mathbf{H}|$. If $|\mathbf{p}_i - \mathbf{H}|$ is larger, the node $\mathbf{p}_i$ moves so that $|\mathbf{p}_i - \mathbf{H}|$ is shortened to the size of $|\mathbf{p}_j - \mathbf{H}|$. In a stable configuration, $\mathbf{p}_i$ and $\mathbf{p}_j$ are considered to be on a circular arc whose center is at $\mathbf{H}$ and whose curvature radius is $|\mathbf{p}_j - \mathbf{H}|$.

On the supposition that the vertex $\mathbf{p}_j$ is fixed, a *displacement* $\mathbf{dp}_{ij}$ of a node $\mathbf{p}_i$ from a current position to a new position is formally defined by the force of the spring model. Let $\mathbf{p}_i$ and $\mathbf{p}_j$ be two nodes, and let $\mathbf{n}_i$ and $\mathbf{n}_j$ be unit normal vectors associated with the nodes, respectively. We assume the inner product $\langle \mathbf{n}_i, \mathbf{n}_j \rangle$ to be positive[10]. An intersection of the two normal lines is denoted by $\mathbf{H}$. Let t_i and t_j be real values satisfying

$$\mathbf{p}_i - \mathbf{H} = t_i \mathbf{n}_i, \tag{20}$$

$$\mathbf{p}_j - \mathbf{H} = t_j \mathbf{n}_j. \tag{21}$$

$|t_i|$ and $|t_j|$ correspond to the distances $|\mathbf{p}_i - \mathbf{H}|$ and $|\mathbf{p}_j - \mathbf{H}|$, respectively, because $\mathbf{n}_i$ and $\mathbf{n}_j$ are unit vectors. We define the displacement $\mathbf{dp}_{ij}$ of node $\mathbf{p}_i$ by our spring model as

$$\mathbf{dp}_{ij} = (t_j - t_i)\mathbf{n}_i. \tag{22}$$

By solving some elemetary equations, we obtain

$$t_j - t_i = \frac{\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n}_i + \mathbf{n}_j \rangle}{1 + \langle \mathbf{n}_i, \mathbf{n}_j \rangle}.$$

Therefore, equation (22) is written as

$$\mathbf{dp}_{ij} = \frac{\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n}_i + \mathbf{n}_j \rangle}{1 + \langle \mathbf{n}_i, \mathbf{n}_j \rangle} \mathbf{n}_i. \tag{23}$$

The denominator always has a non-zero value, because the inner product is assumed to be positive.

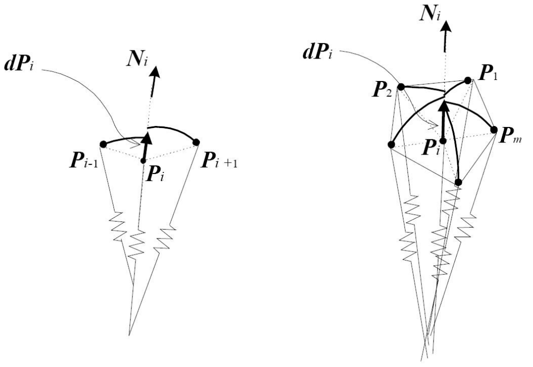
One may consider that a numerical error occurs when two normal lines are parallel because of the absence of $\mathbf{H}$. However, equation (23) does not use $\mathbf{H}$ directly; therefore, the spring

---

[10] Therefore, if two normal vectors with their negative inner product are given, let either $\mathbf{n}_i$ or $\mathbf{n}_j$ be the direction-reversed vector.

model is stable even in the case of parallel normal lines. In this case, equation (23) is deduced to the following equation:

$$\mathbf{dp}_{ij} = \langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n} \rangle \, \mathbf{n}, \quad (\mathbf{n} = \mathbf{n}_i = \mathbf{n}_j).$$



(A) (B)

Figure 3.4: Displacement of a node, if the normal lines have no intersection point.

In the case of a planar curve, normal lines along $\mathbf{n}_i$ and $\mathbf{n}_j$ always have an intersection. However, if we consider a non-planar curve or a surface, normal lines do not always intersect at a point. Therefore, we cannot use equation (20) as it is for non-planar cases. Instead of an intersection point $\mathbf{H}$ in equation (20), we use $\mathbf{H}_i$ and $\mathbf{H}_j$, which are the feet of the shortest line segment connecting two normal lines. Then, for non-planar cases, we modify equation (20) to obtain the following equations

$$\mathbf{p}_i - \mathbf{H}_i = t_i \mathbf{n}_i,$$
$$\mathbf{p}_j - \mathbf{H}_j = t_j \mathbf{n}_j.$$

An equation for $(t_j - t_i)$ can be calculated analogously to the planar case. In fact, the equation for $(t_j - t_i)$ in equation (22) for the non-planar case is the same as for the planar case.

After calculating the displacement $\mathbf{dp}_{ij}$ for each node $\mathbf{p}_i$ caused by the force exerted by the spring model, $\mathbf{p}_i$ moves to a new position along the normal $\mathbf{n}_i$ by the weighted average $\mathbf{dp}_i$ of the displacements $\mathbf{dp}_{ij}$ as follows

$$\mathbf{dp}_i = \frac{\sum_{j=1}^{m} \omega_j \mathbf{dp}_{ij}}{\sum_{j=1}^{m} \omega_j}, \tag{24}$$

where $\omega_j, j = 1, \ldots, m$ are weights for the averaging. In the paper [33] Yamada et al. determined the weight ω_j by inverse of the length of the edge connecting $\mathbf{p}_i$ and $\mathbf{p}_j$

$$\omega_j = \frac{1}{\|\mathbf{p}_i - \mathbf{p}_j\|}, j = 1, \ldots, m.$$

It is important to maintain the regular node distribution during the iteration process, because uneven distribution of nodes results in incorrect estimation of curvature. To obtain the regular node distribution, Yamada et al. use a variation of a Laplacian smoothing operator. However, applying the regular Laplace operator offsets the displacement $\mathbf{dp}_i$ in Equation (24). Therefore, their idea is to use only a component $\mathbf{dp}_{u,i}$, that is perpendicular to the normal $\mathbf{n}_i$, of the displacement of the Laplace operator. Using only this component creates two displacements $\mathbf{dp}_i$ and $\mathbf{dp}_{u,i}$ perpendicular to each other. Therefore, the two displacements do not offset each other. The displacement $\mathbf{dp}_{u,i}$ is written as follows:

$$\begin{aligned}
\mathbf{dp}_{u,i} &= \mathbf{dp}_{u_0,i} - \mathbf{dp}_{u_1,i}, \\
\mathbf{dp}_{u_0,i} &= \frac{1}{m}\sum_{j=1}^{m} \mathbf{p}_j - \mathbf{p}_i, \\
\mathbf{dp}_{u_1,i} &= \langle \mathbf{dp}_{u_0,i}, \mathbf{n}_i \rangle \mathbf{n}_i.
\end{aligned}$$

Basically, the algorithm for curve fairing is the same as the one for surface fairing. The case of a planar curve does not involve any extension of the surface case; however, in the case of a non-planar curve, the calculation of the pseudo-normal $\mathbf{n}_i$ is more difficult than in the surface case. From a sequence of nodes $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, and $\mathbf{p}_{i+1}$, the unit tangent $\mathbf{t}_i$ and the unit binormal $\mathbf{b}_i$ is calculated as follows:

$$\begin{aligned}
\mathbf{t}_i &= \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{\|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|}, \\
\mathbf{b}_i &= \frac{(\mathbf{p}_i - \mathbf{p}_{i-1}) \times (\mathbf{p}_{i+1} - \mathbf{p}_i)}{\|(\mathbf{p}_i - \mathbf{p}_{i-1}) \times (\mathbf{p}_{i+1} - \mathbf{p}_i)\|},
\end{aligned}$$

where × denotes the outer product. As the outer product of $\mathbf{b}_i$ and $\mathbf{t}_i$, we calculate the unit principal normal $\mathbf{n}_i$ as follows:

$$\mathbf{n}_i = \frac{\mathbf{b}_i \times \mathbf{t}_i}{\|\mathbf{b}_i \times \mathbf{t}_i\|}.$$

If $\mathbf{p}_{i-1}, \mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are collinear, $\mathbf{b}_i$ and $\mathbf{n}_i$ are zero vectors; then the displacement $\mathbf{dp}_i$ is a zero vector according to equation (23). The best way to obtain a fair curve is to apply spring forces in the directions of both $\mathbf{b}_i$ and $\mathbf{n}_i$; however, in practice, applying a force only in the direction of $\mathbf{n}_i$ gives a fair curve, even if the problem is a non-planar case.

Constraints

Constraints are considered to be external forces for controlling the shape of a surface. Various kinds of constraints can be considered, depending on the requirements of applications.

A positional constraint fixes a node to a certain position during the iterations. Therefore, the calculation of the displacement can be skipped. A normal constraint fixes a normal of

a node to a certain direction. For the normal fixed node, the pseudo-normal calculation is skipped and the given normal is assigned. In the approach of the paper [33], positional constraints must be given at least to end nodes for the case of an open polygonal curve and to boundary nodes for the case of an open polygonal surface. If it is necessary to modify the shape of the boundary curves of a surface, start from the modeling of boundary curves and go on to the modeling of the surface bounded by the boundary curves.
Another constraint, which is mentioned in [33] is an indirect constraint, which is not directly connected to certain nodes. One major constraint is scattered points. During the iterations, connections between the constraints and the nodes are updated dynamically. This is an important application in Computer Aided Design and Computer Graphics for generating a smooth surface fitted to scattered points in the least-square sense.

Termination Condition

The positions and normals of nodes are updated in each iteration and the iterations are continued until the termination condition is satisfied. In one iteration the updated latest positions are always used to calculate the positions of other nodes. To determine when the iteration terminates, the maximum among the norms of all node displacements is compared with a given threshold ϵ. If the maximum norm is less than the threshold ϵ, the iterations are terminated. The size of the displacement depends on the resolution of the polygonal surface; therefore the displacement should be normalized by the sizes of polygonal faces. Yamada et al. normalize the displacement of each node by the average length of its neighboring edges. The maximum norm of the normalized displacements is then compared with the threshold ϵ.

According to Yamada et al., the algorithm is decidedly robust and stable; however, the convergence of the algorithm needs to be proved in future work.

3.4 Algorithm of Schneider-Kobbelt

Schneider and Kobbelt presented a new algorithm to create fair discrete surfaces satisfying G^1 boundary conditions. As already mentioned in chapter 3.1.2 and 3.1.3 the energy minimization problem can be transformed into the problem of solving a differential equation with constraints. Instead of using variational calculus, the partial differential equation approach can also be seen as a reasonable approach to the fairing problem in its own right, which is especially important for fairing based on geometric invariants [20]. This means instead of searching for intrinsic energy functionals that lead to handy partial differential equations, it seems promising to search directly for simple intrinsic partial differential equations producing fair solutions. This idea leads to the question which partial differential equation based on geometric invariants seems suited for the creation of fair surfaces satisfying G^1 boundary constraints.

The algorithm presented in [20] from Schneider and Kobbelt is based on solving a non-linear fourth-order partial differential equation

$$\Delta_B H = 0, \tag{25}$$

that only depends on intrinsic surface properties, i.e. properties that depend on the geometry alone, instead of being derived from a particular surface parameterization. Here Δ_B is the Laplace-Beltrami operator and H the mean curvature. Equation (25) can be interpreted as a surface analogon to the planar equation $\kappa'' = 0$, where the derivative of the curvature κ is with respect to arc length [21]. To simplify the computation the fourth-order partial differential equation is factorized into a set of two nested second-order problems thus avoiding the estimation of higher order derivatives. Because of the mean value property of the Laplacian, it is guaranteed that the extremal mean curvature values of a solution of (25) will be reached at the border and that there are no local extrema in the interior.

In [20] solutions of (25) were approximated by meshes in the special case where the meshes have subdivision connectivity and the boundary vertices could be regularly sampled on a smooth curve. One year later Schneider and Kobbelt presented in [21] an algorithm for smoothing arbitrary triangle meshes that does not have such limitations. There are also no restrictions concerning the mesh structure and the boundary vertices and we are free to choose an inner fairness criteria. Nevertheless, the resulting construction algorithm is fast and can be implemented compactly. Instead of trying to simulate the continuous case using local quadratic approximations, we use the discrete data of our minimization process directly [21].

The algorithm completely seperates outer and inner fairness by discretizing an intrinsic partial differential equation. The discretization relies on the fact that there is a tight connection between the Laplace-Beltrami operator Δ_B and the mean curvature normal of a surface. We can discretize the equation $\Delta_B H = 0$ at a vertex $\mathbf{q}_i$ as

$$\sum_{\mathbf{q}_j \in N(\mathbf{q}_i)} (cot\,\alpha_j + cot\,\beta_j)(H(\mathbf{q}_i) - H(\mathbf{q}_j)) = 0,$$

where α_j and β_j are the angles of the corners, which are on the opposite of the edge defined by $\mathbf{q}_i$ and $\mathbf{q}_j$. If this equation is satisfied at all inner vertices $\mathbf{q}_i$ and if we further know all mean curvature values for the boundary vertices, this leads us to a sparse linear system in the unknown $H(\mathbf{q}_i)$ for inner vertices

$$S_{ii} = \sum_{\mathbf{q}_j \in N(\mathbf{q}_i)} (cot\,\alpha_j + cot\,\beta_j), \tag{26}$$

$$S_{ij} = \begin{cases} -(cot\,\alpha_j + cot\,\beta_j) & : \mathbf{q}_j \in N(\mathbf{q}_i) \cap V_I(M) \\ 0 & : otherwise, \end{cases} \tag{27}$$

where $V_I(M)$ defines the set of all vertices in the interior of the mesh M. The matrix S is symmetric and - as long as no triangle areas of the mesh vanish - positive definite. An elegant proof of the mathematical structure of this matrix can be found in the paper [18] by Pinkall and Polthier.

Furthermore, a discretization of the mean curvature $H(\mathbf{q}_i)$ at a vertex $\mathbf{q}_i$ is presented, which depends on the vertices in a local neighborhood. There are various techniques to discretize surface curvatures, but to be applicable to the construction algorithm of Schneider Kobbelt, it is important that - for a given mesh connectivity - the discretization of $H(\mathbf{q}_i)$ is a continuous function of those vertices.
The curvature discretization algorithm that seems ideal for their needs was presented by Moreton and Séquin in [17]. The idea of their approach is to use the fact that the normal curvature distribution cannot be arbitrary, but is determined by Euler's theorem (see 2.1.4).

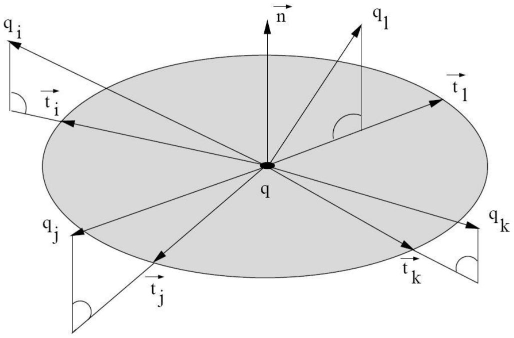


Figure 3.5: Projecting the neighborhood of $\mathbf{q}$ onto the plane defined by $\mathbf{n}$ and normalizing the results we get the normal curvature directions $\mathbf{t}_i$.

To each vertex $\mathbf{q}_j \in N(\mathbf{q})$ we can assign a unit direction vector $\mathbf{t}_j$ by projecting $\mathbf{q}_j$ into the plane defined by the normal $\mathbf{n}$ and scaling this projection to unit length (cf. Figure 3.5). For each $\mathbf{q}_j$ we can now estimate a normal curvature $\widetilde{\kappa}_j$ as the inverse of the circle radius defined by $\mathbf{q}, \mathbf{q}_j$ and $\mathbf{t}_j$

$$\widetilde{\kappa}_j = 2\frac{\langle \mathbf{q}_j - \mathbf{q}, \mathbf{n} \rangle}{\langle \mathbf{q}_j - \mathbf{q}, \mathbf{q}_j - \mathbf{q} \rangle}. \tag{28}$$

Using the Euler formula, the normal curvature κ_n for a direction $\mathbf{t}$ by the principal curvatures κ_1 and κ_2 and the principal curvature directions $\mathbf{e}_1$ and $\mathbf{e}_2$. Let t_x and t_y be the coordinates of $\mathbf{t}$ in the basis b_x, b_y and let e_x and e_y be the coordinates of $\mathbf{e}_1$, then the normal curvature can be expressed as

$$\kappa_n = \begin{pmatrix} t_x & t_y \end{pmatrix} \cdot K \cdot \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

with

$$K = \begin{pmatrix} e_x & e_y \\ -e_y & e_x \end{pmatrix} \cdot \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix} \cdot \begin{pmatrix} e_x & e_y \\ -e_y & e_x \end{pmatrix}.$$

The idea of Moreton and Séquin is to use the normal curvatures $\widetilde{\kappa}_j$ to create a linear system and find estimates for the unknown principal curvature values by determining the least square solution. Let $t_{j,x}$ and $t_{j,y}$ denote the coordinates of $\mathbf{t}_j$ and let m be the valence of $\mathbf{q}$, then we get by using Euler's theorem

$$A\mathbf{x} = \mathbf{b},$$

where

$$A = \begin{pmatrix} t_{1,x}^2 & t_{1,x}t_{1,y} & t_{1,y}^2 \\ t_{2,x}^2 & t_{2,x}t_{2,y} & t_{2,y}^2 \\ \vdots & \vdots & \vdots \\ t_{m,x}^2 & t_{m,x}t_{m,y} & t_{m,y}^2 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \widetilde{\kappa}_1 \\ \widetilde{\kappa}_2 \\ \vdots \\ \widetilde{\kappa}_m \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} e_x^2\kappa_1 + e_y^2\kappa_2 \\ 2e_xe_y(\kappa_1 - \kappa_2) \\ e_x^2\kappa_2 + e_y^2\kappa_1 \end{pmatrix}.$$

e_x and e_y are the coordinates of the principal curvature direction $\mathbf{e}_1$. Since $x_0 + x_2 = \kappa_1 + \kappa_2$ this means the mean curvature is determined by

$$H = \frac{1}{2}(x_0 + x_2).$$

The input data for the algorithm consists of vertices and unit normals that form the G^1 boundary condition and an initial mesh M_0 that interpolates the boundary vertices. The idea of the construction algorithm is to create a mesh sequence $M_k, k = 0, 1, 2, \ldots$ by iteratively updating the vertices, until the outer and inner fairness conditions are sufficiently satisfied. Instead of solving a fourth-order problem directly, we factorize it into two second-order problems which are solved sequentially. The factorization idea is inspired by the following observation: Given a fixed Laplace-Beltrami operator and fixed mean curvature values at the boundary vertices[11] of a mesh M_k, the equation

$$\Delta_B H(\mathbf{q}_i) = 0 \qquad \forall \mathbf{q}_i \in V_I(M_S),$$

where the mesh M_S is a discrete solution of equation (25), can be interpreted as a Dirichlet problem for the H_i. The unknown scalar mean curvature values at the inner vertices are determined by a nonsingular linear system with a symmetric and positive definite matrix S whose coefficients are defined in (26) and (27). Solving the resulting non-singular linear system yields scalar values $\widetilde{H}(\mathbf{q}_i)$ at all inner vertices $\mathbf{q}_i \in V_I(M^k)$, that represent a discrete harmonic function. The idea is now to use this calculated scalar values $\widetilde{H}(\mathbf{q}_i)$ to update each inner vertex $\mathbf{q}_i$ so that $H(\mathbf{q}_i^{k+1}) = \widetilde{H}_i$, which is again a second-order problem. Expressed in two formulas, this factorization of $M^k \to M^{k+1}$ becomes

$$\left.\begin{array}{lll} 1. & \Delta_B \widetilde{H}(\mathbf{q}_i) & = 0 \\ 2. & H(\mathbf{q}_i^{k+1}) & = \widetilde{H}(\mathbf{q}_i) \end{array}\right\} \forall \mathbf{q}_i^k \in V_I(M^k).$$

---

[11] The set of boundary vertices is denoted by $V_B(M)$.

In practice, it is not necessary to solve the Dirichlet problem exactly. In order to be able to separate between inner and outer fairness, we only allow the vertex to move along the surface normal vector. This means we search for a scalar value t so that

$$H(\mathbf{q}_i^{k+1}) = \widetilde{H}(\mathbf{q}_i) \text{ with } \mathbf{q}_i^{k+1} = \mathbf{q}_i^k + t\mathbf{n}. \tag{29}$$

With help of a linearization technique equation (28) turns into

$$\widetilde{\kappa}_j \approx 2\frac{\langle \mathbf{q}_j - \mathbf{q}_i, \mathbf{n}\rangle}{\langle \mathbf{q}_j - \mathbf{q}_i, \mathbf{q}_j - \mathbf{q}_i\rangle} - 2t\frac{1}{\langle \mathbf{q}_j - \mathbf{q}_i, \mathbf{q}_j - \mathbf{q}_i\rangle}.$$

Using this assumption, the discretization of $H(\mathbf{q}_i^{k+1})$ determined by equation $H = \frac{1}{2}(x_0 + x_2)$ and $\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$ becomes a linear function in t. Solving this linear equation for t, we finally update $\mathbf{q}_i^{k+1} = \mathbf{q}_i^k + t\mathbf{n}$ which approximately solves (29).

The concept behind the construction algorithm works best for $\Delta_B H = 0$, but it is not restricted to that. Minimal energy surfaces are also constructed by solving their Euler-Lagrange equation $\Delta_B H = -2H(H^2 - K)$ and surfaces satisfying $\Delta_B H = const.$ However, these inhomogen problems are more costly to solve and a stable construction algorithm is more involved.

4 Explanation of the alternative Fairing Method

In the previous section 3 well-established smoothing and fairing methods where presented, which do not yield the desired result for our purpose. The aim of this section is to generate a mesh which not only has a fair surface but also fair structure lines. This section introduces a new concept, which "fairs" the mesh in an appropriate manner. We focus on fairing the structure lines instead of the surface obtaining an aesthetically appealing and well-shaped mesh. We compare this new method to the Laplacian and the Taubin smoothing method in the following section and prove that this ansatz has several aesthetic-desirable qualities that improve the appearance of the resulting meshes.

Exemplification of the Concept

The idea of the alternative fairing ansatz is to consider structure lines and thus gain a satisfying resulting mesh. Let $\mathbf{v}_{i-1}$, $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ be three consecutive direction vectors of a structure line. A structure line is considered to be fair iff $\left\|\frac{1}{2} \cdot (\mathbf{v}_{i-1} + \mathbf{v}_{i+1}) - \mathbf{v}_i\right\|^2 \approx 0$[12] which corresponds to the energy functional (6) for direction vectors (instead of points) of chapter 3.1.1. Thus the concept is to minimize the sum of the discrete bending energy of three consecutive unit vectors of a mesh

$$\min \sum_{i=2}^{\#edges-1} \left\| \frac{1}{2} \cdot \left(\frac{\mathbf{v}_{i-1}}{||\mathbf{v}_{i-1}||} + \frac{\mathbf{v}_{i+1}}{||\mathbf{v}_{i+1}||} \right) - \frac{\mathbf{v}_i}{||\mathbf{v}_i||} \right\|^2, \tag{30}$$

where the vectors are defined as

$$\begin{aligned} \mathbf{v}_{i-1} &:= \mathbf{p}_i - \mathbf{p}_{i-1}, \\ \mathbf{v}_i &:= \mathbf{p}_{i+1} - \mathbf{p}_i, \\ \mathbf{v}_{i+1} &:= \mathbf{p}_{i+2} - \mathbf{p}_{i+1} \end{aligned}$$

of four consecutive points $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i+2}$ of a structure line. To simplify we transform the formula (30) into a formulation for each point $\mathbf{p}_i$

$$\min \sum_{i=2}^{\#vertices-2} \left\| \frac{1}{2} \cdot \left(\frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{||\mathbf{p}_i - \mathbf{p}_{i-1}||} + \frac{\mathbf{p}_{i+2} - \mathbf{p}_{i+1}}{||\mathbf{p}_{i+2} - \mathbf{p}_{i+1}||} \right) - \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{||\mathbf{p}_{i+1} - \mathbf{p}_i||} \right\|^2. \tag{31}$$

Divsion by the length of the vector makes the optimization problem nonlinear and therefore too complicated for this kind of application. A demand on the fairing concept is to sustain the form of the mesh. Therefore the distance of the faired points and the original points

---

[12] Note that for a straight line $\left\|\frac{1}{2} \cdot (\mathbf{v}_{i-1} + \mathbf{v}_{i+1}) - \mathbf{v}_i\right\|^2 = 0$.

respectively has to be as minimal as possible[13]. Because of this constraint we can estimate the distance of two faired points $\|\mathbf{p}_{i+1} - \mathbf{p}_i\|$ by the length $l_i := \left\|\mathbf{p}^0_{i+1} - \mathbf{p}^0_i\right\|$ of the distance of the original points $\mathbf{p}^0_i$ and $\mathbf{p}^0_{i+1}$. This makes the optimization linear and the formula is reformed into the following objective function

$$\min \sum_{i=2}^{\#vertices-2} \left\| -\frac{1}{2l_{i-1}}\mathbf{p}_{i-1} + \left(\frac{1}{2l_{i-1}} + \frac{1}{l_i}\right)\mathbf{p}_i - \left(\frac{1}{l_i} + \frac{1}{2l_{i+1}}\right)\mathbf{p}_{i+1} + \frac{1}{2l_{i+1}}\mathbf{p}_{i+2}\right\|^2 . \qquad (32)$$

It is well-known [8] that the vector norm $||.||_2$ on $\mathbb{R}^n$ can be written as

$$\|f_i(\mathbf{x})\|^2 = f_i(\mathbf{x})^T \cdot f_i(\mathbf{x}) = (f_i(\mathbf{x}))^2,$$

whereas in this case $f_i(\mathbf{x})$ is defined as

$$f_i(\mathbf{x}) := -\frac{1}{2l_{i-1}}\mathbf{p}_{i-1} + \left(\frac{1}{2l_{i-1}} + \frac{1}{l_i}\right)\mathbf{p}_i - \left(\frac{1}{l_i} + \frac{1}{2l_{i+1}}\right)\mathbf{p}_{i+1} + \frac{1}{2l_{i+1}}\mathbf{p}_{i+2}.$$

It is possible and desirable to transform equation (32) into matrix notation, i.e.

$$f_i(\mathbf{x}) = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i.$$

The matrix $\mathbf{A}_i \in \mathbb{R}^{3\times 3\cdot\#vertices}$ is defined as

$$\mathbf{A}_i = \left(\begin{array}{c|ccc|ccc|ccc|ccc|c} \dots & a_{i-1} & 0 & 0 & a_i & 0 & 0 & a_{i+1} & 0 & 0 & a_{i+2} & 0 & 0 & \dots \\ \dots & 0 & a_{i-1} & 0 & 0 & a_i & 0 & 0 & a_{i+1} & 0 & 0 & a_{i+2} & 0 & \dots \\ & 0 & 0 & a_{i-1} & 0 & 0 & a_i & 0 & 0 & a_{i+1} & 0 & 0 & a_{i+2} & \dots \end{array}\right),$$

where

$$\begin{aligned} a_{i-1} &:= \begin{cases} -\frac{1}{2l_{i-1}} & : \ \mathbf{p}_{i-1} \text{ should be smoothed} \\ 0 & : \ \mathbf{p}_{i-1} \text{ is fixed} \end{cases} \\ a_i &:= \begin{cases} \frac{1}{2l_{i-1}} + \frac{1}{l_i} & : \ \mathbf{p}_i \text{ should be smoothed} \\ 0 & : \ \mathbf{p}_i \text{ is fixed} \end{cases} \\ a_{i+1} &:= \begin{cases} -\frac{1}{l_i} - \frac{1}{2l_{i+1}} & : \ \mathbf{p}_{i+1} \text{ should be smoothed} \\ 0 & : \ \mathbf{p}_{i+1} \text{ is fixed} \end{cases} \\ a_{i+2} &:= \begin{cases} \frac{1}{2l_{i+1}} & : \ \mathbf{p}_{i+2} \text{ should be smoothed} \\ 0 & : \ \mathbf{p}_{i+2} \text{ is fixed.} \end{cases} \end{aligned}$$

Each 3×3-block in the matrix $\mathbf{A}_i$ with the diagonal entries a_{i_j} range from column $3(i_j-1)+1$ to column $3i_j$. In this case the four consecutive points of the structure line are four points with consecutive indices. Typically two consecutive points $\mathbf{p}_{i_1}$ and $\mathbf{p}_{i_2}$ of a structure line do not have consecutive indices (i.e. $i_1 + 1 \neq i_2$). Therefore the 3×3-block related to vertex $\mathbf{p}_{i_1}$ in matrix $\mathbf{A}_i$ with the diagonal entries a_{i_1} range from column $3(i_1 - 1) + 1$ to

---

[13] We will dwell on this constraint later.

column $3i_1$ and the 3×3-block related to vertex $\mathbf{p}_{i_2}$ with the diagonal entries a_{i_2} range from column $3(i_2 - 1) + 1$ to column $3i_2$.

The vector $\mathbf{x} \in \mathbb{R}^{3 \cdot \#vertices \times 1}$ is defined as

$$\mathbf{x} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_{\#vertices} \\ y_{\#vertices} \\ z_{\#vertices} \end{pmatrix}, \text{ whereas } \mathbf{p}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}.$$

The vector $\mathbf{b}_i \in \mathbb{R}^{3 \times 1}$ is defined as

$$\mathbf{b}_i = -\frac{1}{2l_{i-1}}\mathbf{p}'_{i-1} + \left(\frac{1}{2l_{i-1}} + \frac{1}{l_i}\right)\mathbf{p}'_i - \left(\frac{1}{2l_{i+1}} + \frac{1}{l_i}\right)\mathbf{p}'_{i+1} + \frac{1}{2l_{i+1}}\mathbf{p}'_{i+2},$$

where

$$\mathbf{p}'_i := \begin{cases} \mathbf{0} & : \quad \mathbf{p}_i \text{ should be smoothed} \\ \mathbf{p}_i & : \quad \mathbf{p}_i \text{ is fixed.} \end{cases}$$

To sum up, equation (31) can be written as

$$\begin{aligned} & \min \sum_{i=2}^{\#vertices-2} \|\mathbf{A}_i\mathbf{x} + \mathbf{b}_i\|^2 \\ = & \min \sum_{i=2}^{\#vertices-2} (\mathbf{A}_i\mathbf{x} + \mathbf{b}_i)^T \cdot (\mathbf{A}_i\mathbf{x} + \mathbf{b}_i)^T \\ = & \min \ \mathbf{x}^T \underbrace{\sum_{i=2}^{\#vertices-2} \mathbf{A}_i^T\mathbf{A}_i}_{=:\mathbf{D}} \mathbf{x} + 2 \underbrace{\sum_{i=2}^{\#vertices-2} \mathbf{b}_i^T\mathbf{A}_i}_{\mathbf{c}^T} \mathbf{x} + \sum_{i=2}^{\#vertices-2} \mathbf{b}_i^T\mathbf{b}_i. \end{aligned}$$

The matrices $\mathbf{A}_1$, $\mathbf{A}_{\#vertices-1}$ and $\mathbf{A}_{\#vertices}$ are defined as zero matrices. To solve the minimization problem which was explained above a fairing method without any constraints would be used. As already mentioned we include the sustainment of the shape of the original mesh to avoid any shape deformation of the faired mesh.

Sustainment of the shape and closeness to the original vertices

The aim is to sustain the shape of the original mesh. For obtaining this sustainment the distances of the faired vertices to the original vertices will be minimized too. To preserve a certain degree of freedom for the deviation of the faired mesh the constraint will be included in terms of a penalty function. On the basis of the penalty method the distance of the faired new vertices to the original ones can be controlled by the penalty parameter.

The Penalty Method

The penalty method is a classical method for solving constrained optimization problems [7]. A constrained optimization problem $minf(\mathbf{x})$ under the equality constraints $g_j(\mathbf{x}) = 0, (j = 1, \ldots, p)$ is replaced by a series of unconstrained problems

$$P(\mathbf{x}; \alpha) = f(\mathbf{x}) + \frac{\alpha}{2} \sum_{i=1}^{p} g_j(\mathbf{x})^2,$$

which are formed by adding a term to the objective function that consists of a so-called *penalty parameter* $\alpha > 0$ and a measure $\sum_{i=1}^{p}(g_j(\mathbf{x}))^2$ of violation of the constraints. The part $\sum_{i=1}^{p}(g_j(\mathbf{x}))^2$ punishes high values of $g_j(\mathbf{x})$ and therefore one speaks of a *penalty function*. Within an iterative algorithm α is increased until the constraints are fulfilled sufficiently well. The minimization of $P(\mathbf{x}; \alpha)$ with given α is an unconstrained optimization problem. However, the constraints require a large value of α.

In the new fairing method α will be given and is used as a parameter to control the deviation of the faired mesh to the original one. Translating the penalty method into this fairing ansatz the penalty function is defined as

$$\begin{aligned} f_D(\mathbf{x}) &= \sum_{i=2}^{\#vertices-2} \left\|\mathbf{p}_i - \mathbf{p}_i^0\right\|^2 \\ &= \sum_{i=2}^{\#vertices-2} \left(\mathbf{p}_i - \mathbf{p}_i^0\right)^2 . \end{aligned}$$

The penalty function can be written as

$$\begin{aligned} f_D(\mathbf{x}) &= \sum_{i=2}^{\#vertices-2} \left(\mathbf{C}_i\mathbf{x} - \mathbf{d}_i\right)^2 \\ &= \mathbf{x}^T \sum_{i=2}^{\#vertices-2} \mathbf{C}_i^T\mathbf{C}_i\mathbf{x} - 2 \sum_{i=2}^{\#vertices-2} \mathbf{d}_i^T\mathbf{x} + \sum_{i=2}^{\#vertices-2} \mathbf{d}_i^T\mathbf{d}_i, \end{aligned}$$

where the vector $\mathbf{x}$ is defined as above, the vector $\mathbf{d}_i$ is defined as

$$\mathbf{d}_i := \left\{ \begin{array}{lcl} \mathbf{0} & : & \text{iff the point } \mathbf{p}_i \text{ is fixed} \\ \mathbf{p}_i & : & \text{iff the point } \mathbf{p}_i \text{ is smoothed} \end{array} \right.$$

and $\mathbf{C}_i \in \mathbb{R}^{3\times 3\cdot\#vertices}$ is a zero matrix iff the point $\mathbf{p}_i$ is fixed or

$$\mathbf{C}_i = \left(\begin{array}{ccc|ccc|ccc} 0 & & 0 & 1 & 0 & 0 & 0 & & 0 \\ 0 & \dots & 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & & 0 & 0 & 0 & 1 & 0 & & 0 \end{array}\right),$$

iff the point $\mathbf{p}_i$ is smoothed. The identity matrix in $\mathbf{C}_i$ ranges from column $3(i-1)+1$ to column $3i$.

The objective function of the alternative fairing method can now be defined as

$$\begin{aligned} P(\mathbf{x};\alpha) &= \sum_{i=2}^{\#vertices-2} f_i(\mathbf{x}) + \frac{\alpha}{2} \sum_{i=2}^{\#vertices-2} f_D(\mathbf{x})^2 && (33) \\ &= \sum_{i=2}^{\#vertices-2} \left(\mathbf{A}_i\mathbf{x} + \mathbf{b}_i\right)^T \cdot \left(\mathbf{A}_i\mathbf{x} + \mathbf{b}_i\right)^T + \frac{\alpha}{2} \sum_{i=2}^{\#vertices-2} \left(\mathbf{p}_i - \mathbf{p}_i^0\right)^2 . && (34) \end{aligned}$$

To simplify matters we refer to the penalty parameter as the *weight* $\omega := \frac{\alpha}{2}$ of the distance of the faired vertices to the original vertices. In each iteration step of the fairing method the weight can be chosen arbitrarily.

4.1 The Algorithm

In this section the algorithm is described in a more detailed way than above.

Algorithm

Input: mesh, weight, deviation, number of iteration steps
Output: faired mesh

1. *Find the structure lines of the mesh.*
2. *Each vertex will be categorized as a "fixed" or a "need to be faired" vertex. The deviation parameter indicates the magnitude of the deviation of three consecutive direction vectors of a structure line. If the deviation of the consecutive direction vectors is greater than the deviation parameter then the vertex is characterized as part of a zigzag line.*
3. *Compute the coordinates of the "need to be faired" vertices such that the deviation of three consecutive direction vectors and the penalty function (of the weighted distance of the new points to the input points) is minimal; i.e. compute the minimum of equation (34).*

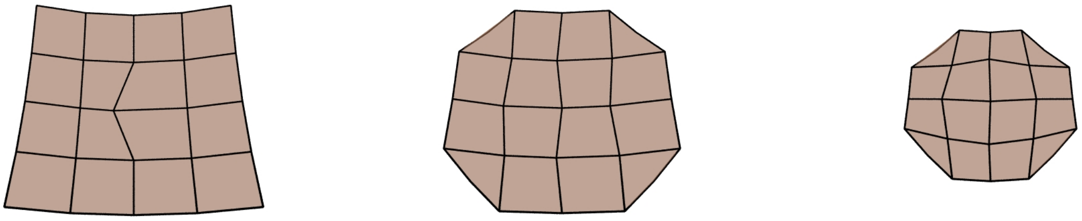
It is possible to assign a category manual to each vertex, i.e. one can fix vertices arbitrary or only move some requested vertices. The four corner vertices are always fixed.

4.2 The Influence of the Parameters

In this section we want to describe the parameters of the Laplacian, the Taubin smoothing and the alternative fairing method in a more detailed way and show some examples of different variations of the specific parameters. The only parameter for the Laplacian and the Taubin fairing method is the number of iterations that do not play a minor role. The changable parameters of the new fairing method are the number of iterations, the deviation of three consecutive direction vectors and the weight of the distances of the new vertices to the original vertices respectively. The deviation and the weight parameter can be changed in each iteration step. The different influences on the mesh with the changing of the parameters are now shown with reference to some examples.

The Number of Iterations

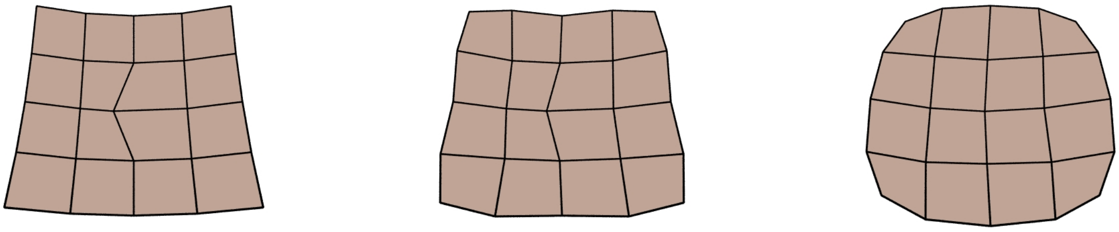
It is quite obvious that the number of iterations plays a major role in a fairing method. For the Laplacian and the Taubin smoothing method the number of iterations declares how smooth the mesh will become; the more iteration steps are performed the more flat the mesh will be. For the Laplacian method the number of iterations should not be too large, because of the (already mentioned) shrinkage effect. As it can be seen in Figure 4.1 the mesh is shrinking vastly after only five iteration steps.



(a) Original **(b)** 1 Iteration **(c)** 5 Iterations

Figure 4.1: Variation of the number of iterations of the Laplacian smoothing method

It seems that the Taubin smoothing method also has an undesirable effect of turning a quadrangular mesh into a "rounded" mesh. The implementation of the Taubin smoothing method which is used in this thesis did not fix the corner vertices. Therefore the corners are cut off and it seems that the mesh gets round. We can ignore this effect and only consider the structure lines without the corner vertices.

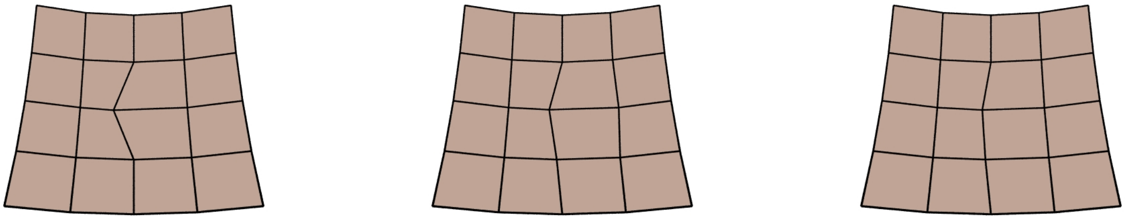


(a) Original **(b)** 1 Iteration **(c)** 10 Iterations

Figure 4.2: Variation of the number of iterations of the Taubin smoothing method

The Interaction between the Number of Iterations and the Magnitude of the Deviation of Three Consecutive Vectors

For the new fairing method the number of iterations is intimately connected with the deviation parameter of three consecutive direction vectors. For each iteration step a deviation parameter can be chosen separately. If the deviation for every three consecutive vectors is smaller than the deviation parameter every vertex will be fixed and the number of iterations does not have any influence on the fairing process. The magnitude of the deviation parameter must therefore be chosen in an appropriate manner so that in each iteration step vertices are moved and that the number of iterations plays a decisive role.

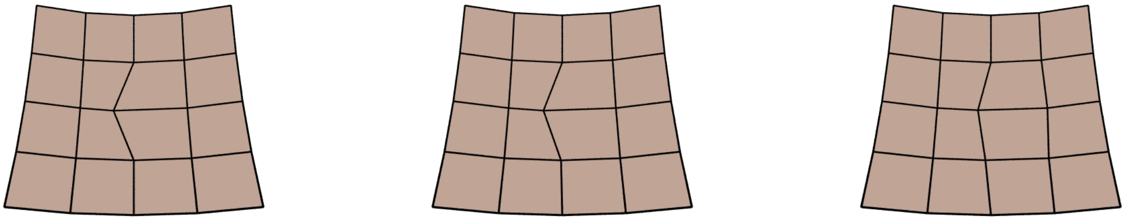


(a) Original **(b)** 1 Iteration **(c)** 10 Iterations

Figure 4.3: Variation of the number of iterations of the alternative fairing method. For this fairing method the deviation parameter is chosen as $0,1$ and the weight is $0,001$.

The Weight

The parameter weight is the penalty parameter of the penalty function of the optimization problem. It controls the distance of the faired to the original vertices. High accuracy in fulfilling the constraints requires a large value of α. Therefore choosing the weight $\omega \gg 1$ the penalty function pulls the vertices close to the original ones. On the contrary for $\omega \approx 0$ the vertices veer away from the original ones and the mesh can possess sizable deformation. The implementation of the alternative fairing method can be extended so that each vertex gets its own weight dependent on the value of the deviation of three consecutive vectors.



(a) Original **(b)** Weight = 1000 **(c)** Weight = 0,0001

Figure 4.4: Variation of the weight of the fairing method. The deviation parameter is chosen as $0,1$ and the number of iterations is one.

5 Examples

In this section the new fairing concept is compared to the well-established smoothing methods of Laplace and Taubin.
As already mentioned the Taubin and Laplacian smoothing methods move each vertex in the barycenter of its 1-ring-neighborhood. Hence the corners are being cut off, because the four corner vertices will be moved in the barycenter of the three surrounding vertices. We disregard this fact in the following examples and compare the remaining mesh anyway.

5.1 Example 1

As the first example we choose a mesh with comparatively few vertices in order to demonstrate the fact that the vertices are moved in the barycenter of the 1-ring-neighborhood and therefore change the characteristics of the structure lines significantly. The surface is similar to a saddle surface. The number of vertices is 100 and the number of faces is 81.

Laplacian Smoothing Method

The number of iterations for the Laplacian smoothing method is 3.
Already after the first iteration step the structure line of the thin strip on the brink of the mesh is moved to the middle of the two neighboring structure lines. This causes a maybe undesirable mesh deformation. After three iterations the structure lines are faired, but also the surface got smoothed through flattened structure lines.

Taubin Smoothing Method

The number of iterations for the Taubin smoothing method with equal weights is 40.
After 40 iteration steps the zigzag lines appear noticeable smoothed. The displacement of the stucture line which was discussed at the Laplacian smoothing method is not as distinctly as it was before. The deformation through the flatten of the surface exists, but is acceptable. More disruptive is the fact that the marginal structure lines move remarkably outwards beyond the original boundary.

New Fairing Concept

The number of iterations for the new fairing concept is 3 and the weight is 0.001. The magnitude of the deviation of three consecutive direction vectors is given by 0.1.
The deformation of the surface is less or equal of the deformation by the Taubin smoothing method. The marginal structure lines are almost identical, except for one part of a structure line which was obviously a zigzag line (near the left corner of the mesh) and therefore allowed to be faired. It would seem that the structure lines got faired along the surface of the mesh.

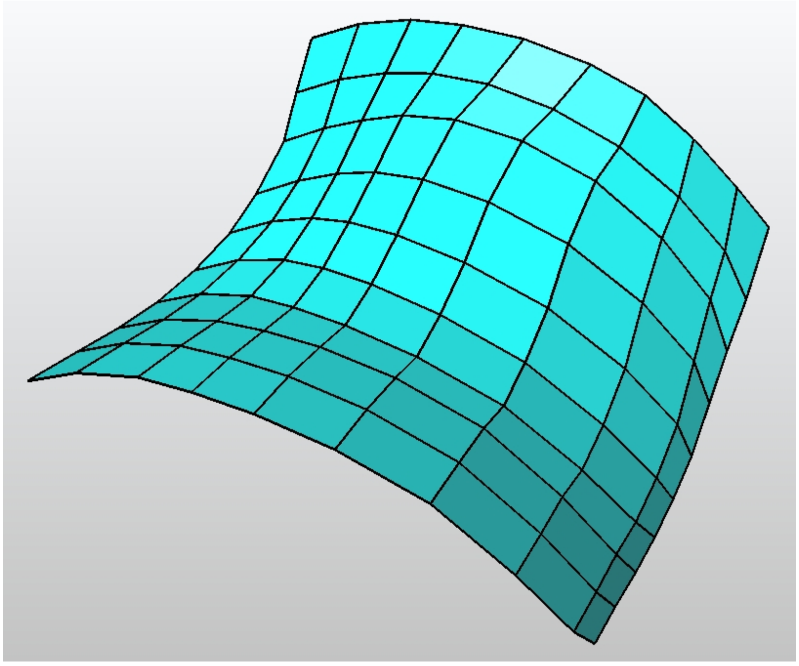


Figure 5.1: The Original Mesh

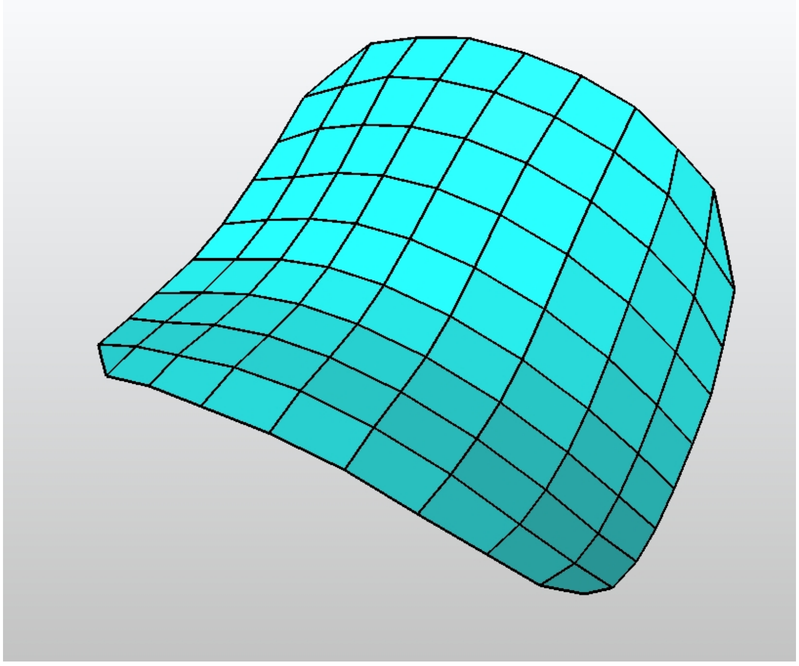


Figure 5.2: The Laplacian Smoothing Method after 3 Iterations

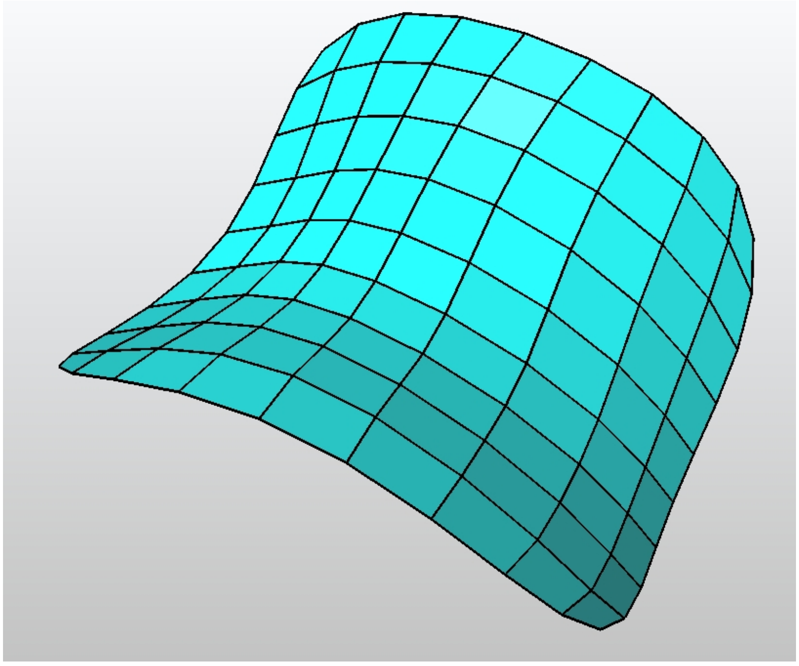


Figure 5.3: The Taubin Smoothing Method after 40 Iterations

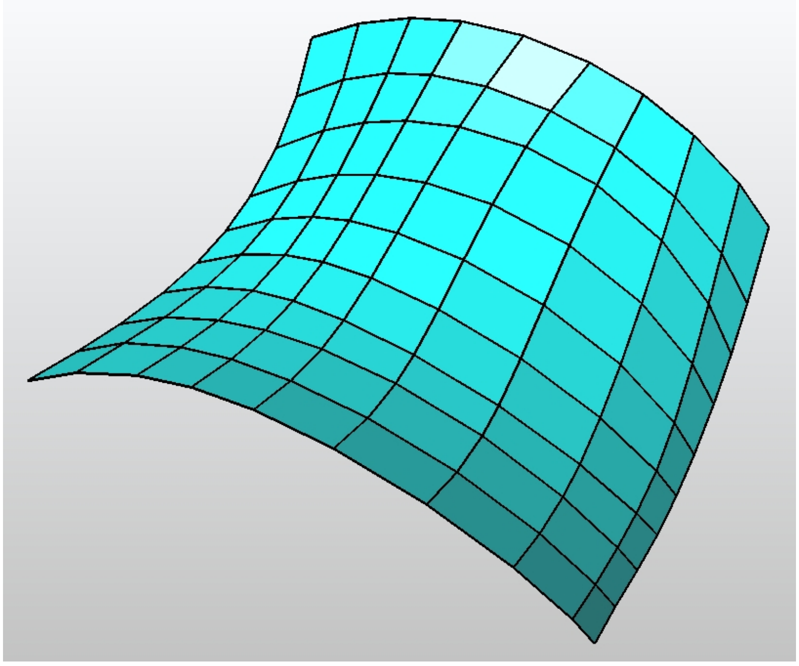


Figure 5.4: The New Fairing Method

5.2 Example 2

The second example is a free-form surface. The number of vertices is 169 and the number of faces is 144.

Laplacian Smoothing Method

The number of iterations for the Laplacian smoothing method is 1.
Already after the first iteration step the s-shape of the surface is flattened and the structure lines move so that they have almost equidistant edges. After the second iteration step the mesh is well smoothed, but it got considerably deformed. After five iterations the mesh is that flattened, that the s-shape of the structure lines is not as remarkable as they were at the beginning.

Taubin Smoothing Method

The number of iterations for the Taubin smoothing method with equal weights is 20.
The boundary structure lines (exept for the four corner vertices) are - in contrary to the first example - identical to the original mesh boundary. The mesh possesses minimal deformation along the s-shaped structure lines. The result is as well faired as the mesh after the application of the method of Laplacian, but does not have the enormous deformation.

New Fairing Concept

The number of iterations for the new fairing concept is 3 and the weight is 0.005. The magnitude of the deviation of three consecutive direction vectors is given by 0.1.
The deformation is as minimal as in the Taubin smoothing process. The boundary structure lines are nearly identical to the original mesh and the curvature of the s-shaped structure lines are almost sustained.

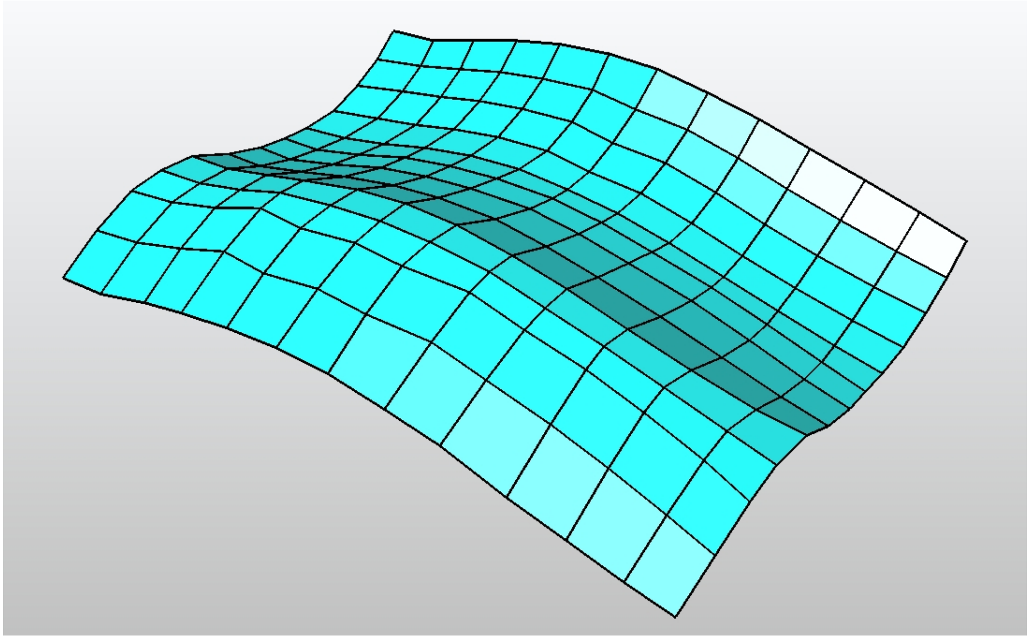


Figure 5.5: The Original Mesh

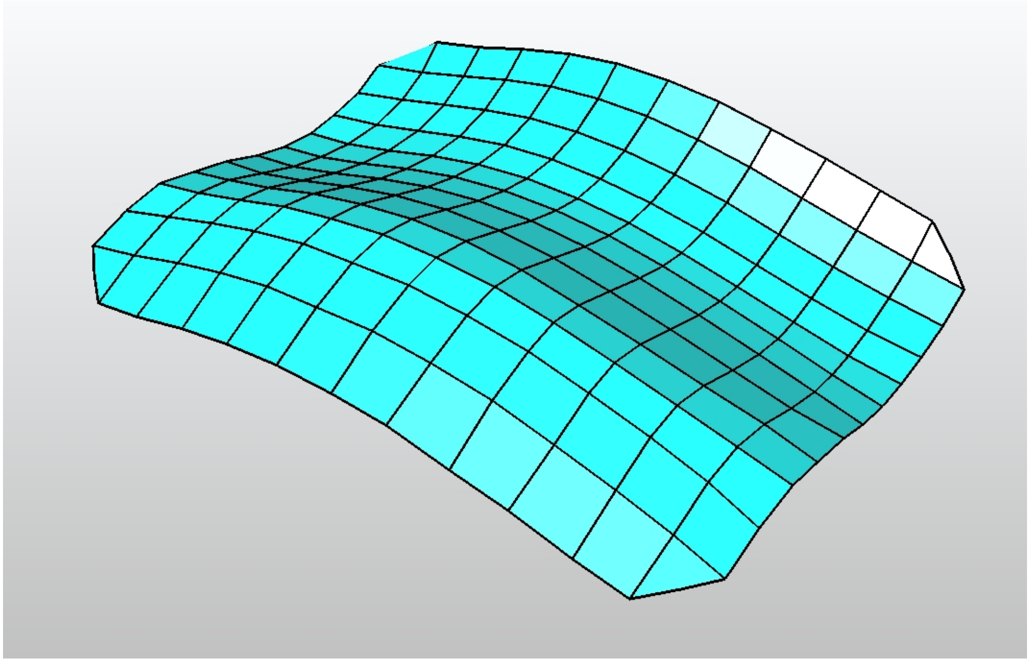


Figure 5.6: The Laplacian Smoothing Method after 1 Iterations

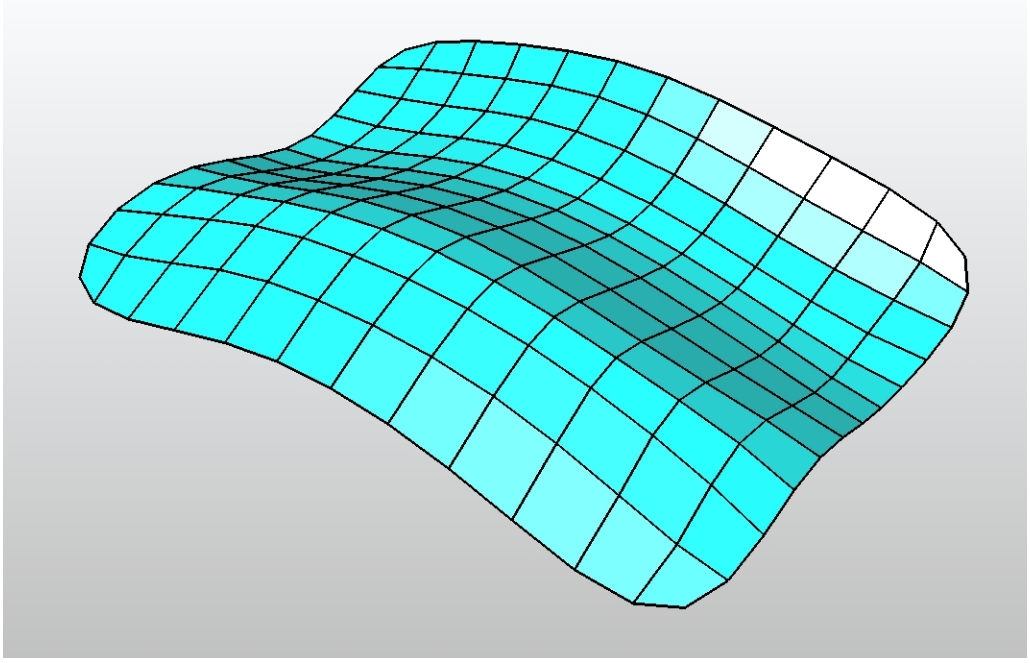


Figure 5.7: The Taubin Smoothing Method after 20 Iterations

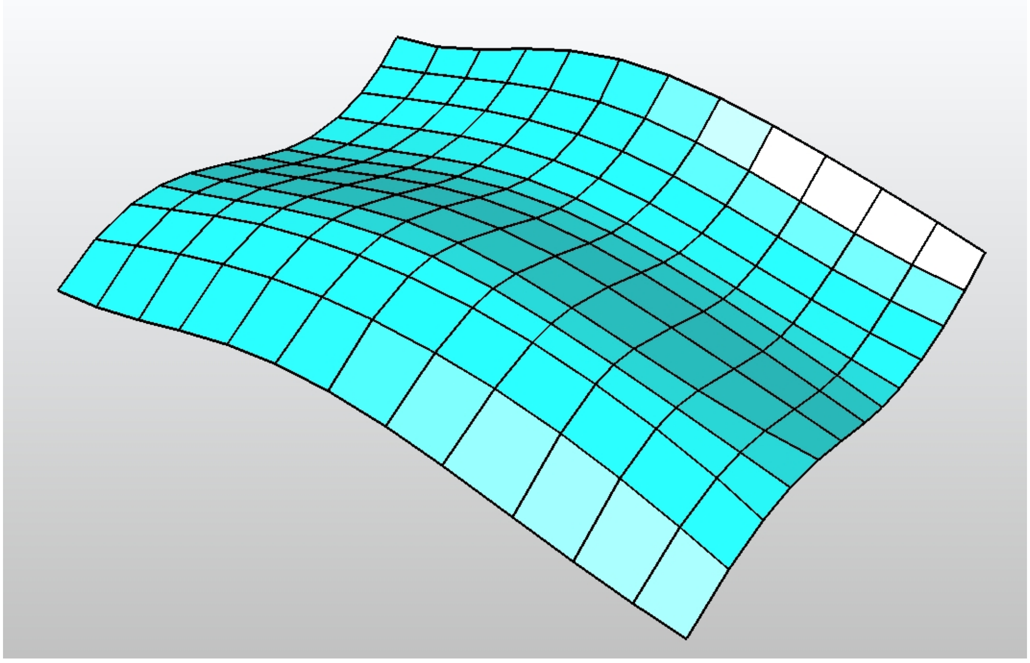


Figure 5.8: The New Fairing Method

5.3 Example 3

The surface of this example is smooth, but the structure lines are zigzag lines. The number of vertices is 902 and the number of faces is 840.

Laplacian Smoothing Method

The number of iterations for the Laplacian smoothing method is 4.
We disregard the fact that the four corners get cut off. The structure lines still possess minimal zigzag lines, but the number of iterations can not be increased anymore because the surface has already started to be flattened. Each iteration step bends the surface such that it is more flat and after four iteration steps there is an apparent deformation of the surface. The structure lines are smoothed, but every vertex is translated so that the smoothed mesh is parallel to the original one. In the region where the curvature changes the smoothed parallel mesh and the original mesh intersects. After ten iteration steps the mesh is shrinking and the connection to the original mesh is not given anymore.

Taubin Smoothing Method

The number of iterations for the Taubin smoothing method with equal weights is 50.
Some structure lines are better faired than with the alternative fairing method and some lines are unsteadier. As with the Laplacian smoothing methods some parts of the mesh are translated in the opposite direction and intersect in the middle of the surface where the curvature changes. The translation is not as distinctive as generated by the Laplacian smoothing method. Near the corners the boundary structure lines overlap the boundary of the original mesh.

New Fairing Concept

The number of iterations for the new fairing concept is 10 and the weight is 0.001. The magnitude of the deviation of three consecutive direction vectors is given by 1 and is divided into halves after each iteration step.
The alternative fairing method neither generates a translation nor possesses any rough deformation of the mesh. The structure lines are sustained without any deformation of the mesh.

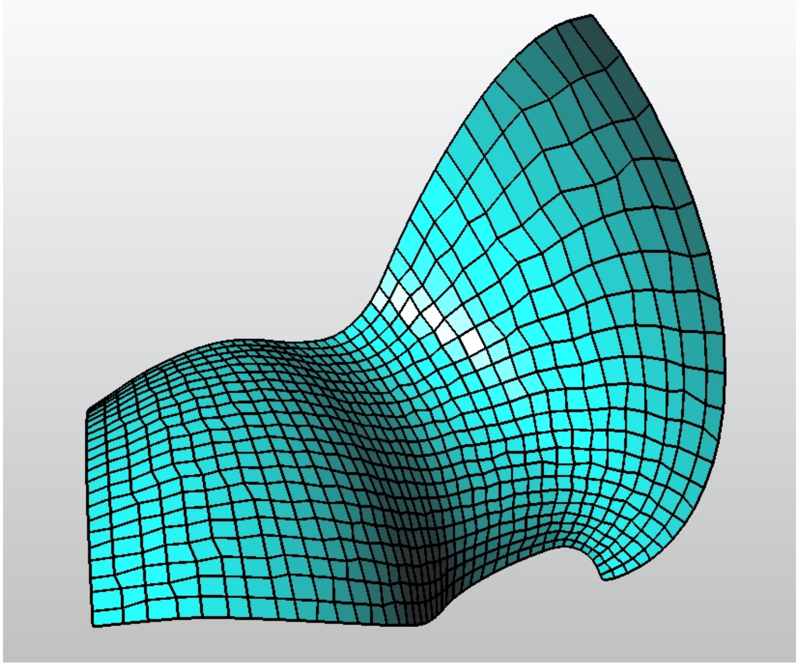


Figure 5.9: The Original Mesh

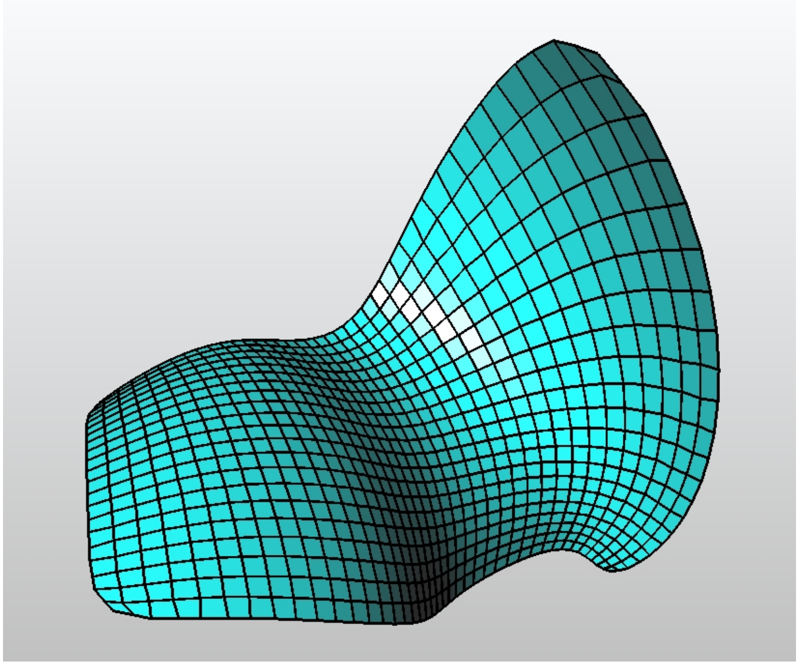


Figure 5.10: The Laplacian Smoothing Method after 4 Iterations

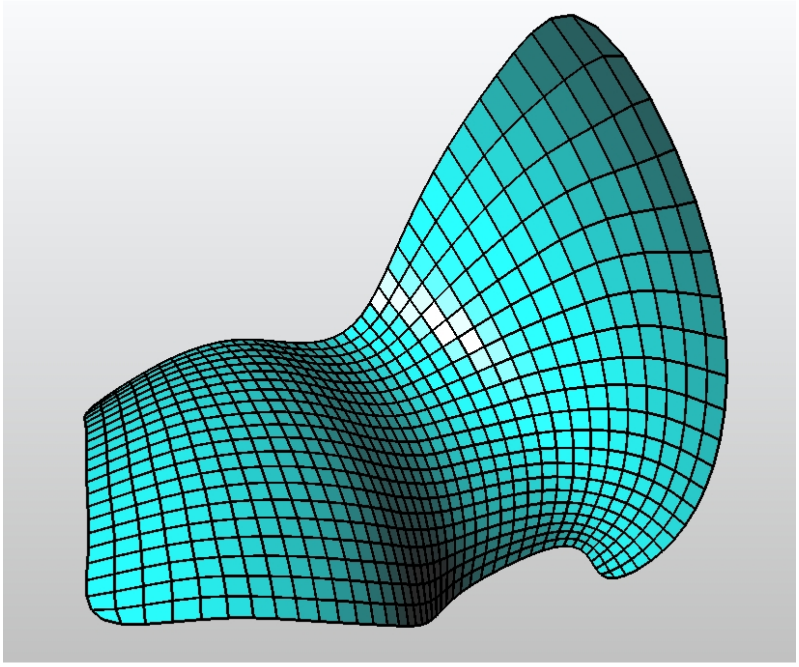


Figure 5.11: The Taubin Smoothing Method after 50 Iterations

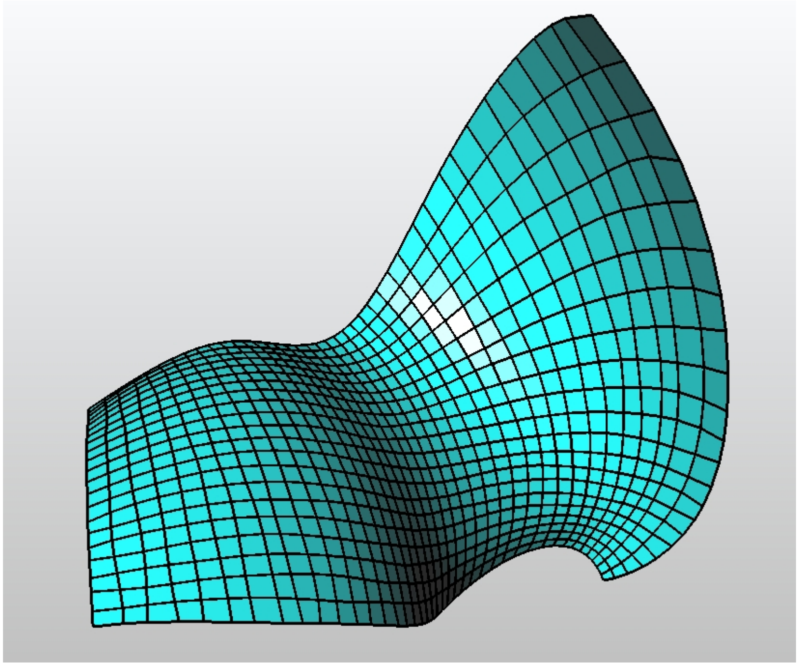


Figure 5.12: The New Fairing Method

5.4 Example 4

The number of vertices is 902 and the number of faces is 840.

Laplacian Smoothing Method

The number of iterations for the Laplacian smoothing method is 4.
Already after four iterations the mesh is shrinking such that the original mesh envelopes the smoothed mesh. The structure lines are smooth but the deviation of the smoothed vertices to the original ones is not maintainable.

Taubin Smoothing Method

The number of iterations for the Taubin smoothing method with equal weights is 30.
Contrary to the Laplacian smoothing method the Taubin smoothing method increases the mesh; the smoothed mesh envelopes the original surface. The part of the mesh with less curvature is flattened and moves away from the original mesh.

New Fairing Concept

The number of iterations for the new fairing concept is 5 and the weight is 0.0005. The magnitude of the deviation of three consecutive direction vectors is given by 1.
The alternative fairing method has neither shrinkage nor growth as the other smoothing methods. Of course there is a deviation to the original mesh, but no remarkable deformation at all. The vertices of the structure lines are moved with small deviation in the surface of the mesh such that the structure lines are fair and the shape is still sustained. The structure lines are perfectly faired and the boundary structure lines are completely preserved too.

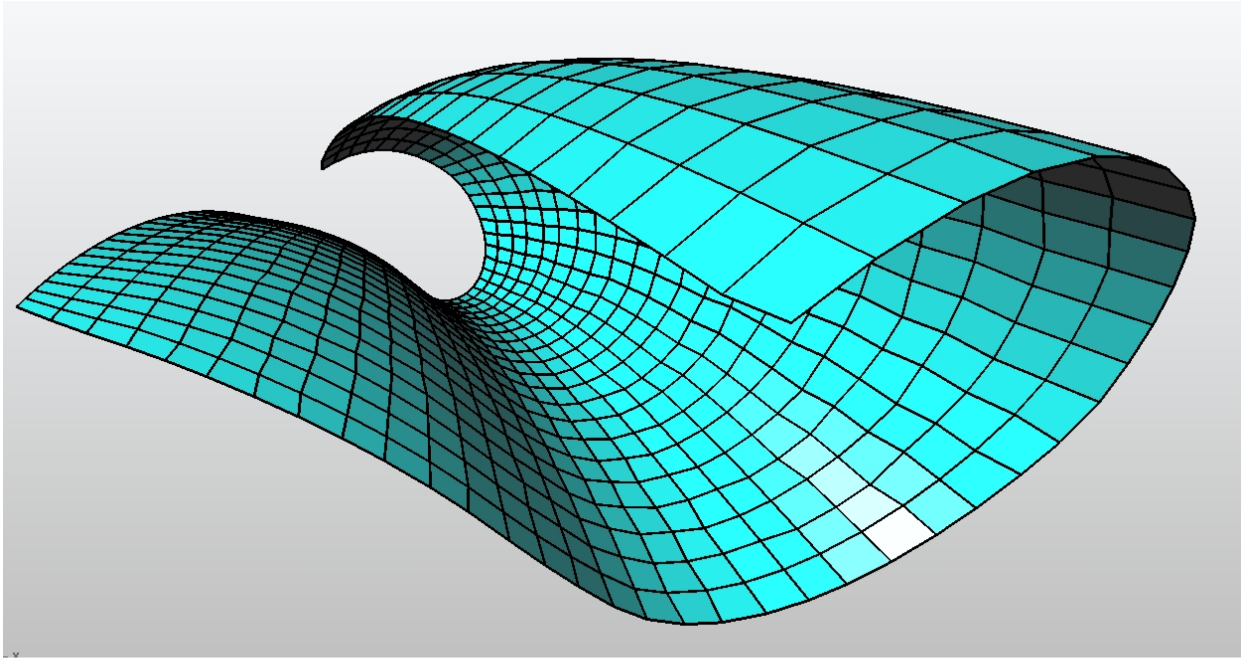


Figure 5.13: The Original Mesh

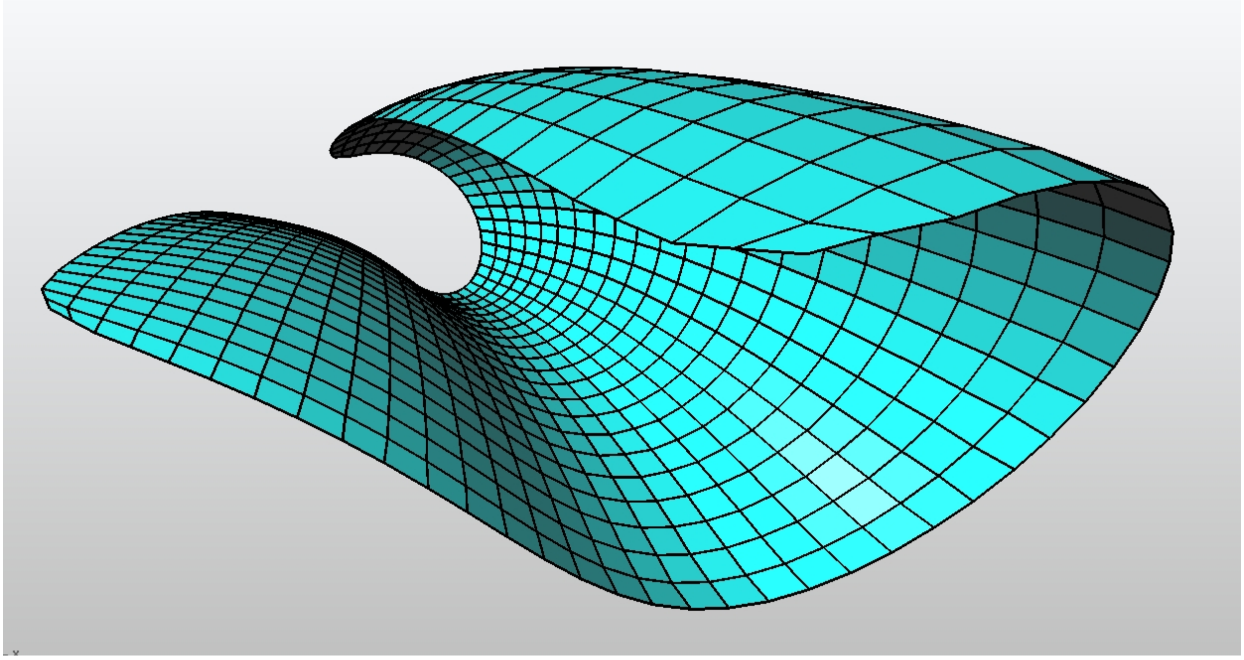


Figure 5.14: The Laplacian Smoothing Method after 4 Iterations

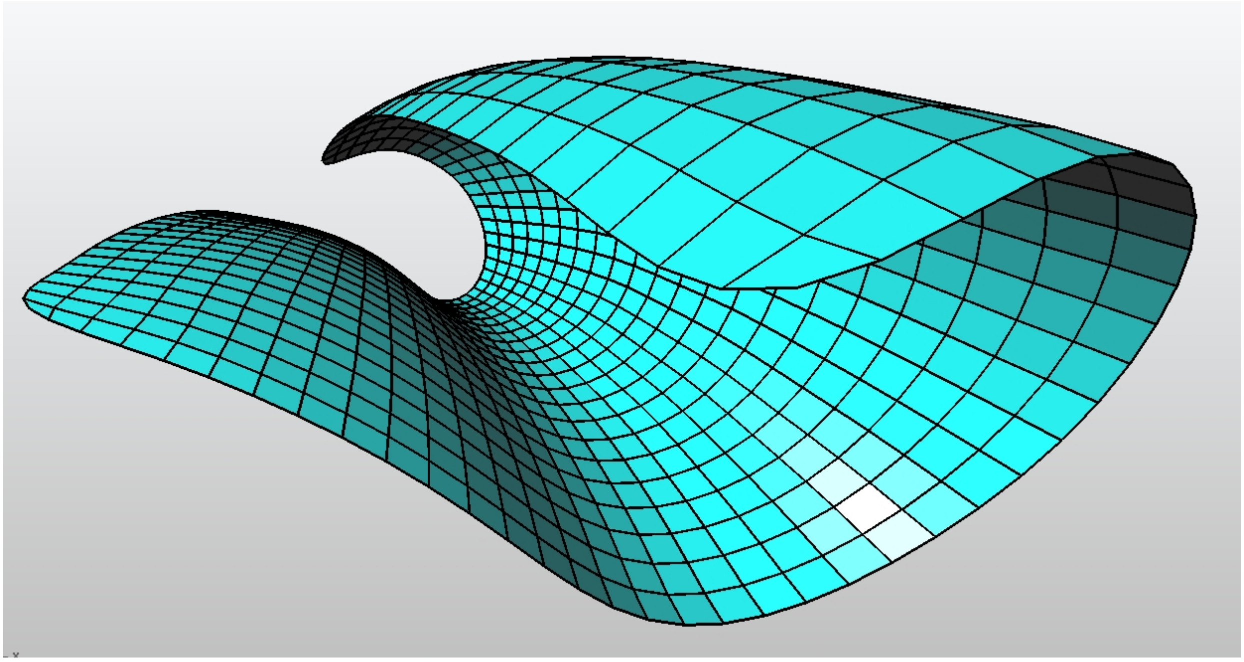


Figure 5.15: The Taubin Smoothing Method after 30 Iterations

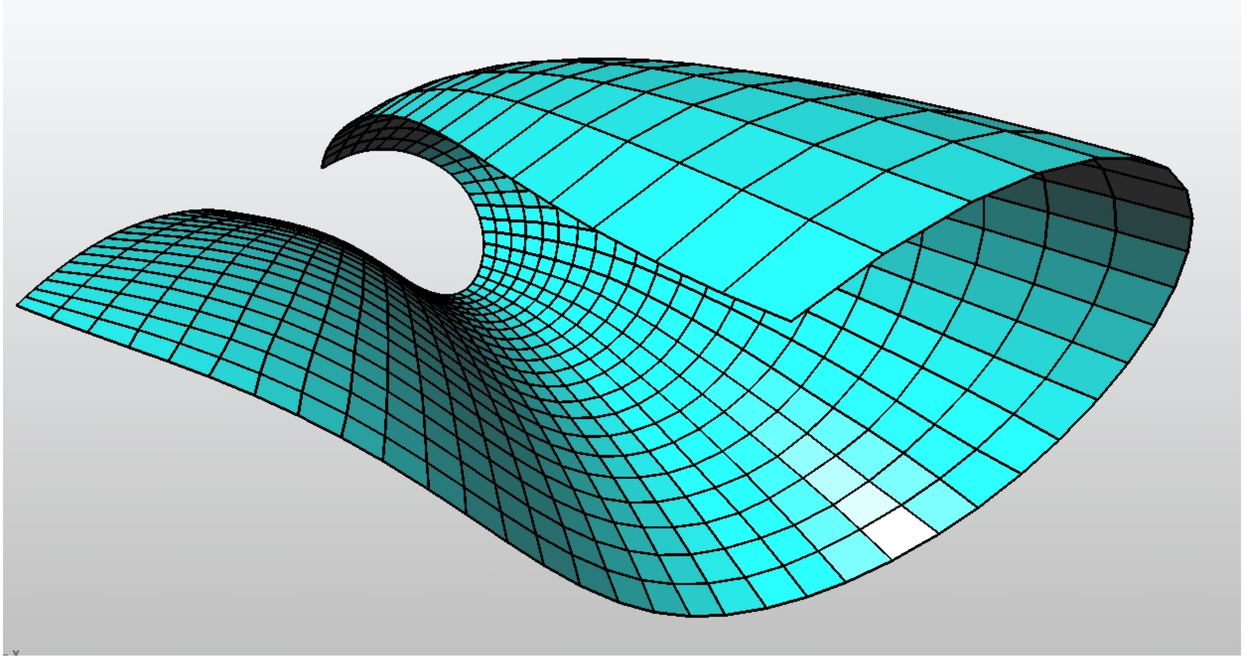


Figure 5.16: The New Fairing Method

6 Conclusion

In this thesis a new method for fairing structure lines of a quadrilateral mesh was presented. The purpose was to get an overview of well-known smoothing and fairing methods before the new concept was introduced. Therefore chapter 3 investigated well-established smoothing methods, eg. minimization of energy functionals, the Laplacian and Taubin smoothing method or the algorithm of Schneider-Kobbelt. Subsequently an alternative fairing method which offers some advantages over the previous methods was presented. The algorithm faired the mesh by minimizing the bending energy of three consecutive direction vectors of a strucutre line. Thereby every vertex is investigated as if it is part of a zigzag line and therefore part of the fairing procedure. The categorization when a line is detected as a zigzag line can be chosen arbitrarily by a parameter of deviation of three consecutive direction vectors. Besides the four boundary vertices which are always fixed, each vertices can be fixed separately too. To sustain the shape of the mesh another parameter which indicates the distance of the faired to the original vertex can be chosen arbitrarily. To underline the aesthetical-appealing qualities of the alternative fairing method chapter 5 has shown some examples. Thereby the meshes which were smoothed with the Laplacian and the Taubin smoothing method were contrasted with the new fairing method.

In terms of future work, investigations to extend this approach can be made to provide a solution not only for quadrilateral meshes, but also triangular meshes and meshes with faces with more than 4 vertices. Because the input meshes have planar faces the integration of a planarity constraint can also be considered.

List of Figures

2.1 Example of a curve network [30] 5
2.2 The Laplace-Beltrami operator Δ_B [3] 13
3.1 Polyline network with tolerance cylinder [11] 20
3.2 Example of surfaces designed using smoothing steps with one interpolartory constraint [28] 31
3.3 Example for different choices of boundary constraints [28] 31
3.4 Displacement of a node [33] 35
3.5 Projection onto the plane and getting the normal curvature direction [20] . 39
4.1 Variation of the number of iterations of the Laplacian smoothing method . 47
4.2 Variation of the number of iterations of the Taubin smoothing method . . 48
4.3 Variation of the number of iterations of the fairing method 48
4.4 Variation of the weight of the alternative fairing method 49
5.1 The Original Mesh 51
5.2 The Laplacian Smoothing Method after 3 Iterations 51
5.3 The Taubin Smoothing Method after 40 Iterations 52
5.4 The New Fairing Method 52
5.5 The Original Mesh 54
5.6 The Laplacian Smoothing Method after 1 Iterations 54
5.7 The Taubin Smoothing Method after 20 Iterations 55
5.8 The New Fairing Method 55
5.9 The Original Mesh 57
5.10 The Laplacian Smoothing Method after 4 Iterations 57
5.11 The Taubin Smoothing Method after 50 Iterations 58
5.12 The New Fairing Method 58
5.13 The Original Mesh 60
5.14 The Laplacian Smoothing Method after 4 Iterations 60
5.15 The Taubin Smoothing Method after 30 Iterations 61
5.16 The New Fairing Method 61

Bibliography

[1] Murat Arikan. Reconstruction of rigid body motions generating 3d geometric texture. Master's thesis, Vienna University of Technology, 2008.

[2] Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. Discrete quadratic curvature energies. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 20–29, New York, NY, USA, 2006. ACM.

[3] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Rössl. Geometric modeling based on polygonal meshes. *Eurographics Tutorial*, 2007.

[4] George Celniker and Dave Gossard. Deformable curve and surface finite elements for free-form shape design. *Proceedings of SIGGRAPH '91*, pages 257–265, 1991.

[5] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *SIGGRAPH Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

[6] Manfredo P do Carmo. *Differentialgeometrie von Kurven und Flächen.* Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, zweite edition, 1992.

[7] Carl Geiger and Christian Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben : [Mit 140 Übungsaufgaben].* Springer-Verlag Berlin Heidelberg New York.

[8] Golub Gene H. and Van Loan Charles F. *Matrix Computations (3rd ed.).* Johns Hopkins University Press, 1996.

[9] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 35–44, 1993.

[10] Michael Hofer and Helmut Pottmann. Energy-minimizing splines in manifolds. volume 23, pages 284–293. ACM Press / ACM SIGGRAPH, 2004.

[11] Michael Hofer, Guillermo Sapiro, and Johannes Wallner. Fair polyline networks for constrained smoothing of digital terrain elevation data. *IEEE Transactions on Geoscience and Remote Sensing*, 44(10):2983–2990, 2006.

[12] Wolfgang Kühnel. *Differentialgeometrie. Kurven - Flächen - Mannigfaltigkeiten.* Friedr. Vieweg & Sohn Verlag/GWV Fachverlag GmbH, Wiesbaden, dritte edition, 2005.

[13] Leif Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13:743–761, 1996.

[14] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graphics*, 25(3), 2006.

[15] Bruno Lévy and Jean-Laurent Mallet. Discrete smooth interpolation: Constrained discrete fairing for arbitrary meshes. *GOCAD Consortium Institut National de Recherche en Informatique et en Automatique. ALICE Geometry and Light*, 1999.

[16] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Proceedings of Visualization and Mathematics*, pages 35–57, 2002.

[17] Henry P. Moreton and Carlo H. Séquin. Functional optimization for fair surface design. *SIGGRAPH Comput. Graph.*, 26(2), 1992.

[18] Ulrich Pinkall, Strasse Des Juni, and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[19] Robert Schneider and Leif Kobbelt. Discrete fairing of curves and surfaces based on linear curvature distribution. In *In Curve and Surface Design: Saint-Malo*, pages 371–380. University Press, 1999.

[20] Robert Schneider and Leif Kobbelt. Generating fair meshes with g^1 boundary conditions. In *GMP '00: Proceedings of the Geometric Modeling and Processing 2000*, 2000.

[21] Robert Schneider and Leif Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design*, 18:359–379, 2001.

[22] Carlo H. Séquin. Cad tools for aesthetic engineering. *Computer-Aided Design & Applications*, 1:301–309, 2005.

[23] John M. Sullivan. Curvature measure for discrete surfaces. In Peter Schröder Ari Stern Mathieu Desbrun, Konrad Polthier, editor, *Discrete Differential Geometry: An Applied Introduction*, pages 10–13, 2006.

[24] John M. Sullivan. Curvatures of smooth and discrete surfaces. In J.M. Sullivan G.M. Ziegler A.I. Bobenko, P. Schröder, editor, *Discrete Differential Geometry*, volume 38, pages 175–188. Oberwolfach Seminars, 2008.

[25] John M. Sullivan. Curves of finite total curvature. In *Discrete Differential Geometry*, volume 38, pages 137–161. Oberwolfach Seminars, 2008.

[26] Gabriel Taubin. Curve and surface smoothing without shrinkage. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, 1995.

[27] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[28] Gabriel Taubin. Geometric signal processing on polygonal meshes. *EUROGRAPHICS 2000 STAR - State of The Art Report*, 2000.

[29] Johannes Wallner, Helmut Pottmann, and Michael Hofer. Fair curve networks in nonlinear geometries.

[30] Johannes Wallner, Helmut Pottmann, and Michael Hofer. Fair webs. *The Visual Computer*, 23(1):83–94, 2007.

[31] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 247–256. ACM, 1994.

[32] Zhiqiang Xu and Guoliang Xu. Discrete schemes for gaussian curvature and their convergence. *Comput. Math. Appl.*, 57(7):1187–1195, 2009.

[33] Atsushi Yamada, Kenji Shimada, Tomotake Furuhata, and Ko hsiu Hou. A discrete spring model for generating fair curves and surfaces. In *In Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications*, pages 270–279, 1999.