TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

DIPLOMARBEIT

A numerical method for calculating discrete surfaces under constraints

Ausgeführt am
Institut für Diskrete Mathematik und Geometrie

im Rahmen des Studiums
Technische Mathematik

unter der Anleitung von
Prof. Martin Peternell
Dr. Martin Kilian

durch
Daniel Herold
Martikelnummer 1325353

Wien, 10.5.2019

(Unterschrift Verfasser)

(Unterschrift Betreuer)

Abstract

In mathematical history the field of discrete differential geometry is a notably young part of interest. It has many applications including computational design, image analysis, geometry processing and more.

In this thesis, we study a relatively new definition of an extended discrete shape operator and its related principal curvatures to optimize discrete surfaces under special constraints, involving these curvatures. One constraint with special interest is the total absolute curvature of a surface $\int_S (|\kappa_1| + |\kappa_2|) dA$. As demonstrated in this thesis, methods which minimize this functional denoise data while preserving features. We will compare surfaces with minimal total absolute curvature to many other surface classes, including minimal surfaces, developable surfaces, Willmore surfaces and more. For computational optimization we introduce an implementation framework based on so-called guided projection that can be used for the presented classes and hypothesizes further useful application.

Kurzfassung

"Ein numerisches Verfahren zur Berechnung diskreter Flächen unter Nebenbedingungen"

In der Geschichte der Mathematik ist die Theorie der diskreten Differentialgeometrie ein junger Forschungszweig. Die Anwendungen gehen von computerunterstütztem Konstruieren und der Analyse von Modellen bis hin zu geometrischer Datenverarbeitung.

In dieser Diplomarbeit untersuchen wir eine relative neue Definition der sogenannten erweiterten diskreten Weingartenabbildung und die damit verbundenen Hauptkrümmungen um damit diskrete Flächen zu optimieren. Diese Flächen sind beschränkt durch spezielle Nebenbedingungen, die von diesen Hauptkrümmungen abhängen. Eine dieser Nebenbedingungen von besonderem Interesse, ist die totale Absolutkrümmung einer Fläche $\int_S (|\kappa_1| + |\kappa_2|) dA$. Wie in der Arbeit gezeigt wird verringern Methoden, die dieses Integral minimieren Störungen in den Daten, aber erhalten gleichzeitig Kanten. Flächen mit minimaler totaler Absolutkrümmung werden verglichen mit anderen Klassen, wie zum Beispiel Minimalflächen, abwickelbaren Flächen und Willmore Flächen. Für den Optimierungsprozess verwenden wir die sogenannte "guided projection" Methode, deren Ausführung für alle gezeigten Beispiele verwendet werden kann und zukunftsträchtige Anwendung verspricht.

Acknowledgements

I want to thank my advisor Martin Kilian for the many times I could show up in his office without prior notice, just telling him about more bugs I encountered, as well as Davide Pellis who gave me advice without even questioning who I actually am. Also I want to thank Martin Peternell for a flawless supervision.

Many thanks to my family and friends, who all had a hard time with my ultimate excuse "thesis first", especially Andrea, who always had an open ear for my countless problems. Also cheers to my study colleagues for an awesome time at this awesome university. And of course, a huge thank-you to my T460p for executing the same code over and over again without getting bored and never letting me down.

Contents

1 Introduction

Motivation

The emerging field of discrete differential geometry is strongly related to computer science. New technologies in 3D scanning, more powerful computers and other development provide new and varying application of methods based on discrete differential geometry. 3D images can be analyzed with geometrical interpretation and noise in data may be eliminated (figure 1.1). Users want smooth surfaces, while features should be preserved. All this applications and more can be summarized under the term of geometric data processing. On the one hand the number of possible utilizations seems endless, on the other hand the underlying theory still needs a lot of adaption and is under constant research.

This thesis aims to address both of this problems. Discrete analogies to smooth definitions of surface curvatures are presented. These operators can be used to estimate quantities and actively optimize surfaces to fulfill certain properties. In practice of computational design architectural drafts can be derived which have specific characteristics, from visual and statical perspective (figure 1.2).

With the underlying theory, the main part of this thesis is the implementation of an iteratively solver for such specific problems. We focus on standard surface classes, occurring in different applications and optimize initial input data to fulfill some objective functionals. The main advantage of our work is the generality of the setup, which is able to deal with different targets.

One rather new idea is to define the total absolute curvature energy of a surface and optimize meshes towards a minimum of this functional. Compared to other methods, like the Willmore energy which deforms a mesh locally towards a sphere, the idea results in less chubby solutions that preserves sharp edges. Our model is able to deal with this functional in a natural way.

Figure 1.1: The model of a bunny. Large estimated curvatures of the surface are colored purple. http://graphics.stanford.edu/data/3Dscanrep/

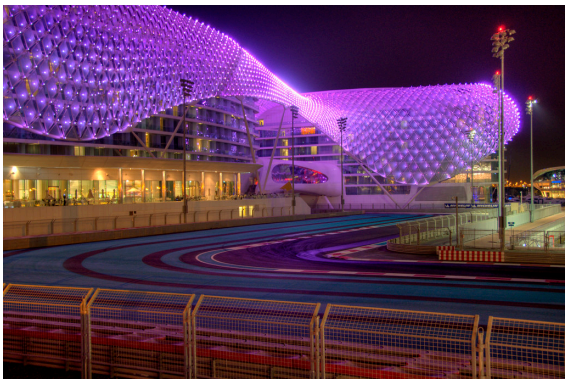


Figure 1.2: The Yas Marina Hotel in Abu Dhabi with a torsion-free support structure.
Rob Alter, https://creativecommons.org/licenses/by/2.0/

Related work

Energy minimizing functionals have been considered for minimal surfaces [31], developable surfaces [37, 22] and Willmore surfaces [4]. In [29] they use a definition of the extended discrete shape operator from [8] to align principal stress and curvature directions. This idea, based on normal cycles, is also considered in this thesis to deal with surfaces under different side conditions. To optimize a mesh using the so-called guided projection is processed in [36]. Finally the question of a discrete surface with minimal total absolute curvature was raised in [23].

Overview

To work round to the goal of optimizing and deriving surfaces under special conditions, we structure this thesis as follows. We start with an introduction to the basic concepts of differential geometry and describe some special surfaces formally. Built on the theory for the smooth case, we define basic concepts of the field of discrete differential geometry and give various methods of definitions of discrete curvatures. Chapter 3 deals with basics of optimization, including least squares problems and linear systems, which arise in chapter 4. In this part we describe our general setup and give information about the specific implementation of the problem. This includes a description and derivation of the guided projection method. With the necessary tools in hand, we apply our method to various examples of different surface classes, including minimal surfaces, developable surfaces, Willmore surfaces and total absolute curvature minimizing surfaces. We compare the results as well as difficulties in the process and give further prospects in the final chapter 6.

2 Differential geometry

The main goal of this chapter is to give a brief introduction to the field of classic differential geometry and approaches how to discretize the achievements of the smooth case. In section 2.1 we will start with standard definitions of curves and surfaces and give major theorems about different kind of curvatures, besides illustrations and interpretations of the different quantities. In section 2.2 basic ideas for discrete versions of the smooth case are presented and discussed. The goal is to derive all necessary preliminaries, to build an implementation for various optimization goals arising also in this chapter.

2.1 Basics of Differential geometry

This section aims to define the most important objects and propose some of the main results of the large area of differential geometry briefly. This wide field of research can only be partially covered and proofs and intermediate steps are often skipped. In general we refer to standard literature like [6], [2] and [24] for more details.

2.1.1 Curves and Surfaces

We start with some basic definitions, beginning with curves.

Definition 2.1. A *parametrized differentiable curve* is a differentiable mapping $\boldsymbol{c}$ of an open interval I to the 3-dimensional space

$$\boldsymbol{c} : \mathbb{R} \supseteq I \to \mathbb{R}^3 : t \mapsto \boldsymbol{c}(t) = (x(t), y(t), z(t))$$

The vector $\boldsymbol{c}'(t) = (x'(t), y'(t), z'(t)) \in \mathbb{R}^3$ is called the *tangent vector* of $\boldsymbol{c}$ at t. The image $\boldsymbol{c}(I) \subseteq \mathbb{R}^3$ is the *trace* of $\boldsymbol{c}$.

The condition *differentiable* in the definition above and all following definitions in this chapter means *differentiable as often as necessary* e.g. C^2. In the following we will mostly require continuous second order derivatives.

The tangent vector $\boldsymbol{c}'(t)$ defines a straight line (containing the point $\boldsymbol{c}(t)$) only if $\boldsymbol{c}'(t) \neq \mathbf{0}$. Points with $\boldsymbol{c}'(t) = \mathbf{0}$ are called *singular points*. The demand for curves without singular points in differential geometry leads to

Definition 2.2. A parametrized differentiable curve $\boldsymbol{c} : I \to \mathbb{R}^3$ is *regular* if $\boldsymbol{c}'(t) \neq \mathbf{0}\ \forall t \in I$.

For regular curves, by definition the *arc length* of a curve is $s(t) = \int_{t_0}^{t} |\boldsymbol{c}'(\tau)\ d\tau$ with $\frac{ds}{dt} = |\boldsymbol{c}'(t)| \neq 0$. By *reparametrization* we can obtain a curve with $|\boldsymbol{c}'(s)| \equiv 1$. W.l.o.g. we will restrict ourselves to those *arc length parametrized curves*.

Definition 2.3. For an arc length parametrized curve $\boldsymbol{c} : I \to \mathbb{R}^3$ the number $\kappa(s) := |\boldsymbol{c}''(s)|$ is called the *curvature* of $\boldsymbol{c}$ at s. For $\kappa(s) \neq 0$ the unit vector $\boldsymbol{n}(s) := \boldsymbol{c}''(s)/\kappa(s)$, named the *normal vector*, is orthogonal[(1)] to the tangent vector $\boldsymbol{c}'(s)$ and together they span a plane, the *osculating plane* at s.

Similarly to curves we can define parametrized surfaces.

Definition 2.4. A subset $S \subseteq \mathbb{R}^3$ is called a *parametrized surface* if for each $q \in S$, there exists an open set $U \subset \mathbb{R}^2$ and an open neighborhood $V \subset \mathbb{R}^3$ of q and a differentiable bijection

$$\boldsymbol{x} : U \to V \cap S : (u, v) \mapsto \boldsymbol{x}(u, v) = (x(u, v), y(u, v), z(u, v))$$

If the differential $d\boldsymbol{x}_p : \mathbb{R}^2 \to \mathbb{R}^3$ is one-to-one for all $p \in U$ (i.e. the vectors $\frac{\partial \boldsymbol{x}}{\partial u}, \frac{\partial \boldsymbol{x}}{\partial v}$ are linearly independent for all $p \in U$) the surface is said to be *regular*.

The function $\boldsymbol{x}$ is called the *parametrization* of the surface S. Some regular surfaces can be covered with a single parametrization, while others cannot (e.g. the sphere).

Proposition 2.5. *For a regular surface the vectors*

$$\boldsymbol{x}_u(q) := \frac{\partial \boldsymbol{x}}{\partial u}(q), \boldsymbol{x}_v(q) := \frac{\partial \boldsymbol{x}}{\partial v}(q)$$

form a basis of the tangent plane $T_q(S)$ *at every point $q = \boldsymbol{x}(p) \in S$, as shown in [6].*

For a given point $q \in S$ a curve $\boldsymbol{c} : \mathbb{R} \supset I \to S$ on the surface through $q = \boldsymbol{c}(t_0)$ can be written as $\boldsymbol{c} = \boldsymbol{x} \circ \beta$ with $\beta : I \to U \subset \mathbb{R}^2 : t \mapsto \beta(t) = (u(t), v(t))$ and $\boldsymbol{c}(t_0) = \boldsymbol{x}(\beta(t_0)) = q$ respectively. β is a curve in the parameter set U of S. The curve $\boldsymbol{c}$ has a tangent vector at $c(t_0) = q$ in the tangent plane of the surface with coordinates in the basis $\boldsymbol{x}_u, \boldsymbol{x}_v$:

$$\boldsymbol{c}'(t_0) = \boldsymbol{x}_u u'(t_0) + \boldsymbol{x}_v v'(t_0)$$

2.1.2 First and second fundamental form and shape operator

Restricted to the tangent plane $T_q(S)$ the natural inner product $\langle \cdot, \cdot \rangle_q : T_q(S)^2 \to \mathbb{R}$ is linear and symmetric and induces a quadratic form.

Definition 2.6. For $q \in S$ the *first fundamental form* $\mathrm{I}_q : T_q(S) \to \mathbb{R}$ is defined by

$$\mathrm{I}_q(\boldsymbol{w}_1, \boldsymbol{w}_2) := \langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle$$

For vectors in the standard basis $\{\boldsymbol{x}_u, \boldsymbol{x}_v\}$ we can associate $\mathrm{I}_q(\boldsymbol{w}_1, \boldsymbol{w}_2) = \boldsymbol{w}_1^T \mathrm{I}(q) \boldsymbol{w}_2$ with the symmetric matrix $\mathrm{I}(q) = \left(\begin{smallmatrix} E & F \\ F & G \end{smallmatrix}\right)$ with coefficients

$$E := \langle \boldsymbol{x}_u, \boldsymbol{x}_u \rangle \qquad F := \langle \boldsymbol{x}_u, \boldsymbol{x}_v \rangle \qquad G := \langle \boldsymbol{x}_v, \boldsymbol{x}_v \rangle$$

(1) The orthogonality can be derived via differentiation of the equation $\boldsymbol{c}'(s) \cdot \boldsymbol{c}'(s) = 1$. This trick can be used in similar ways for many further results.

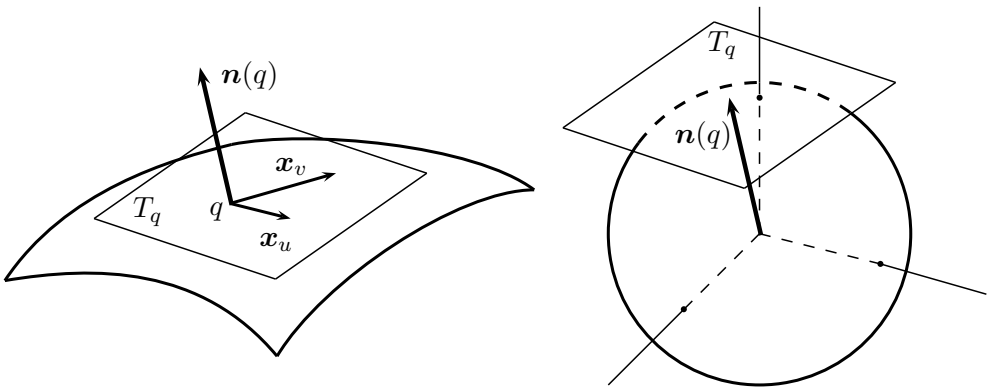


Figure 2.1: Gauß map $\boldsymbol{n}(q)$ and tangent plane T_q, spanned by $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ on the surface (left) and correspondingly on the sphere (right).

The index q indicates that the first fundamental form depends on the position on the surface. In particular the matrix I is generally different for distinct points on the surface. Often this index will be skipped.

With the first fundamental form it is possible to measure lengths, angles and areas on the surface. For instance the area element can be expressed in terms of $\mathrm{I}(q)$ using Lagrange's identity.

$$dA := |\boldsymbol{x}_u \times \boldsymbol{x}_v| du\, dv = \sqrt{\langle \boldsymbol{x}_u, \boldsymbol{x}_u\rangle\langle \boldsymbol{x}_v, \boldsymbol{x}_v\rangle - \langle \boldsymbol{x}_u, \boldsymbol{x}_v\rangle^2} du\, dv = \sqrt{EG - F^2} du\, dv \tag{2.1}$$

Proposition 2.7. *The definition of the area $A(\boldsymbol{x}(U)) = \int_U dA$ is independent of the choice of parametrization $\boldsymbol{x}$.*

Proof. Let $\tilde{\boldsymbol{x}}$ be another parametrization from $\widetilde{U}$ onto the same region of the surface. Let $\frac{\partial(u,v)}{\partial(\tilde{u},\tilde{v})}$ be the Jacobian of the change of parameters $h = \boldsymbol{x}^{-1} \circ \tilde{\boldsymbol{x}}$. Using the theorem of change of variables in multidimensional integrals we obtain

$$\begin{aligned}\int\int_{\widetilde{U}} |\tilde{\boldsymbol{x}}_u \times \tilde{\boldsymbol{x}}_v| d\tilde{u}\, d\tilde{v} &= \int\int_{\widetilde{U}} |\boldsymbol{x}_u \times \boldsymbol{x}_v| \left|\frac{\partial(u,v)}{\partial(\tilde{u},\tilde{v})}\right| d\tilde{u}\, d\tilde{v} \\ &= \int\int_U |\boldsymbol{x}_u \times \boldsymbol{x}_v| du\, dv\end{aligned}$$

□

Definition 2.8. For a fixed parametrization and a point $q \in \boldsymbol{x}(U) \subset S$ we can define the unit normal vector (which is independent from the parametrization, up to orientation) by

$$\boldsymbol{n}(q) := \frac{\boldsymbol{x}_u \times \boldsymbol{x}_v}{|\boldsymbol{x}_u \times \boldsymbol{x}_v|}$$

A surface is called *orientable* if this function $\boldsymbol{n} : S \to S^2 \subset \mathbb{R}^3$ can be extended to a differentiable function on the whole surface S. On an orientable surface the function $\boldsymbol{n} : S \to S^2$, mapping each point to the unit sphere, is called the *Gauß map* of S.

Some surfaces are not orientable, as the example of the Möbius strip shows.

The differential of $\boldsymbol{n}$ at a point $q \in S$ is $d\boldsymbol{n}_q$. Since the tangent planes of the surface in q and the corresponding point $\boldsymbol{n}(q)$ on the sphere are parallel, $d\boldsymbol{n}_q$ can be seen as a linear function from $T_q(S) \to T_q(S)$.

Definition 2.9. For $q \in S$ the linear function $d\boldsymbol{n}_q : T_q(S) \to T_q(S)$ is named the *shape operator* or *Weingarten map*.

The shape operator is self-adjoint with respect to I_q and defines therefore a quadratic form (see [6, 25]).

Definition 2.10. The quadratic form $\mathrm{II}_q : T_q(S) \to \mathbb{R}$ defined by

$$\mathrm{II}_q(\boldsymbol{w}) = -\langle d\boldsymbol{n}_q(\boldsymbol{w}), \boldsymbol{w}\rangle$$

is called the *second fundamental form*. With respect to the basis $\{\boldsymbol{x}_u, \boldsymbol{x}_v\}$ the second fundamental form can be associated with a symmetric matrix $\mathrm{II}(q) := \left(\begin{smallmatrix} e & f \\ f & g \end{smallmatrix}\right)$

$$e := \langle \boldsymbol{n}, \boldsymbol{x}_{uu}\rangle \qquad f := \langle \boldsymbol{n}, \boldsymbol{x}_{uv}\rangle \qquad g := \langle \boldsymbol{n}, \boldsymbol{x}_{vv}\rangle$$

As in case of the first fundamental form, the index q might be skipped. The negative linear shape operator can also be seen as a matrix W applied to a vector in the basis $\{\boldsymbol{x}_u, \boldsymbol{x}_v\}$. We use W as the matrix for the negative shape operator $-d\boldsymbol{n}$, because of definitions in following sections. With the matrices for I and II we get

$$\mathrm{W} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} e & f \\ f & g \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1}$$

and the coefficients are

$$w_{11} = \frac{eG - fF}{EG - F^2} \qquad w_{12} = \frac{fE - eF}{EG - F^2}$$
$$w_{21} = \frac{fF - gF}{EG - F^2} \qquad w_{22} = \frac{gE - fF}{EG - F^2}$$

2.1.3 Principal curvatures, mean and Gaussian curvature

We will give an interpretation of II after the following

Definition 2.11. For a regular curve $\boldsymbol{c}$ on S with normal vector $\boldsymbol{n}_{\boldsymbol{c}}$ and curvature κ at $q \in S$ the number

$$\kappa_n(c, q) := \langle \kappa \boldsymbol{n}_{\boldsymbol{c}}, \boldsymbol{n}(q)\rangle$$

is the *normal curvature* of $\boldsymbol{c}$ at q.

The normal curvature is the projection of the curvature of a curve onto the normal of the surface (figure 2.2). For a regular curve $\boldsymbol{c}$ on the surface the second fundamental form applied to a tangent vector $\boldsymbol{c}'(t)$ is

$$\mathrm{II}_{\boldsymbol{c}(t)}(\boldsymbol{c}'(t)) = \kappa_n(c, c(t))$$

The normal curvature only depends on the tangent vector of a curve, and all curves with the same tangent vector have the same normal curvature.

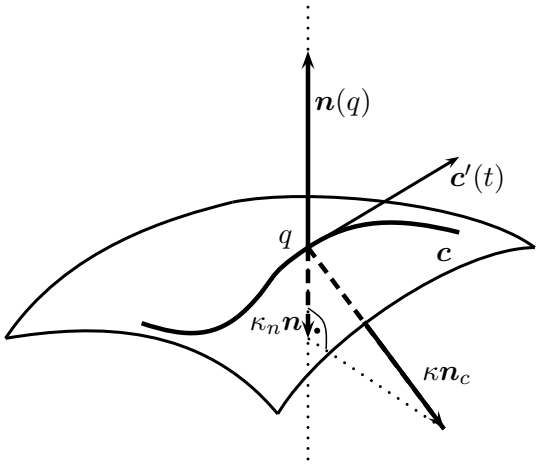


Figure 2.2: The normal curvature $\kappa_n = \langle \kappa \boldsymbol{n}_c, \boldsymbol{n}(q) \rangle$ for the curve $\boldsymbol{c}$ in point q.

As shown in [6] the tangent plane $T_q(S)$ has an orthonormal basis $\{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ such that for the (linear) shape operator $d\boldsymbol{n}_q$ the basis vectors are eigenvectors to the eigenvalues κ_1, κ_2. The eigenvalues are the maximum and minimum values of II_q restricted to the unit circle on the tangent plane, see [25].

Definition 2.12. The eigenvalues κ_1, κ_2 of the shape operator $d\boldsymbol{n}_q$ are called the *principal curvatures* at q and the orthogonal eigenvectors $\boldsymbol{a}_1, \boldsymbol{a}_2$ the *principal directions* at q.

$$d\boldsymbol{n}_q(\boldsymbol{a}_1) = -\kappa_1 \boldsymbol{a}_1 \qquad\qquad d\boldsymbol{n}_q(\boldsymbol{a}_2) = -\kappa_2 \boldsymbol{a}_2$$

In matrix notation this is the solution of the generalized eigenvalue problem

$$\begin{pmatrix} e & f \\ f & g \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \kappa \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

The principal curvature directions vectors are not unique, but the directions are (iff the eigenvalues do not coincide). Therefore we will assume $\boldsymbol{a}_1, \boldsymbol{a}_2$ to be of unit length 1.

Theorem 2.13. *For a vector in the tangent plane with coordinates in the new (orthogonal) basis* $\boldsymbol{w} = \cos\theta \boldsymbol{a}_1 + \sin\theta \boldsymbol{a}_2$ *we get* Euler's formula*:*

$$\mathrm{II}(\boldsymbol{w}) = \cos^2\theta \kappa_1 + \sin^2\theta \kappa_2$$

Proof. For $\boldsymbol{w}$ defined as above we have

$$\begin{aligned}
\mathrm{II}_q(\boldsymbol{w}) &= -\langle d\boldsymbol{n}_q(\boldsymbol{w}), \boldsymbol{w} \rangle \\
&= -\langle d\boldsymbol{n}_q(\cos\theta \boldsymbol{a}_1 + \sin\theta \boldsymbol{a}_2), \cos\theta \boldsymbol{a}_1 + \sin\theta \boldsymbol{a}_2 \rangle \\
&= -\langle \cos\theta d\boldsymbol{n}_q(\boldsymbol{a}_1) + \sin\theta d\boldsymbol{n}_q(\boldsymbol{a}_2), \cos\theta \boldsymbol{a}_1 + \sin\theta \boldsymbol{a}_2 \rangle \\
&= -\langle \cos\theta(-\kappa_1)\boldsymbol{a}_1 + \sin\theta(-\kappa_2)\boldsymbol{a}_2, \cos\theta \boldsymbol{a}_1 + \sin\theta \boldsymbol{a}_2 \rangle \\
&= \cos^2\theta \kappa_1 + \sin^2\theta \kappa_2
\end{aligned}$$

□

Definition 2.14. Half of the trace of the negative shape operator $-d\boldsymbol{n}_q$ is called the *mean curvature* H of S and the determinant of $-d\boldsymbol{n}_q$ is the *Gaussian curvature* K of S. In terms of the principal curvatures we get

$$H := \frac{\kappa_1 + \kappa_2}{2} \qquad\qquad K := \kappa_1 \cdot \kappa_2$$

With the coefficients of the matrix W of the shape operator we obtain the following identities

$$\begin{aligned} H = \frac{1}{2}\operatorname{trace}(\mathrm{W}) &= \frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2} \\ K = \det(\mathrm{W}) &= \frac{eg - f^2}{EG - F^2} \\ \kappa_{1,2} &= H \pm \sqrt{H^2 - K} \end{aligned}$$

Another interpretation for the Gaussian curvature is the following identity. For a ball $B_\varepsilon(q)$ of radius ε around $q \in S$ let $V_\varepsilon := B_\varepsilon(q) \cap S$. Then

$$|K(q)| = \lim_{\varepsilon \to 0} \frac{\operatorname{area}(\boldsymbol{n}(V_\varepsilon))}{\operatorname{area}(V_\varepsilon)}$$

An interesting fact is also this theorem by Gauß.

Theorem 2.15 (theorema egregium)**.** *The Gaussian curvature K is an intrinsic invariant, it is invariant under local isometry. K depends only on the coefficients of the first fundamental form.*

For proofs of the equations and identities we refer to standard literature about differential geometry, such as [6].

2.1.4 Extended shape operator

The shape operator is only defined in the tangent plane T_q but can be extended to $\mathbb{R}^3$ via $d\boldsymbol{n}_q(\boldsymbol{n}(q)) = \mathbf{0}$, as considered in [38].

Proposition 2.16. *Let a_1, a_2 the unit principal directions and κ_1, κ_2 the principal curvatures at $q \in S$, then the extended shape operator can be associated with the symmetric matrix*

$$\overline{\mathrm{W}}_q = \kappa_1 \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + \kappa_2 \boldsymbol{a}_2 \otimes \boldsymbol{a}_2$$

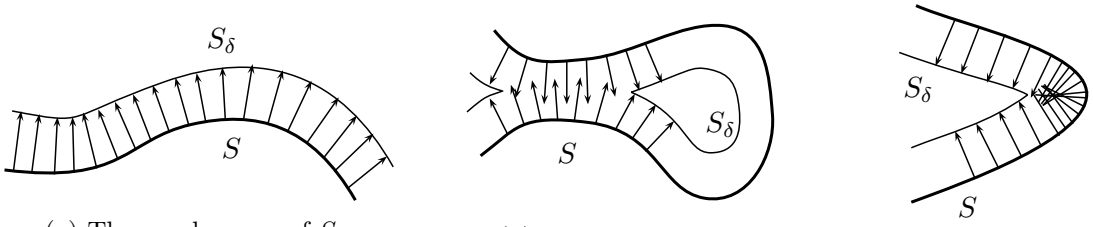
The operation $\otimes$ is the outer vector product $\boldsymbol{v} \otimes \boldsymbol{w} = \boldsymbol{v}\boldsymbol{w}^T \in R^{3\times 3}$.

Proof. We verify that the definition is an extension of the previous shape operator by applying it to $\boldsymbol{a}_i,\ i = 1, 2$ and $\boldsymbol{n}(q)$.

$$\overline{\mathrm{W}}_q(\boldsymbol{a}_i) = \kappa_1 \boldsymbol{a}_1 \underbrace{\boldsymbol{a}_1^T \boldsymbol{a}_i}_{\delta_{1,i}} + \kappa_2 \boldsymbol{a}_2 \underbrace{\boldsymbol{a}_2^T \boldsymbol{a}_i}_{\delta_{2,i}} = \kappa_i \boldsymbol{a}_i$$

$$\overline{\mathrm{W}}_q(\boldsymbol{n}) = \kappa_1 \boldsymbol{a}_1 \underbrace{\boldsymbol{a}_1^T \boldsymbol{n}}_{0} + \kappa_2 \boldsymbol{a}_2 \underbrace{\boldsymbol{a}_2^T \boldsymbol{n}}_{0} = \mathbf{0}$$

□



(a) The regular case of S_δ. (b) Intersections of S_δ. (c) Singularities of S_δ.

Figure 2.3: The δ-shift S_δ and special cases, which can be avoided for δ sufficiently small.

The identity $H = \text{trace}(\overline{\mathrm{W}})$ still holds true.

For the discretization in section 2.2 we also need the shape operator with swapped eigenvalues to the original eigenvectors. We denote this by $\widetilde{\mathrm{W}}$ and the extension, defined in the same way as above, with $\widetilde{\overline{\mathrm{W}}}$. $\widetilde{\mathrm{W}}\boldsymbol{a}_1 = \kappa_2 \boldsymbol{a}_1$ and vice versa.

2.1.5 Steiner's theorem

Additionally to the mean and Gaussian curvature and the (extended) shape operators we define equivalent measures on the surface.

Definition 2.17. For a region $Q \subset S$ we define the area integrals *mean curvature measure*

$$H(Q) := \int_{p \in Q} H(p) dA(p)$$

Gaussian curvature measure

$$K(Q) := \int_{p \in Q} K(p) dA(p)$$

and *extended shape operator measure*

$$\overline{\mathrm{W}}(Q) := \int_{p \in Q} \overline{\mathrm{W}}(p) dA(p) \qquad \widetilde{\overline{\mathrm{W}}}(Q) := \int_{p \in Q} \widetilde{\overline{\mathrm{W}}}(p) dA(p)$$

In the following we will consider the *δ-shift* S_δ of a surface S, which is the set of points at a (signed) normal distance of maximum δ from a surface

$$S_\delta := \{q + \gamma \boldsymbol{n}(q) | q \in S, \gamma \in [0, \delta]\}$$

The offset δ can be chosen negative, then the offset is "inside" the surface with respect to the orientation. It is also possible to define $\delta(q)$ depending on $q \in S$. For a smooth surface, with δ sufficiently small we can avoid *intersections*, when different parts of the surface come to close (see figure 2.3b) and *singularities*, when the curvature is to large (see figure 2.3c). The latter happens, if $[0, \delta]$ contains the inverse of a principal curvature (a *principal curvature radius*). For non-positive principal curvatures (e.g. the boundary of a convex set with a Gauß map pointing outwards) $\delta > 0$ can be arbitrary without having intersections or singularities.

Theorem 2.18 (Steiner's formula)**.** *The volume of the δ-shift Q_δ can be expressed as*

$$V(Q_\delta) = \delta A(Q) - \delta^2 H(Q) + \frac{\delta^3}{3} K(Q) \tag{2.2}$$

for a sufficiently small region $Q \subset S$ (to prevent intersections) and a sufficiently small $\delta > 0$, that is $[0, \delta]$ not containing inverse principal curvatures (to prevent singularities).

If the conditions on δ and Q do not hold true, then parts of the shift will be measured more than one time or negatively.

Proof. Q_δ is the image over $U \times [0, \delta]$ of the function

$$(u, v, \gamma) \overset{s}{\mapsto} \boldsymbol{x}(u, v) + \gamma \boldsymbol{n}(\boldsymbol{x}(u, v))$$

With substitution we calculate

$$\begin{aligned} V(Q_\delta) &= \int_{Q_\delta} dq = \int_{u,v} \int_0^\delta \left| \det \left(\frac{\partial s}{\partial u}, \frac{\partial s}{\partial v}, \frac{\partial s}{\partial \gamma} \right) \right| d\gamma \, d(u, v) = \\ &= \int_{u,v} \int_0^\delta \left| \det \left(\boldsymbol{x}_u + \gamma d\boldsymbol{n}(\boldsymbol{x}_u), \boldsymbol{x}_v + \gamma d\boldsymbol{n}(\boldsymbol{x}_v), \boldsymbol{n}(\boldsymbol{x}(u, v)) \right) \right| d\gamma \, d(u, v) \end{aligned}$$

We do another substitution to get a basis of (local) principal directions $\boldsymbol{a}_1, \boldsymbol{a}_2$. This transformation has $\det(...) = 1$ since the two vectors $\boldsymbol{x}_u, \boldsymbol{x}_v$ must only be rotated about the same angle and $\boldsymbol{n}$ can be kept fixed. The new basis simplifies to

$$\boldsymbol{a}_i + \gamma d\boldsymbol{n}(\boldsymbol{a}_i) = \boldsymbol{a}_i + \gamma(-1)\kappa_i \boldsymbol{a}_i = (1 - \gamma\kappa_i)\boldsymbol{a}_i, \quad i = 1, 2$$

and the determinant of the matrix in the integral above is

$$\det\left(\begin{pmatrix} (1 - \gamma\kappa_1) & 0 & 0 \\ 0 & (1 - \gamma\kappa_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} (\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{n}) \right) = (1 - \gamma\kappa_1)(1 - \gamma\kappa_2) 1 \cdot \det(\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{n})$$

Under the assumptions we can skip the norm for the inner integral and get

$$\int_0^\delta (1 - \gamma\kappa_1)(1 - \gamma\kappa_2) d\gamma = \delta - \frac{\delta^2}{2}(\kappa_1 + \kappa_2) + \frac{\delta^3}{3}\kappa_1\kappa_2 = \delta - \delta^2 H + \frac{\delta^3}{3} K$$

For the rest of the integral, one observes, that $\det(\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{n}) = ||\boldsymbol{a}_1 \times \boldsymbol{a}_2||$ and gets the area integral from equation (2.1). This proofs equation (2.2). □

2.1.6 Minimal surfaces

In this part of the section we give a definition of a minimal surface and a characterization of minimal surface area involving the mean curvature. We will follow the notation of [2] and refer to [9] for a more comprehensive study of surfaces with this property.

Definition 2.19. A *minimal surface* is a regular surface with constant zero mean curvature for all points on that surface, $H \equiv 0$.

We want to explain the name "minimal". For a parametrization $\boldsymbol{x}$ of the surface S, we define a *normal variation* with a differentiable function $h : U \to \mathbb{R}$ to the parameter $t \in (-\delta, \delta)$ by

$$\boldsymbol{x}^t(u, v) := \boldsymbol{x}(u, v) + th(u, v)\boldsymbol{n}(u, v)$$

Lemma 2.20. *Let $\boldsymbol{x}^t$ be a normal variation of a regular surface $\boldsymbol{x}$ over a compact region U. For δ small enough, $\boldsymbol{x}^t$ is also regular and the area of this surface is*

$$A(t) := A(\boldsymbol{x}^t(U)) = \int_U \sqrt{1 - 4thH + t^2 R(u,v,t)}\sqrt{EG - F^2}d(u,v) \tag{2.3}$$

with a function $R(u,v,t)$ polynomial in t and the coefficients E, F, G of the first fundamental matrix I.

Proof. The surface of the normal variation has

$$\begin{aligned}\boldsymbol{x}^t_u &= \boldsymbol{x}_u + th_u\boldsymbol{n} + th\boldsymbol{n}_u\\ \boldsymbol{x}^t_v &= \boldsymbol{x}_v + th_v\boldsymbol{n} + th\boldsymbol{n}_v\end{aligned}$$

Thus, we get the coefficients of the first fundamental form of $\boldsymbol{x}^t$

$$\begin{aligned}E^t &= E + 2th\langle \boldsymbol{x}_u, \boldsymbol{n}_u\rangle + t^2h^2\langle \boldsymbol{n}_u, \boldsymbol{n}_u\rangle + t^2h_u^2\\ F^t &= F + th(\langle \boldsymbol{x}_u, \boldsymbol{n}_v\rangle + \langle \boldsymbol{x}_v, \boldsymbol{n}_u\rangle) + t^2h^2\langle \boldsymbol{n}_u, \boldsymbol{n}_v\rangle + t^2h_uh_v\\ G^t &= G + 2th\langle \boldsymbol{x}_v, \boldsymbol{n}_v\rangle + t^2h^2\langle \boldsymbol{n}_v, \boldsymbol{n}_v\rangle + t^2h_v^2\end{aligned}$$

Therefore, we calculate with coefficients of the shape operator e, f, g

$$\det(\mathrm{I}^t) = E^tG^t - (F^t)^2 = EG - F^2 + 2th(-Eg + 2Ff - Ge) + t^2\bar{R}$$

where $\bar{R}$ is a polynomial in t. With the mean curvature of $\boldsymbol{x}$

$$H = \frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2}$$

we get

$$E^tG^t - (F^t)^2 = (EG - F^2)(1 - 4thH) + t^2\bar{R} = (EG - F^2)(1 - 4thH + t^2R)$$

with $R = \frac{\bar{R}}{EG-F^2}$. The regularity of $\boldsymbol{x}^t$ follows from the compactness of the region: This provides boundaries for h, H and R and the determinant of I^t converges to the determinant of I. Equation (2.3) follows from the definition of the area of a surface, equation (2.1). □

Proposition 2.21. *Let $\boldsymbol{x} : U \to S$ be a regular surface and let $A(t)$ be defined by* (2.3)*. Then $\boldsymbol{x}(U)$ is a minimal surface ($H \equiv 0$) if and only if $\frac{d}{dt}A(0) = 0$ over all compact sets U and for all h defining a normal variation.*

Proof. We calculate for any compact U

$$A'(t) = \frac{d}{dt}A(t) = \int_U \frac{-4hH + 2tR + t^2R_t}{2\sqrt{1 - 4thH + t^2R}}\sqrt{EG - F^2}d(u,v)$$

and get

$$A'(0) = -2\int_U hH\sqrt{EG - F^2}d(u,v)$$

It follows, that for a minimal surface $A'(0) = 0$ for all h and compact U.

Figure 2.4: Soap bubbles are naturally surfaces with constant mean curvature. Source: pixabay.com

On the other hand we suppose that $A'(0) = 0$ for all h and compact U and choose $h = H$.

$$A'(0) = -2 \int_U H^2 \sqrt{EG - F^2} d(u, v)$$

If $H(u_0, v_0) \neq 0$ we can find for continuity reasons a set U with $H(u, v)^2 > 0$ for all $(u, v) \in U$. Since $\sqrt{EG - F^2} > 0$ we get $A'(0) < 0$ on U, which is a contradiction, and therefore $H(u, v) = 0$ for all (u, v) in U. □

The first time, that minimal surfaces have been studied, was in 1762 by Lagrange, who considered a variational problem of finding a minimal surface to a given boundary. This problem is named the *Plateau's problem*. Meusnier discovered the helicoid and catenoid to satisfy Lagrange's equation and concluded, that surfaces with zero mean curvature minimize the area. Many mathematicians contributed to this field of research during the next centuries and Douglas and Radó finally completely solved the problem, given a fixed boundary. During the 20^{th}, many more examples of minimal surfaces have been found.

Equivalent definitions of minimal surfaces include for example the variational definition, a soap film definition (any small region of a minimal surface is equal to a idealized soap film between its boundary), a differential equation definition (Euler-Lagrange equation) and many more [20].

Constant mean curvature surfaces

Constant mean curvature surfaces are a generalization of minimal surfaces. Their mean curvature is $H \equiv c$ with some constant scalar. They have been investigated for a long time and the research is still not complete. Beginning with the conjecture, that the only closed compact CMC surface in $\mathbb{R}^3$ is the sphere by H. Hopf, the research was continued by H. Wente [41] who showed the existence of CMC surfaces, which are topologically equivalent to a torus.

In application, CMC surfaces are used as models for minimal surfaces with different air pressure on both sides. For example a soap bubble is a CMC surface (figure 2.4).

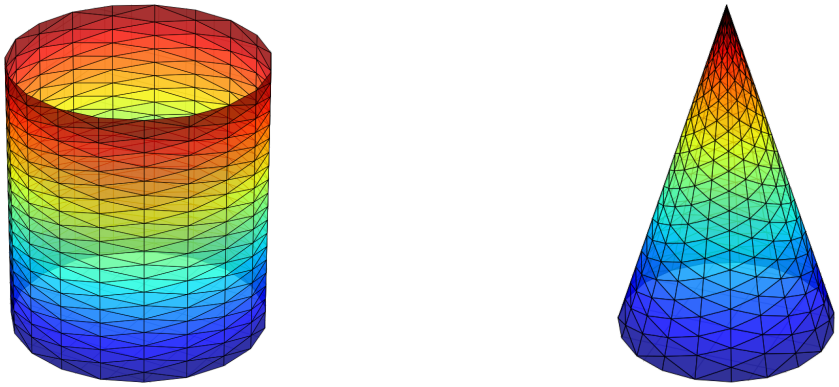


Figure 2.5: A cylinder and a cone are special representatives of developable surfaces.

2.1.7 Developable surfaces

We want to study another special class of surfaces.

Definition 2.22. A *developable surface* is a regular surface S with

$$K \equiv 0$$

for all points on that surface.

In fact, that means, that one or both principal curvatures are 0. An interesting property of developable surfaces can be derived with the theorema egregium (2.15) of Gauß and the transposition by Minding.

Corollary 2.23. *Developable surfaces in* $\mathbb{R}^3$ *are isometric to a plane.*

A *ruled surface* is a surface S with a straight line trough every point $q \in S$, that lies on S. Developable surfaces are necessarily ruled surfaces. Conversely, not all ruled surfaces are developable (for example the hyperboloid is ruled, but not developable).

The image of the Gauß map $\boldsymbol{n}(S)$ of a developable surface is either a single point or a curve.

In $\mathbb{R}^3$ there are four different types of developable surfaces (see figure 2.5).

1. A plane (sometimes seen as a special case of the other types)
2. A conical surface (including a cone)
3. A generalized cylinder (a cylinder with not only a circle, but any curve as cross section)
4. A tangent developable (the union of tangents of a space curve)

Constant Gaussian curvature surfaces

We want to generalize developable surfaces to some, with *constant Gaussian curvature*. In 1827 Gauß stated his famous *theorema egregium*, saying that the Gaussian curvature is constant under local isometry. Some years later, Minding stated that surfaces with the same constant Gaussian curvature, are locally isometric. Liebmann proofed, that the only regular closed surface with

constant curvature $K > 0$ in $\mathbb{R}^3$ is a sphere. There exists no regular closed surface in $\mathbb{R}^3$ with constant negative curvature. In section 5.4 we treat the example of the so-called pseudosphere, a surface of revolution with some non regular singularities.

More recent studies of surfaces with constant Gaussian curvature include the question of finding such surfaces to a given boundary. See for instance [14] in the positive, and [33] for the negative case.

2.2 Discrete differential geometry

This younger but not less comprehensive part of differential geometry arose with the need of using the computer to work with data from real or virtual objects, often given by a point cloud of sample points. Results, that required smooth objects, are not applicable any more, since those finite data points form only somehow piecewise linear meshes with edges and corners. Invented methods vary in their applications and have different downsides. We will focus on ideas, that fit our problems best and only refer to other approaches.

2.2.1 Basics of discrete differential geometry

We want to start with the definition of a mesh.

Definition 2.24. A *polyhedral mesh* (V, E, F) is a finite set of points, the *vertices*

$$V = (v_1, ..., v_n), \quad v_i \in \mathbb{R}^3, \quad i = 1..n$$

straight line segments between those points, called *edges*

$$E = (e_{i_1 j_1}, ..., e_{i_k, j_k}), \quad e_{i,j} = (v_i, v_j)$$

and planar polygons, the *facets*

$$F = (f_{i_1, j_1, ...}, ..., f_{i_m, j_m, ...}), \quad f_{i,j,...} = (v_i, v_j, ...)$$

A mesh with three vertices forming each facet $f_{i,j,k} = (v_i, v_j, v_k)$ is called a *triangle mesh* and the faces are triangles.

A mesh with four vertices forming each face is called a *quadrilateral* or *quad mesh*.

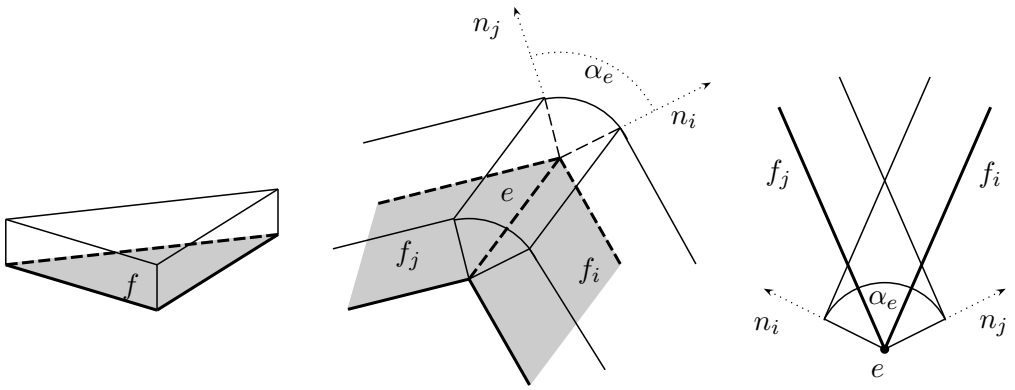
Note: In this thesis we will consider only triangular meshes, except if mentioned explicitly.

Definition 2.25. Analog to the Gauß map in the smooth case, we define the *facet normals* on a mesh as the oriented unit normals on each facet.

$$n(f_{i,j,k,...}) = \frac{e_{i,j} \times e_{i,k}}{||e_{i,j} \times e_{i,k}||}$$

We want to define the *vertex normals* as the normalized sum of adjacent facet.

$$n(v) = \frac{\sum_{f; v \in f} n(f)}{||\sum_{f; v \in f} n(f)||}$$



(a) Volume over facet f. (b) Volume over convex edge e. (c) Case of concave edge e.

Figure 2.6: The δ-shift for polyhedral meshes.

2.2.2 Discrete curvature via normal cycles

To find useful definitions for discrete mean and Gaussian curvatures, as well as for the shape operator, we can intuitively start with Steiner's formula (2.2). The idea of computing the volume of a δ-shift can be transformed onto a polyhedral mesh. In the discrete case, we do not have a continuous Gauß map, so first we have to give a (for the smooth case equivalent) definition for S_δ

$$S_\delta := \{p \in \mathbb{R}^3 | d(p, S) \leq \delta, p \text{ outside of } S\}$$

with the distance function $d(p, S) := \min_{q \in S} ||p - q||$ and again a sufficiently small δ.

We will define the signed volumes, which should sum up to the volume of the S_δ. separately for facets, edges and vertices of our mesh (V, E, F).

1. The shift set over a facet f in direction of the normal $n(f)$ is a right prism and has the volume
$$V(\delta, f) := \delta A(f)$$

2. The shift set over an edge can be one of two cases:

   a) If the oriented surface (or the body having this surface as boundary, respectively) at the edge e is convex, the corresponding volume is a sector of a cylinder and we have
   $$V(\delta, e) := \delta^2 \frac{\alpha_e}{2} \operatorname{len}(e) \tag{2.4}$$
   The (positive) angle α_e is defined between the two normals of the adjacent facets f_i, f_j: $\cos(\alpha_e) = n(f_i) \cdot n(f_j)$ and $\operatorname{len}(e_{i,j}) := ||v_i - v_j||$

   b) In the case of a concave edge, we counted volume elements twice by both adjacent facet volume definitions (figure 2.6c) and therefore we want to have a negative volume. We use the same definition[(2)] $V(\delta, e)$, but with a negative angle $\alpha_e < 0$.

[(2)] We note, this does not represent the volume in a correct way. Instead to subtract the volume counted twice, for concave edges the definition $V(\delta, e) := -\delta^2 |\tan(\frac{\alpha_e}{2})| \operatorname{len}(e)$ would be better. However, this would lead to computational problems for implementations. Anyways, the situation at vertices is even more complicated, so we refer to the notes on page 17.

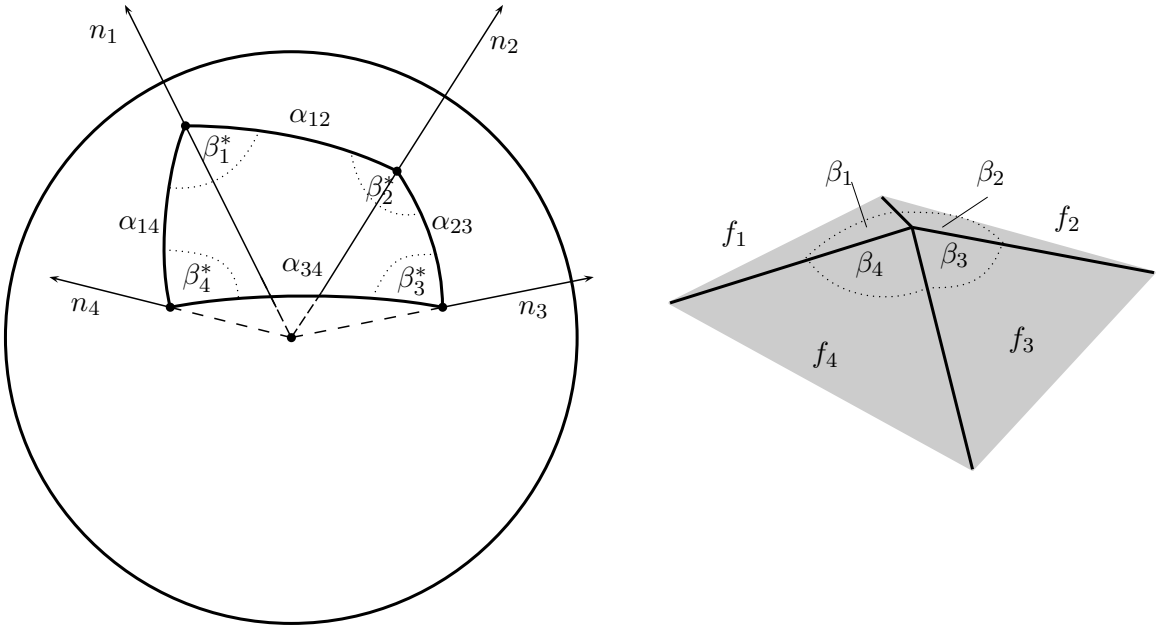


Figure 2.7: Spherical polygon between 4 adjacent face normals.

3. For a vertex we get a corresponding spherical polygon in the following way. The normals of adjacent facets are mapped to the unit sphere and adjacent facet normals are connected via arcs on that sphere. These arcs have the length α_e and the corners of the polygon on the sphere have internal angles $\beta^*_{v,f}$. They are the angles between the planes $e_i^\perp$ and $e_j^\perp$ of the edges contained in the facet f and sharing the vertex v, and therefore $\pi - \beta_{v,f}$, with $\beta_{v,f}$ being the angle between the two edges e_i, e_j in the facet f. To get the area of this spherical polygon, we use the theorem of incident angles for spherical geometry[3] (see [18]). n is the number of adjacent faces/edges at v.

$$A = \sum_{f;v\in f} \beta^*_{v,f} - (n-2)\pi = \sum_{f;v\in f} (\pi - \beta_{v,f}) - (n-2)\pi = 2\pi - \sum_{f;v\in f} \beta_{v,f}$$

The volume of this spherical sector on a sphere with radius δ is then

$$V(\delta, v) := \frac{A}{4\pi}\frac{4}{3}\delta^3\pi = \frac{\delta^3}{3}(2\pi - \sum_{f;v\in f} \beta_{v,f}) \tag{2.5}$$

Again, we note that this guarantees to represent the correct total volume only in the convex case.

Theorem 2.26. *With the definitions above, for an orientable convex closed polyhedral mesh* $(V < E < F)$ *the volume of* S_δ *is for all* $\delta > 0$

$$\mathrm{vol}(S_\delta) = \sum_{f\in F} V(\delta, f) + \sum_{e\in E} V(\delta, e) + \sum_{v\in V} V(\delta, v)$$

With this motivation we can now define a discrete mean and Gaussian curvature.

[3] This statement is an application of the famous *Gauß-Bonnet theorem*, found in standard literature about differential geometry, e.g. [6].

Definition 2.27 (Discrete mean and Gaussian curvature, version 1)**.** For an orientable polyhedral mesh (V, E, F) with signed angles α_e between adjacent facet normals and angles $\beta_{v,f}$ between edges of facet f at vertex v we define

$$H(e) := -\frac{\alpha_e}{2}\operatorname{len}(e) \qquad\qquad K(v) := 2\pi - \sum_{f, v\in f} \beta_{v,f}$$

Corollary 2.28. *With the definitions above theorem 2.26 substitutes to*

$$\operatorname{vol}(S_\delta) = \delta \sum_{f\in F} A(f) - \delta^2 \sum_{e\in E} H(e) + \frac{\delta^3}{3} \sum_{v\in V} K(v)$$

This is a discrete version of Steiner's formula (2.2) and has first been considered in [35]. It explains the motivation of definition 2.27. As in the smooth case we can define curvature measures.

Definition 2.29. For a region Q in the polyhedral mesh (V, E, F) we define the *discrete mean curvature measure* $H(Q)$ and the *discrete Gaussian curvature measure* $K(G)$

$$H(Q) := -\sum_{e\in E} \frac{\alpha_e}{2}\operatorname{len}(e \cap Q) \qquad\qquad K(Q) := \sum_{v\in V\cap Q} K(v)$$

Research on the quality of these operators has been done in [8]. The exact definition of discrete operators analogously to the smooth case, is achieved by the theory of normal cycles (introduced by [42, 43]). This theory gives also a foundation for the definition (2.4) and (2.5). For a sufficiently fine ε-sample [1], Delaunay [10] triangle mesh M and a smooth surface S, which may be non-convex, the error between the defined smooth and discrete measure operators is linear in ε.

Another definition encountered in [8] is the discrete shape operator.

Definition 2.30. For a region Q in the polyhedral mesh (V, E, F) we define the *discrete extended shape measures* $\overline{W}(Q)$ and $\overline{\widetilde{W}}(Q)$ in analogy to definition 2.17

$$\overline{W}(Q) = -\sum_{e\in E} \frac{\operatorname{len}(e \cap Q)}{2}[(\alpha_e - \sin\alpha_e)\boldsymbol{e}^+ \otimes \boldsymbol{e}^+ + (\alpha_e + \sin\alpha_e)\boldsymbol{e}^- \otimes \boldsymbol{e}^-]$$
$$\overline{\widetilde{W}}(Q) = -\sum_{e\in E} \operatorname{len}(e \cap Q)\alpha_e \boldsymbol{e} \otimes \boldsymbol{e}$$

with $\boldsymbol{e}$ denotes a unit 3-vector in direction of edge e and $\boldsymbol{e}^\pm = \frac{n(f_i)\pm n(f_j)}{||n(f_i)\pm n(f_j)||}$ the normalized sum and difference between unit normals of adjacent faces f_i, f_j.

We note, that the discrete extended shape operator with swapped eigenvalues has a much nicer representation. Besides the theory in [8] the definition can be motivated by proposition 2.16. The projection operators $\boldsymbol{a}_i \otimes \boldsymbol{a}_i$ weighted with κ_1 onto the 1-dimensional spaces $[\boldsymbol{a}_i]$ are transfered to the operators onto the edges e with weights $\alpha_e \operatorname{len}(e)$. The weights represent the bending over that edge. This explains also the nicer representation of $\overline{\widetilde{W}}$, because the curvature κ_1 is actually the bending over the vector $\boldsymbol{a}_2$ and vice versa. For the definition of the sum to make sense we also have to consider Euler's formula, theorem 2.13, and the assumption, that the directions of edges at a vertex are equally distributed.

Figure 2.8: Minimal curvature directions estimated on a mesh of Michelangelo's David, based on definition 2.30. Image from [8].

Remark. Again, the discrete shape operators generalize the mean curvature.

$$\operatorname{trace}(\overline{W}(Q)) = \operatorname{trace}(\overline{\overline{W}}(Q)) = H(Q)$$

The principal curvatures and directions (plus the eigenvalue 0 and the normal vector) can be found by solving the 3×3 matrix eigenvalue problems.

$$\begin{aligned} \overline{W}\boldsymbol{a}_1 &= \kappa_1 \boldsymbol{a}_1 & \qquad \overline{\overline{W}}\boldsymbol{a}_1 &= \kappa_2 \boldsymbol{a}_1 \\ \overline{W}\boldsymbol{a}_2 &= \kappa_2 \boldsymbol{a}_2 & \qquad \overline{\overline{W}}\boldsymbol{a}_2 &= \kappa_1 \boldsymbol{a}_2 \\ \overline{W}\boldsymbol{n} &= \boldsymbol{0} & \qquad \overline{\overline{W}}\boldsymbol{n} &= \boldsymbol{0} \end{aligned}$$

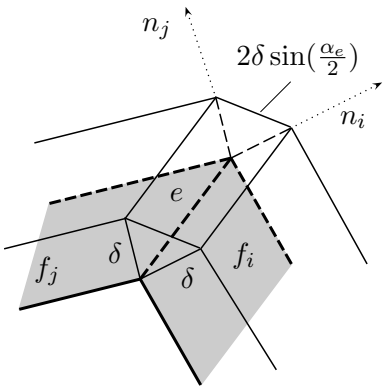


Figure 2.9: A second approach in defining the δ-shift. The offset at edges is connected via plane surfaces (compare figure 2.6b).

2.2.3 Discrete mean curvature - cotangent formula

A second approach to derive a discrete definition for the mean curvature uses the surface version of Steiner's formula. The intuitive approach is similar to the one before. We "define" now the surface areas of the δ-shift, $A(\delta, f) = A(f)$ and $A(\delta, e) = 2\delta \sin(\frac{\alpha_e}{2}) \operatorname{len}(e)$ as well as $A(\delta, v) = \delta^2(...)$. The difference to the previous definition of the volumes is the connection at edges via planar surfaces instead of cylindrical sections (figure 2.9) and at vertices via plane instead of spherical polygons.

Definition 2.31 (Discrete mean curvature, version 2)**.** For an orientable polyhedral mesh (V, E, F) with signed angles α_e between adjacent facet normals we define

$$H(e) := -\sin\frac{\alpha_e}{2} \operatorname{len}(e)$$

Furthermore we define the *mean curvature edge vector*

$$\overrightarrow{H}(e) := H(e)\frac{n_i + n_j}{||n_i + n_j||}$$

for indices i, j of adjacent faces $e \in f_i, f_j$.

Finally we define the *mean curvature vertex vector*

$$\overrightarrow{H}(v) := \sum_{e:v\in e} \overrightarrow{H}(e) \tag{2.6}$$

In case of a vertex surrounded by convex edges, the angles $\alpha_e \geq 0$ and therefore $-\sin(\alpha_e/2) \leq 0$. The vectors $\overrightarrow{H}(e)$ are pointing into the surface and the sum $\overrightarrow{H}(v)$ too, which is opposite to the normal vector $n(v)$. This corresponds with the negative mean curvature of convex surfaces.

Proposition 2.32. *Let (V, E, F) be a polyhedral mesh. We use the notation $v_i \sim v$ for adjacent vertices and positive oriented indexing around all neighbors, so $v_{i+1} \sim v_i$. The mean curvature vertex vector defined by* (2.6) *is*

$$\overrightarrow{H}(v) = \frac{1}{2}\sum_{v_i:v_i\sim v} J_i(v_i - v_{i+1})$$

where J_i denotes a $90°$-rotation in the plane of f_i spanned by (v, v_i, v_{i+1}).

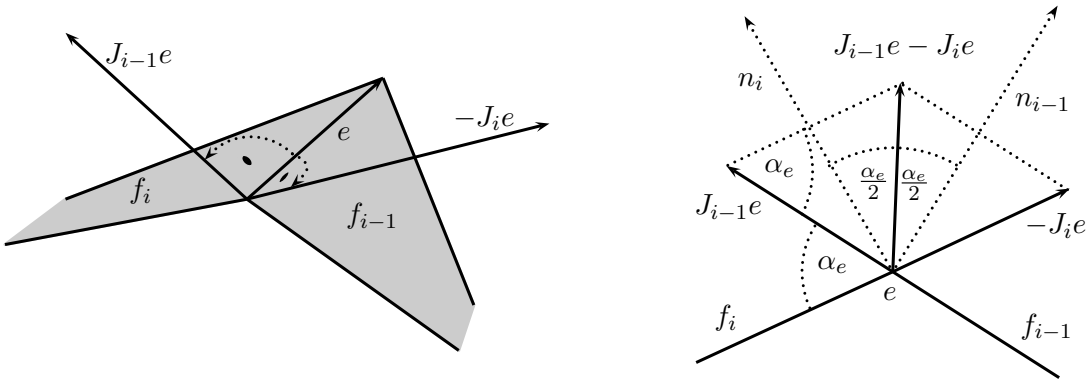


Figure 2.10: The edge e is rotated in the planes of adjacent facets.

The 90°-rotation of an edge $v_i - v_{i+1}$ by J_i lets it point away from the vertex v.

Proof. We consider rotation of an edge $e = v_i - v$ in two different planes. Notice, that $-J(e) = J(-e)$.

$$||J_{i-1}(v_i - v) - J_i(v_i - v)|| = 2\,\mathrm{len}(v_i - v)\left|\sin\left(\frac{\alpha_{e_{v,v_i}}}{2}\right)\right| = 2|H(e_{v,v_i})|$$

Actually, we have $J_{i-1}(v_i - v) - J_i(v_i - v) \in [n_{i-1} + n_i]$ and therefore

$$-\frac{1}{2}(J_{i-1}(v_i - v) - J_i(v_i - v)) = \overrightarrow{H}(e_{v,v_i})$$

The correct orientation follows from geometrical considerations (see figure 2.10).

Further, we have

$$\begin{aligned}\overrightarrow{H}(v) = \sum_{i:v\sim v_i} \overrightarrow{H}(e_{v,v_i}) &= \sum_{i:v\sim v_i} -\frac{1}{2}(J_{i-1}(v_i - v) - J_i(v_i - v)) \\ &= \sum_{i:v\sim v_i} \frac{1}{2}J_i(v_i - v) - \frac{1}{2}J_i(v_{i+1} - v) \\ &= \frac{1}{2}\sum_{i:v\sim v_i} J_i(v_i - v_{i+1})\end{aligned}$$

This proves the equation. □

Remark. The mean curvature vertex vector $\overrightarrow{H}(v)$ also has the representation

$$\overrightarrow{H}(v) = \frac{1}{2}\sum_{e:v\in e}(\cot\gamma_e + \cot\tilde{\gamma}_e)\vec{e}$$

With γ_e and $\tilde{\gamma}_e$ being the angles in the adjacent facets of e opposite to the edge (figure 2.11). This can be derived either straight from proposition 2.32 or via the definition of the *discrete Laplace-Beltrami operator*. See [40] for theory about the Laplace-Beltrami operator.

Proposition 2.33. *Let $T = (V, E, F)$ be a triangle mesh with moving vertices $v(t)$. For the area $A(t)$ of this mesh, we have*

$$\frac{d}{dt}A(T) = -\sum_{v\in V}\langle\overrightarrow{H}(v), \dot{v}\rangle$$

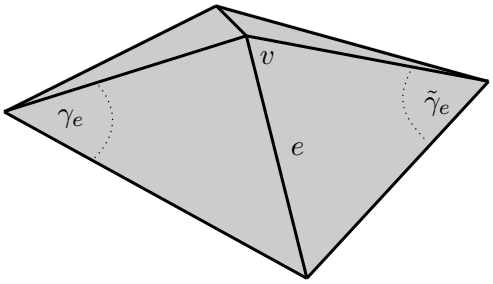


Figure 2.11: The definition of the angles in the alternative cotangent-formula of $\vec{H}(v)$.

so that means $\nabla A = -(\overrightarrow{H}(v))_{v\in V} \in (\mathbb{R}^n)^V$.

Proof. It is sufficient, to proof the statement in case of exact 1 moving vertex $v(t)$ and stationary points for the rest and show $\frac{d}{dt}A(T) = -\langle \overrightarrow{H}(v), \dot{v}\rangle$. The general case follows from superposition. For one facet $f_{v,v_i,v_{i+1}}$ we choose a local coordinate system identified with $\mathbb{R}^2$. The area of the triangle $f_{v,v_i,v_{i+1}}$ is $A = \frac{1}{2}\det(v_{i+1} - v_i, v - v_i) = \frac{1}{2}\langle J_i(v_{i+1} - v_i), v - v_i\rangle$. For a variation of $v(t)$ in this plane, we have the formula

$$\dot{A} = \frac{1}{2}\langle J_i(v_{i+1} - v_i), \dot{v}\rangle \tag{2.7}$$

For any variation orthogonal to the facet $f_{v,v_i,v_{i+1}}$, the area will increase, so we have $\dot{A} = 0$ in this case. The tangential vector $\dot{v}$ is orthogonal to $J_i(v_{i+1} - v_i)$ and (2.7) holds for all variations. Because of linearity of differentiation (2.7) is true in general for any coordination. Summation gives

$$\frac{d}{dt}A(T) = \sum_{i:v\sim v_i} \frac{1}{2}\langle J_i(v_{i+1} - v_i), \dot{v}\rangle = -\sum_{e:v\in e} \langle \overrightarrow{H}(e), \dot{v}\rangle = -\langle \overrightarrow{H}(v), \dot{v}\rangle$$

Superposition yields the desired result. □

Corollary 2.34. *A triangle mesh* (V, E, F) *has minimal surface, if the mean curvature vertex vector, defined in equation* (2.6)*, is zero for all vertices,* $\overrightarrow{H}(v) = 0, \forall v \in V$.

2.2.4 Discrete curvature - osculating paraboloid

A different approach of calculating curvatures on discrete surfaces is via the *osculating paraboloid*, as described in [17]. The surface is locally approximated by a bivariate second order polynomial and then the principal curvatures of this paraboloid are calculated.

For each vertex v, the vertex normal $n(v)$ corresponds to the z-axis of a local coordinate system (x, y, z) with v at its origin. Via projection, each neighbor vertex $v_j \sim v$ has a representation in this basis $v_j = (x_j, y_j, z_j)$. The osculating paraboloid is then the function

$$p(x, y) = ax^2 + bxy + cy^2$$

such that the least squares problem

$$f(a, b, c) = \sum_j (p(x_j, y_j) - z_j)^2$$

is minimized (see chapter 3 for theory about non-linear least squares problems[4]).

The curvatures of the osculating paraboloid are

$$H = a + c \qquad\qquad K = 4ac - b^2$$

$$\kappa_{1,2} = H \pm \sqrt{H^2 - K} = a + c \pm \sqrt{(a - c)^2 + b^2}$$

While this method is comparably easy to implement and compute, it lacks the definition of operators and functionals which have derivatives, necessary in any optimization process. Although iterative algorithms with a paraboloid fitting have been studied [34], we will use this method only to compare our calculated results with the estimation based on paraboloid fitting.

[4] Alternatively one could consider a *reweighted* least squares problem to achieve more accurate fitting. We will not consider this in this thesis.

3 Optimization techniques

In this chapter we provide a short introduction to some well known optimization methods. We focus on theory, used in this thesis and refer to [12] and [13] for a more detailed description.

3.1 Least squares problem

Our main focus lies on the so called *least squares problem*. Consider a finite family of differentiable functions

$$(f_i)_{i=1..m}, \qquad f_i : \mathbb{R}^n \to \mathbb{R}$$

The objective function f is defined as the sum of squares of f_i.

$$f(x) := \sum_{i=1}^{m} (f_i(\boldsymbol{x}))^2, \qquad \boldsymbol{x} \in \mathbb{R}^n$$

In the *unconstrained* case, the aim is to find a minimum $\boldsymbol{x}^* \in \mathbb{R}^n$ for this function.[1]

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x})$$

From analysis, we know that a necessary condition is, that $\boldsymbol{x}^*$ is a critical point of f, that is if the gradient of f is zero. We get n gradient equations.

$$\frac{\partial f}{\partial x_k} = \sum_{i=1}^{m} 2 f_i \frac{\partial f_i}{\partial x_k} = 0, \qquad k = 1, \ldots, n$$

First, we consider linear functions, resulting in a *linear least squares problem*.

$$f_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} - b_i, \qquad \boldsymbol{a}_i \in \mathbb{R}^n, b_i \in \mathbb{R}$$

With the matrix $A \in \mathbb{R}^{m \times n}$, containing the vectors $\boldsymbol{a}_i$ as rows, and the vector $b \in \mathbb{R}^m$, with entries b_i, we can state the problem as follows. $||\cdot||$ denotes the standard Euclidean norm.

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} ||A\boldsymbol{x} - b||^2 \tag{3.1}$$

Typically, we have more functions than unknowns, $m > n$. In this case, $A\boldsymbol{x} = b$ is called an *overdetermined system of linear equations* (or just *overdetermined linear system*). In general,

[1] A *constrained* problem involves a subset $X \subset \mathbb{R}^n$ and one searches for $\boldsymbol{x}^* \in X$ with $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x} \in X} f(\boldsymbol{x})$.

this problem has no solution, $||A\boldsymbol{x}-b||^2 > 0, \boldsymbol{x} \in \mathbb{R}^n$. To find a minimum, we derive the gradient equations.

$$\begin{aligned} f(\boldsymbol{x}) &= ||A\boldsymbol{x} - b||^2 \\ &= \sum_{i=1}^{m}(\sum_{j=1}^{n} a_{ij}x_j - b_i)^2 \\ 0 = \frac{\partial f}{\partial x_k}(\boldsymbol{x}) &= \sum_{i=1}^{m} 2(\sum_{j=1}^{n} a_{ij}x_j - b_i)a_{ik} \\ &= 2\sum_{i=1}^{m}(\sum_{j=1}^{n} a_{ij}a_{ik}x_j - b_i a_{ik}) \\ &= 2\sum_{j=1}^{n}(\sum_{i=1}^{m} a_{ij}a_{ik})x_j - 2\sum_{j=1}^{n} b_i a_{ik} \end{aligned}$$

We can see, that in the last line each inner sum of the left part is an entry of the matrix A^TA and the right part is the k-th component of A^Tb. The system of gradient equations is therefore

$$A^TA\boldsymbol{x} = A^Tb \tag{3.2}$$

These are called the *normal equations*. The matrix A^TA is symmetric with size $n \times n$ and positive semi-definite. Equation (3.2) can be used with the concepts of section 3.2 to find a solution for (3.1).

In our application we will find quadratic equations $f_i(\boldsymbol{x}) = \boldsymbol{x}^TA_i\boldsymbol{x} + b_i\boldsymbol{x} + c_i$ (see section 4.5). To solve such a *non-linear least squares problem*, we approximate the model and iteratively generate solution vectors $\boldsymbol{x}^l, l \in \mathbb{N}$ until some criterion is fulfilled. Solving the problem for the update vector $\Delta\boldsymbol{x}$ gives the next guess $\boldsymbol{x}^l = \boldsymbol{x}^{l-1} + \Delta\boldsymbol{x}$. The approximation is a linearized problem, derived via Taylor's expansion, using the previous solution as start.

$$f_i(\boldsymbol{x}^l) \approx f_i(\boldsymbol{x}^{l-1}) + \nabla f_i(\boldsymbol{x}^{l-1})^T\Delta\boldsymbol{x}$$

Therefore this algorithm needs some initial starting vector $\boldsymbol{x}^0$. In notation from above, in each step one has to solve the linear least square problem with $a_{ij} = \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}^{l-1})$ and $b_i = -f_i(\boldsymbol{x}^{l-1})$. This procedure is known as the *Gauß-Newton method*. It remains to solve the occurring linear system.

3.2 Solving systems of linear equations

Many approaches have been invented to solve linear systems computationally. Depending on the size and structure of the problem, different methods appeared to be useful. We start with **direct solvers**. These methods lead to (up to numerical errors) exact solutions and can be applied in general to small systems (compare [21]).

Definition 3.1. A matrix $M \in \mathbb{R}^{n\times n}$ is *symmetric*, if $M^T = M$.
A symmetric matrix $M \in \mathbb{R}^{n\times n}$ is said to be *positive definite* if

$$x^TMx > 0, \qquad \forall x \in \mathbb{R}^n\backslash\{0\}$$

The matrix is *positive semi-definite* if

$$x^T M x \geq 0, \qquad \forall x \in \mathbb{R}^n$$

Proposition 3.2 (Cholesky decomposition)**.** *Given a symmetric, positive definite matrix $M \in \mathbb{R}^{n\times n}$. There exists a unique lower triangular matrix $L \in \mathbb{R}^{n\times n}$ with real, positive diagonal entries, such that*

$$M = LL^T$$

If M is positive semi-definite, the matrix L may have zero diagonal entries and might be not unique.

Given a Cholesky decomposition of M, the linear system $Mx = r$ can be solved via forward and back substitution:

$$L\boldsymbol{y} = r, \quad L^T\boldsymbol{x} = y$$

A decomposition can be found with special algorithms, based on Gaussian elimination.

Definition 3.3. A symmetric matrix $Q \in \mathbb{R}^{m\times m}$ is called orthogonal, if

$$Q^T Q = I$$

In fact, the transpose of Q is its inverse: $Q^T = Q^{-1}$.

Proposition 3.4 (QR decomposition)**.** *Given a matrix $M \in \mathbb{R}^{m\times n}$ with $m \geq n$. There exists an $m \times m$ orthogonal matrix Q and an $m \times n$ upper triangular matrix R (that is a matrix with an $n \times n$ upper triangular block and zeros in $(m - n)$ rows), such that*

$$M = QR$$

For the linear system of normal equations, the solution can also be found without forming the matrix $A^T A$ explicitly. For a given QR decomposition of the matrix $A = QR$, the normal equation is

$$R^T R\boldsymbol{x} = R^T Q^T QR\boldsymbol{x} = (QR)^T QR\boldsymbol{x} = A^T A\boldsymbol{x} = A^T b = R^T Q^T b$$

A solution to this is equivalent to a solution of $R_n\boldsymbol{x} = Q_n^T b$ with the $n \times n$ upper part R_n of R and the $n \times m$ upper part Q_n^T of Q^T.

Methods for finding a QR decomposition include a Gram-Schmidt process, Householder transformation and Givens rotation (see [21]).

Large linear systems with sparse matrices require a different approach: **iterative methods**. Starting with an initial guess, the solution vector is adapted each step until some final convergence criterion is fulfilled. The advantage of this methods is the possible handling of huge matrices at the cost of an exact solution, that might not be found after a finite number of iteration steps (depending on the algorithm). Convergence analysis of a used algorithm is essential to predict the outcome. A broad overview of different methods is found in [16].

The *conjugate gradient method* is the most popular iterative solver for linear systems. Given a symmetric, positive definite matrix $M \in \mathbb{R}^{n\times n}$, we define the M-inner product $\langle\cdot,\cdot\rangle_M$ by

$$\langle u, v\rangle_M := \langle Mu, v\rangle = u^T Mv$$

A pair of vectors u, v is said to be *conjugate* if $\langle u, v\rangle_M = 0$. With the conjugate gradient algorithm, one computes iteratively a basis $(p_1, p_2, ..., p_n)$ of conjugate vectors and weights α_k, to build a solution vector $\boldsymbol{x}^* = \sum_{k=1}^{n} \alpha_k p_k$. In each iteration step the vector $x_k = x_{k-1} + \alpha_k p_k$ is updated and due to good convergence, generally the algorithm can stop before it reaches n steps.

Better convergence can be achieved via so called *preconditioning*. Instead of $M\boldsymbol{x} - b = 0$, the problem $C^{-1}(M\boldsymbol{x} - b) = 0$ is solved, resulting in a better *condition number* (see [16]).

To adapt to more general matrix properties, methods like the *Biconjugate gradient method* [11] or the *Least Squares QR algorithm* [27] have been invented.

4 Implementation

We state the following definition of our problem.

General surface optimization problem with special side conditions

Let $M = (V, E, F)$ be a polyhedral mesh. Let a set of variables $\boldsymbol{x} = (x_1, \ldots, x_k)$ containing the vertex positions $V \subset \boldsymbol{x}$ fulfill some equations $(c_1, \ldots, c_l)$ that are based on the mesh connectivity

$$c_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, l$$

We call these equations *hard constraints*.
Another family of equations $(f_1, \ldots, f_m)$ is called *soft constraints*.

$$f_i(\boldsymbol{x}) \quad i = 1, \ldots, m$$

Let $(E_1, ..., E_n)$ be a family of *energy functions*.

$$E_i(\boldsymbol{x}) \quad i = 1, \ldots, n$$

Our optimization problem is the following. Given some weights $(v_1, \ldots, v_m)$ and $(w_1, \ldots, w_n)$. Find a mesh $M^* = (V^*, E, F)$ with new vertex positions and a set of variables $\boldsymbol{x}^* \supset V^*$ with

$$\left\| \sum_{i=1}^{m} v_i f_i(\boldsymbol{x}^*) \right\|^2 + \left\| \sum_{i=1}^{n} w_i E_i(\boldsymbol{x}^*) \right\|^2 = \min_{\boldsymbol{x}} \left\| \sum_{i=1}^{m} v_i f_i(\boldsymbol{x}) \right\|^2 + \left\| \sum_{i=1}^{n} w_i E_i(\boldsymbol{x}) \right\|^2 \quad (4.1)$$

$$\text{such that } c_i(\boldsymbol{x}^*) = 0 \qquad i = 1, \ldots, l \quad (4.2)$$

In our cases, the set of formally independent variables $\boldsymbol{x}(V, E, F)$ contains some redundant values, that could be calculated directly from others. This includes for example the normals n_i and principal curvatures κ_j. However for now we do not restrict our search space directly, but with use of the hard constraint equations. More on this in section 4.2. We will give a complete list of values in $\boldsymbol{x}$ in section 4.1 and also explain the reason for this special treatment. The soft constraints prescribe some *fairness energy*. These properties assure a "good looking" and useful result (section 4.3). The energies E_i are our desired conditions. They vary from one application to another. A general description is given in section 4.4 and special applications in chapter 5. Finally the weights are discussed in section 4.5 and the used software is presented in 4.6.

4.1 Setup and initialization

The achievement of a solution to the stated problem is fairly complex. Some reasons are

- The *number of unknowns* is generally large. In application meshes contain thousands to millions of vertices, edges and facets.

- The problem is highly *non linear* in the unknowns. For example an energy can be defined on the principal curvatures, which are eigenvalues to a matrix, which is itself not linearly dependent on V,E and F.
- Discretization of a smooth surface leads to approximation error, and so do the discrete versions of our curvature measures. Therefore an optimum is sometimes impossible without refinement of the mesh.
- The solution may *not be unique* and the system can converge to a not desired result. This leads to the last reason:
- A result may be optimal in numerical reasons, but not in application. Some pathological local minimums or even global minimums are *far away from what a user would expect.* Examples will be presented in the following sections.

To address all these problems, we introduce several techniques. Our implementation will involve a linear system, which is computed with numerical ideas from section 3.2. To derive this linear system we use the *guided projection* method. Basically we reduce our system to quadratic equations by introducing new variables. Thirdly we play with different fairness energies, to assure a good solution. Together, the study and examination of these ideas were the main part of research in this thesis.

Linearization of the involved equations is difficult, because of their large polynomial order. We introduce **additional variables**, to decrease the order of the involved equations.

Name	number of unknowns	description
v_i	$3\|V\|$	vertex coordinates in $\mathbb{R}^3$
n_i	$3\|F\|$	face normal coordinates
$(n \times n)_i$	$2 \cdot 3\|E\|$	cross product along halfedge[(1)] between face normals
W_i	$6\|V\|$	symmetric 3×3 extended shape operator[(2)] at v_i
a_i^1, a_i^2	$2 \cdot 3\|V\|$	eigenvectors of W_i at v_i
κ_i^1, κ_i^2	$2\|V\|$	eigenvalues of W_i at v_i
$\|\kappa\|_i^1, \|\kappa\|_i^2$	$2\|V\|$	absolute values of eigenvalues of W_i at v_i
$(n \times e)_i$	$2 \cdot 3\|E\|$	cross product along halfedge between normal and edge

Table 4.1: Complete list of available variables in the general guided projection setup.

Note, that for example $(n \times n)_i$ is the name of an independent number, although it indicates the meaning of this variable, e.g. the cross product of normals, or $|\kappa|_i^1$ is the absolute value of a principal curvature. We prefer this notation over using completely new letters.

In total we have $19|V| + 12|E| + 3|F|$ unknown variables. For triangle meshes, this is roughly $61|V|$ unknowns.

These variables are linked via quadratic or linear equations. In the implementation, they are also **initialized** via these equations. We use indexing and notation as in figure 4.1.

(1) See section 4.6 about the openmesh data structure.

(2) For simplicity we use the version of swapped eigenvalues $\widetilde{\widetilde{W}}$.

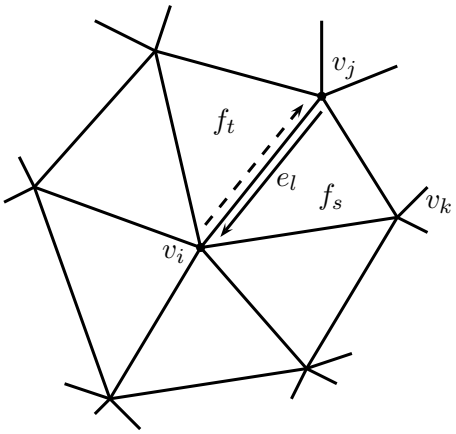


Figure 4.1: Section of the mesh and the indices for initialization equations (4.3) - (4.9).

$$n_s = \frac{(v_k - v_i) \times (v_j - v_i)}{||(v_k - v_i) \times (v_j - v_i)||} \tag{4.3}$$

$$(n \times n)_l = n_s \times n_t \tag{4.4}$$

$$W_i = -\frac{1}{A_i} \sum_{l,j:e_l=(v_j,v_i)} (n \times n)_l \otimes (v_i - v_j) \tag{4.5}$$

$$a_i^{1,2} = \text{eigenvector}(W_i), \quad ||a_i^{1,2}|| = 1 \tag{4.6}$$

$$\kappa_i^{1,2} = \text{eigenvalue}(W_i), \quad W_i a_i^{1,2} = \kappa_i^{1,2} a_i^{1,2} \tag{4.7}$$

$$|\kappa|_i^{1,2} = |\kappa_i^{1,2}| \tag{4.8}$$

$$(n \times e)_l = n_s \times (v_j - v_i) \tag{4.9}$$

The operation $\otimes$ is the outer vector product: $x \otimes y = x \cdot y^T \in \mathbb{R}^{3\times 3}$.

Remarks and explanations to equation (4.3) **-** (4.9)**.**

Equation (4.3) is not unique, since the facet f_s belongs to 3 vertices (in the case of a triangular mesh). We can use any two edges, to calculate the facet normal, as long as we presume orientation of the mesh (all normals point to one side of the mesh).

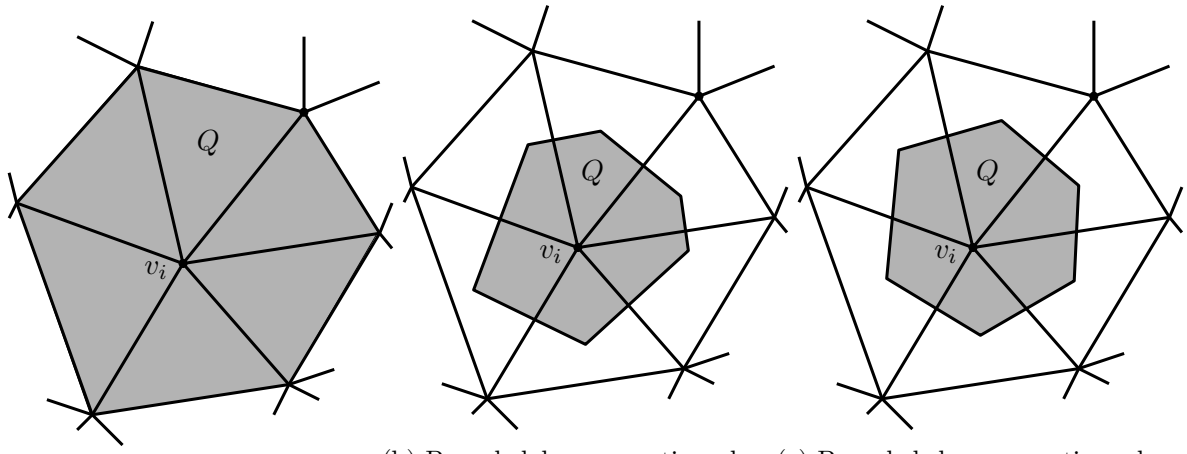
One out of two equations (4.4) is redundant, since $f_s \times f_t = -(f_t \times f_s)$. Anyways, we keep both for correct sign and orientation throughout.

Equation (4.5) is different to the previously derived definition of the discrete extended shape measure with swapped eigenvalues (definition 2.30). Let Q be a region of our polyhedral mesh and α_e the signed angle between adjacent facet normals. $\boldsymbol{e}$ defines the edge vector with $||\boldsymbol{e}|| = 1$. Then

$$\widetilde{\overline{W}}(Q) = -\sum_{e \in E} \text{len}(e \cap Q) \alpha_e \boldsymbol{e} \otimes \boldsymbol{e}$$

We fix Q to be the region around v_i including all adjacent facets. Different approaches can be used here (figure 4.2), but for sufficiently smooth meshes, they qualitatively give equal results. Similar to [29], we choose $Q = \bigcup_{f_s \sim v_i} f_s$ (figure 4.2a). This leads to

$$\text{len}(e \cap Q) = \begin{cases} \text{len}(e) & v_i \in e \\ 0 & v_i \notin e \end{cases} \qquad A_i := \text{area}(Q) = \sum_{fs \sim v_i} \text{area}(f_s)$$



(a) Bounded by adjacent vertices. (b) Bounded by connections between circumcircle centers. (c) Bounded by connections between center of masses.

Figure 4.2: Different choices of a region Q around vertex v_i in the definition of the extended shape operator measure.

The value A_i will be used in computation, but is not variable. We calculate these quantities at each iteration step separately and treat them as constants.

$\widetilde{\widetilde{W}}(Q)$ is a measure over Q. To obtain an operator at vertex v_i we define

$$W(v_i) := -\frac{1}{A_i} \sum_{e:v_i \in e} \text{len}(e)\alpha_e \boldsymbol{e} \otimes \boldsymbol{e}$$

For sufficiently smooth meshes, the small angle α_e can be approximated by $\sin(\alpha_e)$. With the left and right unit facet normals, we have $\sin(\alpha_e) = ||n_s \times n_t||$ and the edge vector is parallel to this cross product $\boldsymbol{e} \parallel n_s \times n_t$. We also have $\text{len}(e) = ||v_i - v_j||$ and $\boldsymbol{e} \parallel (v_i - v_j)$. Finally, both $n_s \times n_t$ and $v_i - v_j$ are pointing in the same direction for $\alpha_e > 0$ and in different directions for $\alpha_e < 0$. Therefore we have

$$W_i = -\frac{1}{A_i} \sum_{e:v_i \in e} (n_s \times n_t) \otimes (v_i - v_j)$$

This explains equation (4.5) and the introduction of the variables $(n \times n)_l$. This equation is again quadratic in our unknowns, for fixed area A_i. The resulting matrix W_i is obviously symmetric and we store only the 6 distinct entries. Figure 4.3 shows a graphic example of approximated principal curvature directions based on the definition of W_i.

The initialization of $a_i^1, a_i^2, \kappa_i^1, \kappa_i^2$ is in general fairly simple. We compute the 3 distinct eigenvectors to W_i with methods from linear algebra and compare their corresponding eigenvalues. The eigenvalue closest to 0 indicates the eigenvector in direction of the vertex normal (because of the construction of the *extended* shape operator, compare section 2.1.4). The other two eigenvectors are assigned to a_i^1, a_i^2 and the eigenvalues to κ_i^1, κ_i^2. However, we have to treat some special cases.

- If the smallest absolute eigenvalue is not especially close to 0, we have a bad mesh triangulation. In this case we can try get a better input data, or guess the two principal directions as the two eigenvectors, which are orthogonal to the vertex normal. This has shown to be more stable than simply choosing the two largest absolute values of the eigenvalues.

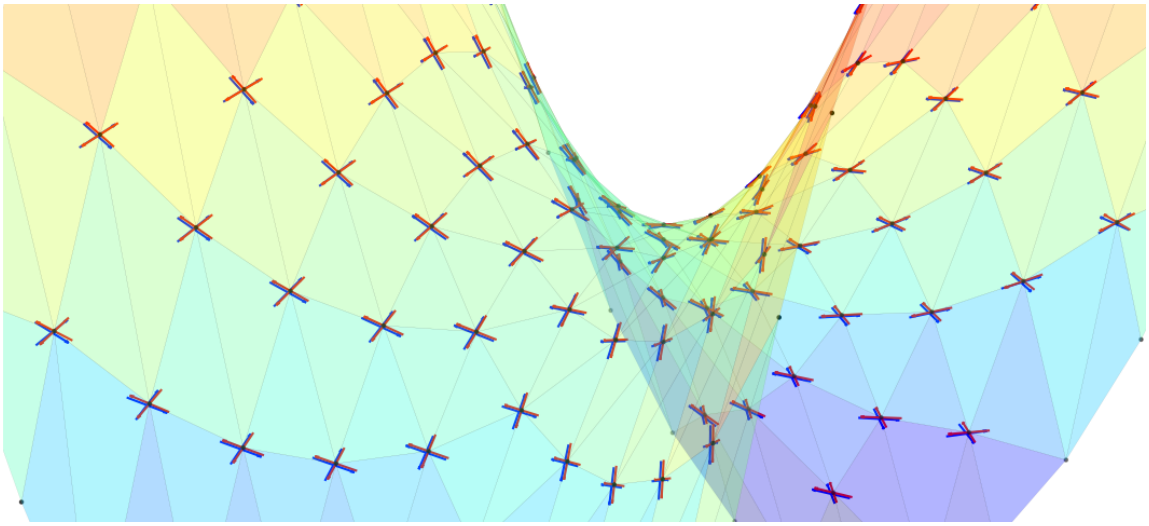


Figure 4.3: Approximation (blue) of exact (red) principal curvature directions with W_i. Surface function $\boldsymbol{x}(u, v) = [u, v, u^2 - v^2]$

- For some surfaces the principal curvatures are close to zero themselves (e.g. for a plane). In this case the eigenvectors form a 2- or 3-dimensional space and we might have eigenvectors not tangential or orthogonal to the surface at all. We set a_i^1 and a_i^2 orthogonal to the vertex normal n_i via a cross product.

In the end we have to normalize the vectors $a_i^{1,2}$.

The vertex normals n_i are used for initialization and also during the iteration process. Like the area elements A_i, we treat them as constants and update them at each step.

The definition in equation (4.9) is useful for the optimization of minimal surfaces and will be considered in section 5.1.

Vertices at the boundary play a special role. Definitions of W_i and other operators do not make sense and therefore we set these variables to 0 throughout (section 4.2.1).

4.2 Hard constraints

We introduced a set of variables and their initialization. In the guided projection setup, hard constraints play an important role. In general we aim to solve minimize equations (4.1) and solve equations (4.2) exactly. In this section we consider the second question, the constraints. To linearize these equations, we want them to be of quadratic order in the unknowns.

We will collect all unknowns in the large vector $\boldsymbol{x}$ of length N. A general quadratic hard constraint has the form

$$c_i(\boldsymbol{x}) = \boldsymbol{x}^T H_i \boldsymbol{x} + \boldsymbol{b}_i^T \boldsymbol{x} + d_i = 0$$

with some matrix $H_i \in \mathbb{R}^{N \times N}$, a vector $\boldsymbol{b}_i \in \mathbb{R}^N$ and a constant $d_i \in \mathbb{R}$.

Again, in this section we consider the case of triangular meshes. For polyhedral meshes some numbers may change. Equations and their quantities are listed here. We use again the same indices as in figure 4.1.

- The face normals yield two constraints, orthogonality and unit length.

$$\begin{aligned} n_s \cdot (v_j - v_i) &= 0 && 3|F| \text{ equations} \\ n_s \cdot n_s - 1 &= 0 && |F| \text{ equations} \end{aligned}$$

- The cross product between adjacent faces and the correlation between two halfedges at the same edge give

$$\begin{aligned} (n \times n)_l - n_s \times n_t &= 0 && 3 \cdot 2|E| \text{ equations} \\ (n \times n)_{l_1} - (n \times n)_{l_2} &= 0 && 3 \cdot 2|E| \text{ equations} \end{aligned}$$

  For each coordinate and for each halfedge one hard constraint is stated. Halfedges ending at a boundary vertex are not considered, therefore we may have fewer equations.

- The shape operator needs

$$A_i W_i + \sum_{l,j:e_l=(v_j,v_i)} (n \times n)_l \cdot (v_i - v_j)^T = 0 \qquad 6|V| \text{ equations}$$

  The 3×3 matrix W_i is symmetric and we only store and constrain 6 distinct entries. For vertices at the boundary we do not calculate the matrix W_i, we may have fewer equations. The area A_i, the sum of face areas of adjacent faces, is a constant in this equation, updated prior to the actual computation step of the unknown variables.

- The eigenvalue problem is stated as

$$\begin{aligned} W_i a_i^1 - \kappa_i^1 a_i^1 &= 0 && 3|V| \text{ equations} \\ W_i a_i^2 - \kappa_i^2 a_i^2 &= 0 && 3|V| \text{ equations} \end{aligned}$$

  Vertices at the boundary are omitted.

- Some other constraints for the principal curvatures were shown to be useful. The vertex normals n_i are constants.

$$\begin{aligned} a_i^1 a_i^2 &= 0 && |V| \text{ equations} \\ a^1 n_i &= 0 && |V| \text{ equations} \\ a^2 n_i &= 0 && |V| \text{ equations} \\ a_i^1 a_i^1 - 1 &= 0 && |V| \text{ equations} \\ a_i^2 a_i^2 - 1 &= 0 && |V| \text{ equations} \end{aligned}$$

- The absolute values of κ are also important. Note, that these equations are only piecewise linear and not differentiable at 0.

$$\begin{aligned} |\kappa_i^1| - |\kappa|_i^1 &= 0 && |V| \text{ equations} \\ |\kappa_i^2| - |\kappa|_i^2 &= 0 && |V| \text{ equations} \end{aligned} \tag{4.10}$$

- Also the special variables, used for minimal surfaces are constrained

$$(n \times e)_l - n_s \times (v_j - v_i) = 0 \qquad 3 \cdot 2|E| \text{ equations}$$

- A number of other constraints have been considered. Some of them have shown to be more useful than others, for example

$$\begin{aligned} \kappa_i^1 + \kappa_i^2 - \text{trace}(W_i) &= 0 && |V| \text{ equations} \\ W_i n_i &= 0 && 3|V| \text{ equations} \end{aligned}$$

Totally we have roughly $85|V|$ equations.

In practice it is not possible to fulfill all these equations. Especially when fairness and energy functions come into account. Therefore we are treating them in a similar manner as the fairness and energy functions and simply try to minimize them, but with different weights. More in section 4.5.

As presented in section 4.5, for a given vector $\boldsymbol{x}$ we need the first derivatives $\nabla c_i(\boldsymbol{x})$ of these equations. These are vectors of size $\mathbb{R}^N$. For the general case of a quadratic equation $c_i(\boldsymbol{x}) = \boldsymbol{x}^T H_i \boldsymbol{x} + \boldsymbol{b}_i^T \boldsymbol{x} + d_i$ the derivative is $\nabla c_i(\boldsymbol{x}) = H_i \boldsymbol{x} + \boldsymbol{b}_i$. To calculate the derivatives of the above equations is left to the reader. However, we have to clarify one exception. The equation $|\kappa_i^{1,2}| - |\kappa|_i^{1,2}$ is not quadratic and not even polynomial or differentiable. We can overcome this problem with the following idea.

Introduce a new pair of dummy variables at each vertex $\sqrt{|\kappa|}_i^1$ and $\sqrt{|\kappa|}_i^2$. They are, as the absolute value variables $|\kappa|_i^{1,2}$ indeed new unknowns, but with a name revealing their initialization and definition of constraints. A setup with truly quadratic equations would be (note the difference in the power of two $(\cdot)^2$ and the index ^{2}):

$$\begin{aligned} (\kappa_i^1)^2 - (|\kappa|_i^1)^2 &= 0 \\ (\kappa_i^2)^2 - (|\kappa|_i^2)^2 &= 0 \\ (\sqrt{|\kappa|}_i^1)^2 - |\kappa|_i^1 &= 0 \\ (\sqrt{|\kappa|}_i^2)^2 - |\kappa|_i^2 &= 0 \end{aligned} \tag{4.11}$$

If for given $\kappa_i^{1,2}$ variables $|\kappa|_i^{1,2}$ fulfill these equations (with dummy values $\sqrt{|\kappa|}_i^{1,2}$), they are for sure not negative and have the same absolute value as $\kappa_i^{1,2}$. Anyways, this approach has some downsides. We have to introduce even more variables, to achieve strictly non negative values and gain a pair of constraint equations with them. This enlarges our system. Additionally the convergence of $|\kappa|_i^{1,2}$ is worse. In section 4.5 we will see, that we linearize all equations. The quadratic relation in equations (4.11) will only be approximated, although an exact (piecewise) linear relation is actually at hand (4.10). We use this fact, omit the idea above and define the piecewise continuous derivatives to this functions.

$$\frac{\partial(|\kappa_i^1| - |\kappa|_i^1)}{\partial \kappa_i^1}(\boldsymbol{x}) = \begin{cases} 1 & \kappa_i^1 > 0 \\ 0 & \kappa_i^1 = 0 \\ -1 & \kappa_i^1 < 0 \end{cases} \qquad \frac{\partial(|\kappa_i^2| - |\kappa|_i^2)}{\partial \kappa_i^2}(\boldsymbol{x}) = \begin{cases} 1 & \kappa_i^2 > 0 \\ 0 & \kappa_i^2 = 0 \\ -1 & \kappa_i^2 < 0 \end{cases}$$

$$\frac{\partial(|\kappa_i^1| - |\kappa|_i^1)}{\partial |\kappa|_i^1}(\boldsymbol{x}) = -1 \qquad \frac{\partial(|\kappa_i^1| - |\kappa|_i^1)}{\partial |\kappa|_i^1}(\boldsymbol{x}) = -1$$

In practice the application of this leads to better results and convergence.

4.2.1 Boundary conditions

While solving the problem the vertices are almost freely floating around and the variables follow them with the described properties. To avoid unbounded behavior we must give some constraints at least on some vertices. This can be achieved in one of the following ways.

1. We do not define any equations on boundary vertices at all. Therefore they are not being changed in any way by the algorithm. We eliminate them of any equations by treating the coordinate entries of the boundary vertices as constants and reduce the vector of unknowns. This leads to a (slightly) smaller problem. The implementation of this approach is tricky, since many iterators (see section 4.6) do not treat boundary vertices in a special way and the constant values must be implemented in the system of equations carefully.
2. A similar approach is not to eliminate the boundary vertex coordinates, but to constrain them to be at their original position. We get additional equations

   $$v_i - v_i^* = 0$$

   for some boundary vertices and initial constant values v_i^*, that are not changed during the whole process. In this setup it is fairly important to assign strong weights to these boundary constraint equations, see section 4.5.
3. A more general attempt would be a boundary curve with free floating boundary vertices on this curve. Depending on the initial mesh, it may be useful to give this freedom to the boundary vertices to achieve a better solution.
4. Shape approximation is another way of restricting vertices: Some reference shape, provided by initialization or even changing during the process is approximated by the surface mesh and energy equations define the quality of the approximation. With this method a large number of applications is possible.

During the research of this thesis methods 1. and 2. have been considered and implemented. The second one gave slightly better results and we chose this for the results in section 5.

Method 3. and 4. were considered in many other papers, for instance in [30]. In [36] they provide also a setup of shape changing by the user and real-time computation of the mesh.

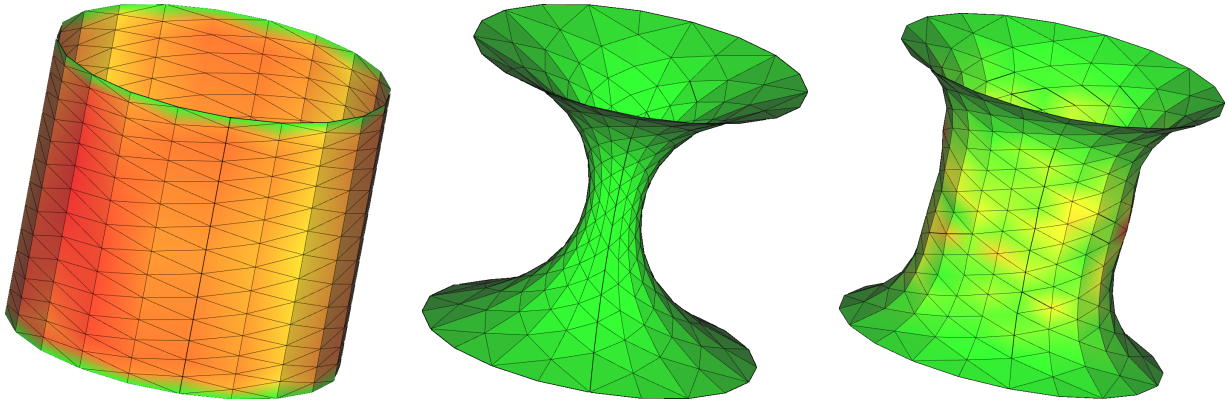
4.3 Fairness methods

We mentioned before, that a mesh needs some fairness energy, to avoid degeneration. Again, a number of different approaches can be used. Some examples are found in [19]. We introduce the most useful for our setup, similar to [36].

The general form of a fairness function is again at most quadratic.

$$f_j(\boldsymbol{x}) = \boldsymbol{x}^T H_j \boldsymbol{x} + \boldsymbol{b}_j^T \boldsymbol{x} + d_j \to 0$$

with some matrix $H_i \in \mathbb{R}^{N\times N}$, a vector $\boldsymbol{b}_j \in \mathbb{R}^N$ and a constant $d_j \in \mathbb{R}$.



(a) (b) (c)

Figure 4.4: Linear midpoint of adjacent vertices fairness for a cylinder with weight 1, in initial condition (a) and after one iteration (b). Squared distance to midpoint after 50 iterations (c). The colors indicate the fairness energy at each vertex from small (green) to high (red).

- One very popular idea is

$$\frac{1}{A_i}(v_i - \frac{1}{k}\sum_{v\sim v_i} v) \to 0 \qquad\qquad 3|V| \text{ equations}$$

  Here k denotes the number of neighbor vertices of v_i. Each coordinate is therefore linearly declared to be the barycenter of its neighbors. Of course, we have to be careful with the weights for this equation, since a restriction to this definition will lead to a stationary solution, somehow similarly to a minimal surface (figure 4.4).

- Similarly, we can define the quadratic distance of v_i to the barycenter of its neighbors.

$$\frac{1}{A_i^2}\left\|v_i - \frac{1}{k}\sum_{v\sim v_i} v\right\|^2 \to 0 \qquad\qquad |V| \text{ equations}$$

  In a similar way, this energy behaves like a minimizer of the surface area, but with more focus on outliers, due to the quadratic behavior. In our research this energy provided worse results and convergence behavior (figure 4.4 (c)).

  Better results were achieved with squared distances in each coordinate separately.

- An idea to avoid the strong influence of the first two fairness energies is the *tangential projection fairness*. The neighbors of v_i are projected onto the tangent plane in v_i defined by the vertex normal n_i and then their barycenter is calculated. v_i is demanded to be this point. The process is happening in the 2-dimensional tangent plane and therefore only 2 equations (in the linear case) are necessary, compared to 3 in the regular case. Given a tangential orthonormal basis $\{t_i^1, t_i^2\}$ we have

$$\frac{1}{A_i}(v_i \cdot t_i^1 - \frac{1}{k}\sum_{v\sim v_i} v \cdot t_i^1) \to 0 \qquad\qquad |V| \text{ equations}$$

$$\frac{1}{A_i}(v_i \cdot t_i^2 - \frac{1}{k}\sum_{v\sim v_i} v \cdot t_i^2) \to 0 \qquad\qquad |V| \text{ equations}$$

This method leads to less shrinking while obtaining a fairly good mesh. A downside is the dependency of tangential vectors (which can be computed from n_i). For meshes which are not smooth, the vertex normals are not close to their desired values and so are the tangent planes. Convergence to a smooth mesh is achieved harder.

- A method, considered in [36] is fairness of the Gauß image, the vertex normals n_i respectively. To mix those methods, we have to consider the scale of the mesh and the corresponding Gauß image.
- Principal curvature and principal directions can be considered in fairness ideas as well. Smooth meshes have small changes in curvature along adjacent vertices and therefore a fairness, similarly to the vertex fairnesses, is possible. However, energies that work directly on the curvatures influence them as well and in practice they often conflict. Weights for such fairing functions must be relatively small.
- To avoid a strong smoothing influence by the fairness, we come up with another approach. A nice representation of a mesh has equally distributed edge lengths, in particular the edges of one facet f_s should be more or less of same length. For vertex v_i denote by v_j, v_k the other vertices of facet f_s.

$$\frac{1}{A_i}[(v_j - v_i) \cdot (v_j - v_i) - (v_k - v_i) \cdot (v_k - v_i)] =$$
$$\frac{1}{A_i}[v_j \cdot v_j - v_k \cdot v_k - 2v_j \cdot v_i + 2v_k \cdot v_i] \to 0 \qquad 3|F| \text{ equations}$$

The factor $\frac{1}{A_i}$ is important for correct scaling. For example, a vertex 1 unit off its barycenter of neighbors, can be sufficiently good for a large sized triangles, but even outside of its neighbor ring, for triangles with less than $\frac{1}{10}$ unit edge length.

In application we mostly use linear and tangential fairness of the vertices, as well as edge length fairness.

4.4 Energy

Depending on the special application, we introduce a quadratic energy function, which is in general of the form

$$E_k(\boldsymbol{x}) = \boldsymbol{x}^T H_k \boldsymbol{x} + \boldsymbol{b}_k^T \boldsymbol{x} + d_k \to 0$$

We want to minimize this energy, to achieve a surface with a special property. For example minimizing the mean curvature $H = \frac{\kappa_1 + \kappa_2}{2}$ gives a linear energy function in the two principal curvatures. In section 5 we introduce a variety of them and present different results obtained by varying energies.

4.5 Guided projection

Based on our set of quadratic equations c_i, f_j, E_k we derive a linear least square problem. According to chapter 3, each quadratic function can be linearized via first order Taylor expansion. We show this using the example of c_i

$$c_i(\boldsymbol{x}_0 + \Delta\boldsymbol{x}) \approx c_i(\boldsymbol{x}_0) + \nabla c_i(\boldsymbol{x}_0) \cdot \Delta\boldsymbol{x}$$

In the notation from above, the gradient is $\nabla c_i(\boldsymbol{x}) = H_i\boldsymbol{x} + \boldsymbol{b}_i$ and the problem

$$c_i(\boldsymbol{x}_0) + \nabla c_i(\boldsymbol{x}_0) \cdot \Delta\boldsymbol{x} = 0, \quad i = 1, \ldots, l$$

can be written in matrix notation as

$$\begin{pmatrix} (H_1\boldsymbol{x}_0 + \boldsymbol{b}_1)^T \\ \vdots \\ (H_l\boldsymbol{x}_0 + \boldsymbol{b}_l)^T \end{pmatrix} \Delta\boldsymbol{x} = - \begin{pmatrix} \boldsymbol{x}_0^T H_1\boldsymbol{x}_0 + \boldsymbol{b}_1^T\boldsymbol{x}_0 + d_1 \\ \vdots \\ \boldsymbol{x}_0^T H_l\boldsymbol{x}_0 + \boldsymbol{b}_l^T\boldsymbol{x}_0 + d_l \end{pmatrix}$$

This is a standard linear system for the update vector $\Delta\boldsymbol{x}$. This system has more equations than unknowns and is therefore overdetermined. However, some equations are redundant or at least "almost redundant" (due to discretization errors). The fairness functions f_j provide some stability, and the energy functions E_k too. We also avoid instability by using the distance to a previous approximation as a regularization $||(\boldsymbol{x}_0 + \Delta\boldsymbol{x}_0) - \boldsymbol{x}_0||^2 = ||\Delta\boldsymbol{x}||^2 \to \min$. Our hard constraints should be approximated as good as possible, while regularization only works as *guidance*. We have to use large and small weights, to express this behavior. Together we obtain the *linear least square problem*

$$\underbrace{\left\| \begin{pmatrix} u_1\nabla c_1(\boldsymbol{x}_0)^T \\ \vdots \\ u_l\nabla c_l(\boldsymbol{x}_0)^T \end{pmatrix} \Delta\boldsymbol{x} + \begin{pmatrix} u_1 c_1(\boldsymbol{x}_0) \\ \vdots \\ u_l c_l(\boldsymbol{x}_0) \end{pmatrix} \right\|^2}_{\text{hard constraints}} + \underbrace{\left\| \begin{pmatrix} v_1\nabla f_1(\boldsymbol{x}_0)^T \\ \vdots \\ v_m\nabla f_m(\boldsymbol{x}_0)^T \end{pmatrix} \Delta\boldsymbol{x} + \begin{pmatrix} v_1 f_1(\boldsymbol{x}_0) \\ \vdots \\ v_m f_m(\boldsymbol{x}_0) \end{pmatrix} \right\|^2}_{\text{fairness}} + \tag{4.12}$$

$$+ \underbrace{\left\| \begin{pmatrix} w_1\nabla E_1(\boldsymbol{x}_0)^T \\ \vdots \\ w_n\nabla E_n(\boldsymbol{x}_0)^T \end{pmatrix} \Delta\boldsymbol{x} + \begin{pmatrix} w_1 E_1(\boldsymbol{x}_0) \\ \vdots \\ w_n E_n(\boldsymbol{x}_0) \end{pmatrix} \right\|^2}_{\text{target energy}} + \underbrace{\left\| \varepsilon \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \Delta\boldsymbol{x} + \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\|^2}_{\text{regularization}} \longrightarrow \min$$

To fit this into the least square setup with one matrix and one vector from chapter 3, we note that combining the matrices and vectors to one large system leads to the same solution. For matrices and vectors $A^1 \in \mathbb{R}^{m_1 \times n}$, $A^2 \in \mathbb{R}^{m_2 \times n}$, $\boldsymbol{b}^1 \in \mathbb{R}^{m_1}$, $\boldsymbol{b}^2 \in \mathbb{R}^{m_2}$, denote with A^i_j the j-th row and with b^i_j the j-th coordinate. Then

$$\left\|A^1\boldsymbol{x} + \boldsymbol{b}^1\right\|^2 + \left\|A^2\boldsymbol{x} + \boldsymbol{b}^2\right\|^2 = \sum_{j=1}^{m_1}(A^1_j\boldsymbol{x} + b^1_j)^2 + \sum_{j=1}^{m_2}(A^2_j\boldsymbol{x} + b^2_j)^2 = \left\| \begin{pmatrix} A^1 \\ A^2 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} \boldsymbol{b}^1 \\ \boldsymbol{b}^2 \end{pmatrix} \right\|^2$$

A solution of this problem is achieved according to section 3.2. We linearized the quadratic equations, therefore the result $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \Delta\boldsymbol{x}$ may not solve our initial problem optimally. Another iteration with the new starting guess $\boldsymbol{x}_1$ gives another result. This is in general $\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \Delta\boldsymbol{x}$ with a solution $\Delta\boldsymbol{x}$ of (4.12) to $\boldsymbol{x}_{j-1}$.

We summarize the guided projection steps.

0. Define the unknowns $\boldsymbol{x}$ and their hard constraints c_i. Define the fairness functions f_j and the target energies E_k. All equations must be quadratic in the vector $\boldsymbol{x}$. Define the weights $u_i, v_j, w_k, \varepsilon$.
1. Initialize the vector $\boldsymbol{x}_0$.
2. In step j solve the linearized least square problem (4.12) and update $\boldsymbol{x}_j$. Update the constants A_i, n_i.
3. Repeat this process iteratively until a convergent solution is found or some halt condition is fulfilled.

We point out, that this method differs from the problem, stated at the very beginning of this chapter. Instead of the solution of a least square problem (4.1) under the constraints of (4.2), we solve one single unconstrained least squares problem with the methods of chapter 3.

4.5.1 Weights

The weights u_i, v_j, w_k play an important role in the guided projection setup. In general the weights are of the following sizes.

- u_i: The weights for the hard constraints are in a magnitude of 1. They assure correct dependencies between the introduced variables.
- v_j: These weights for the fairness equations are of magnitude $10^{-3} - 10^{-5}$. Smaller values can result in clunky meshes, while larger values have a lot of influence on the shape.
- w_k: The weights for the target energies are of special importance. They are in a magnitude of 10^{-2} to 10^{-5}. A stronger influence leads to desired results, but can deform the mesh or end in no convergence.

Specifically some weights are discussed here. However, they have to be considered carefully from one application to another and our description may not be optimal for other purposes.

The first group of equations are some of our hard constraints. We found out, that the two sets of equations for the face normals (orthogonality to the edges and unit length) need strong weights, namely $u_i \approx 1$. The same holds true for the cross product variables $(n \times n)_l$. Their equations (cross product of face normals and correlation between halfedges on the same edge) need weights of size 1 too.

In our application the equations for W_i had slightly smaller weights, with values of $\frac{1}{2}$. We trace this back to the discretization error in W_i and give more freedom to these variables. Energies have more influence on the entries in the matrix and this can change the mesh closer to desired results. Similarly the eigenvalue equations $Wa = \kappa a$ have weights of size $\frac{1}{2}$.

The eigenvectors/principal curvature directions however should be orthogonal to each other and of unit length. We use weights of size 1, because these properties should be strictly conserved. Loose equations can lead to swapped principal curvatures and unexpected behavior.

Two equations, that turned out to influence the behavior quite a lot are the orthogonality between the principal directions and the vertex normal, as well as the conditions $\kappa_i^1 + \kappa_i^2 = \text{trace}(W_i)$ and $W_i n_i = 0$. The first was shown to be quite not fulfilled, even after initialization.

The inner product between principal direction and approximated vertex normal was as large as 10^{-2}. Even worse, a strong weight on the equation led to much influence on the output result. Strong fairness energies had to be applied, to prevent this performance, but then energies in general needed a strong weight, overruling the hard constraints. Similar observations were made with the second condition. In practice the following handling turned out to be useful. Far away from our expected solution, at the first steps of our iteration process, we put weights in the size of 10^{-3} on these equations. This prevents eigenvalues and -vectors far off principal curvatures and directions. Closer to our desired result, we lower these weights down to 0 and obtain a stable and convergent solution.

The piecewise linear constraints on the absolute values of principal curvatures (4.10) can be weighted with $u_i = 1$.

Boundary conditions are weighted with strong weights, like 100 or more. Giving some flexibility with small weights to those equations can be helpful to find a result, but should be considered with care, since the mesh can deform quickly.

For fairness it turned out, that sometimes the weights v_i had to be adapted during the process. Similarly to above, we start with somehow larger values to achieve useful results, but lower them to smaller values, giving more influence to the energy function at the end. The linear midpoint fairness, as well as the tangential midpoint fairness, described in section 4.3 are weighted with values of magnitude 10^{-3} at the beginning and are dropped as small as 10^{-5} in the end. The fairness of edge length is weighted with sizes around 10^{-6}. However, in all cases, the actual object size comes into account, since larger absolute edge lengths give more influence on all these equations. Therefore the weights have to be adapted according to the total size of the mesh and the refinement, leading to different edge lengths. Weights on other fairness equations, like κ values and so on must be treated with care, since they influence the output mesh in a strong way. Weights for these equations have to be set to 0 close at a result. Otherwise they prevent a good convergence, as our experiments have shown.

The energies must be treated in the opposite way. Starting with small values of about 10^{-5} give fairness methods more power, but can be increased up to 10^{-2} or even 10^{-1} to achieve stable results. The weights depend on the kind of energy, we use in application, considered in chapter 5.

Last the weight ε for the regularizing identity matrix is discussed. It can be fairly small, in magnitudes of 10^{-6} down to 10^{-10}. We experimented with stronger weights on some of the matrix entries, namely on the vertex coordinates. These affected variables did change less and the other constraints converged quite fast. However it is not practicable for application, since a stationary solution, almost not different to the initialization is quite useless, but can be used for testing of the implementation.

In practice we use equal weights on equal classes of equations. Theoretically our setup allows completely different weights on every single equation. We experimented with adaptive values, motivated by the theory of *reweighted linear least square problems* [7]. Especially for fairness functions it would be useful, to weight strong outliers with larger values, to bring them back into a useful area. Experiments showed no significant improvements with such techniques and we therefore restrict our application to equal weights for the same class of functions. However, as mentioned above, changing these weights for the whole class during the iteration process is used.

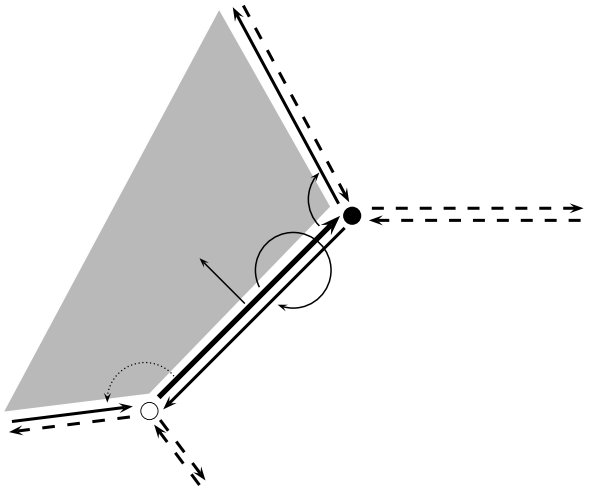


Figure 4.5: The halfedge data structure and its linked objects in OpenMesh.

4.6 Software

The implementation code itself is written in *C++*. Some libraries and external programs are used to meet the necessary requirements. A short introduction to these resources and their usage in our application is given on the next pages.

OpenMesh

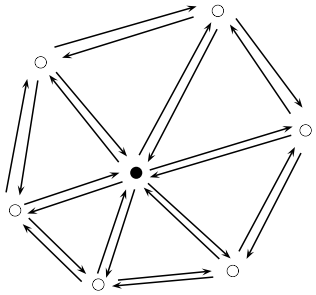
OpenMesh [5] is a data structure, to represent and manipulate polygonal meshes. It is developed at the Computer Graphics Group, RWTH Aachen. With OpenMesh we can store information about a polygonal mesh, like vertices, edges, facets and the connectivity between them, but also access these values in an efficient way. The open source structure comes as a C++ library and has also python bindings, which makes it possible to use in both languages.

The main concept of OpenMesh is the *halfedge data structure*. Descriptively this is an oriented edge, pointing from one vertex to another. It holds information about the following objects (see also figure 4.5).

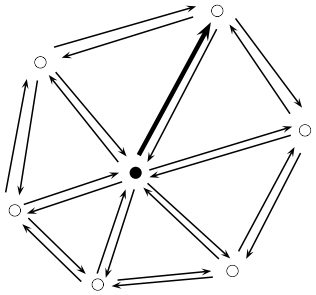
- the vertex it points to
- the facet it belongs to
- the next halfedge in this facet
- the opposite halfedge on the same edge
- optional: the previous halfedge

OpenMesh provides various iterators for all purposes. A very common access problem is the so-called *one-ring* neighborhood of a vertex. With the halfedge data structure this can be done in an efficient way. Figure 4.6 illustrates the procedure.

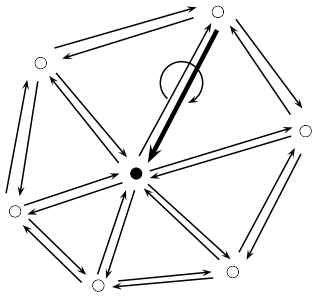
Moreover, OpenMesh supports default and arbitrary properties on its objects, like colors and other graphical data.



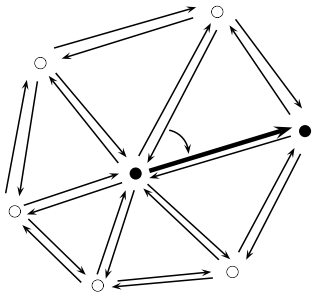
(a) Begin with a vertex.



(b) Find one outgoing halfedge at this vertex.



(c) Switch to the opposite halfedge.



(d) Take next halfedge, pointing to the next vertex.

Figure 4.6: Iterating over the vertices of the one-ring of a vertex v in OpenMesh using the halfedge data structure. Repeat steps (c) and (d) until all vertices are traversed.

Eigen

Eigen [15] is an open source C++ library with template headers. It is used for linear algebra related problems on matrices and vectors, both in dense and sparse case. The library provides many numerical solvers for exact or iterative solutions and comes with other related algorithms as well.

In our application we use the data structure for vectors, small 3×3 and large sparse matrices and use solvers for eigenvalue problems and large linear systems. In particular, we use a preconditioned CG method for large linear systems, and QR factorization for small least square problems, arising in the computation of the osculating paraboloid.

For parallelization we made use of the *OpenMP* library[(3)].

Evolute

For visualization and basic functionality, we use parts of the software EvoluteTools [32]. Our C++ code is imported as a plugin. The graphical interface is implemented with Qt[(4)].

[(3)] https://www.openmp.org/
[(4)] https://www.qt.io/

Python mesh generator

The generation of all presented meshes is done with a self written *Python3*[5] program. We can produce different geometrical shapes, based on their parametrization and apply some perturbation. The mesh can be triangular or quad (figure 4.7). Data is stored in the **.obj*[6] format and then imported into Evolute for processing.

[5] https://www.python.org/
[6] https://www.fileformat.info/format/wavefrontobj/egff.htm

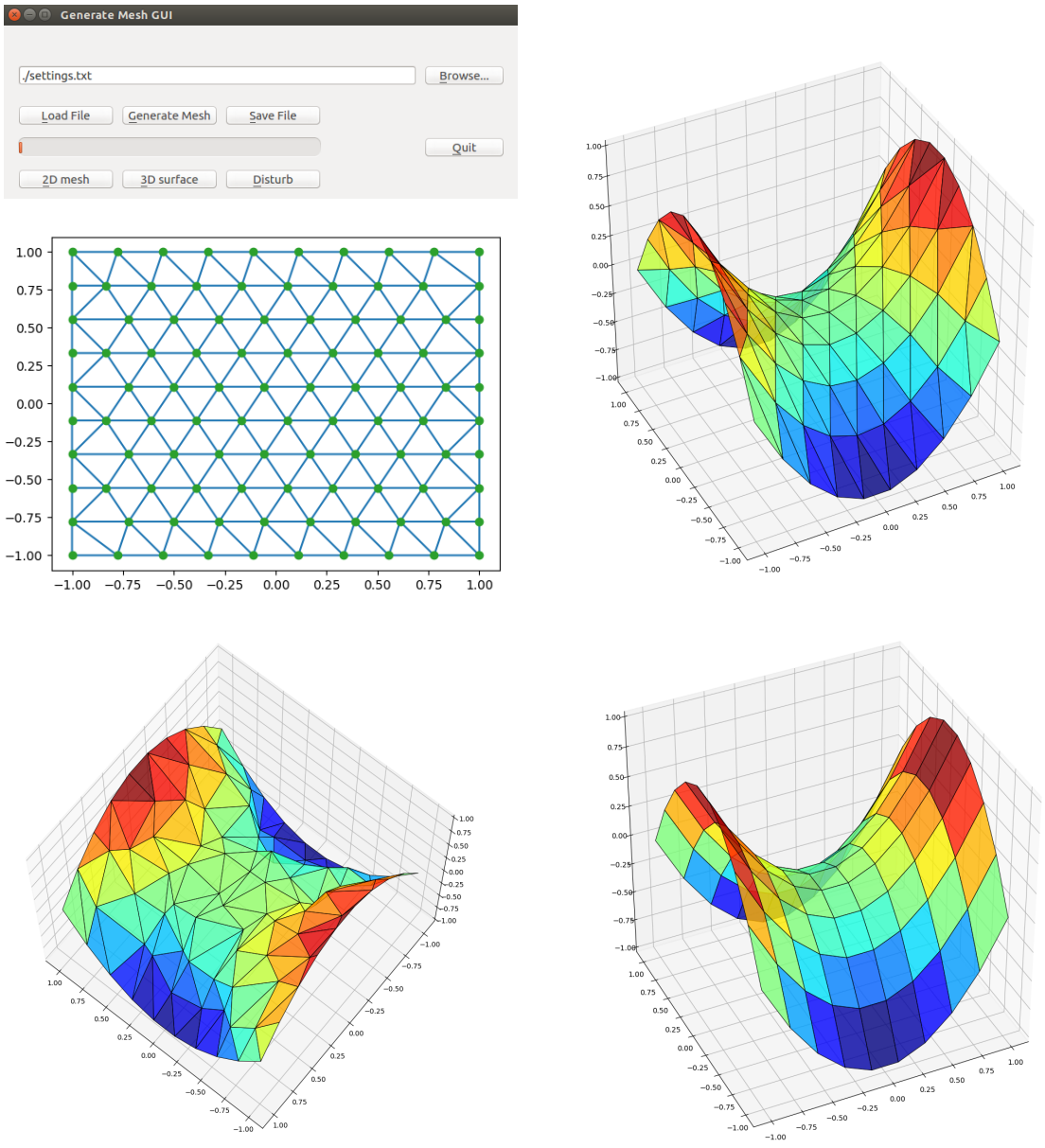


Figure 4.7: From top left to bottom right: Python3 mesh generator user interface, 2D Delaunay triangle mesh, 3D triangular mesh, 3D triangular mesh with perturbation, 3D quad mesh.

5 Application

In this chapter we present various results of application of our guided projection method. The large number of different variables in the setup can be used to describe a wide variety of problems, all with some energy integral, depending on those variables. Mainly we focus on energy functions $E(\kappa_1, \kappa_2)$ depending on the principal curvatures.

In the first section 5.1, minimal surfaces are discussed. Therefore we use the mean curvature H and its different representations in the discrete setup. In section 5.2 we discuss the more general case of surfaces, with constant mean curvature. It is followed by section 5.3, about developable surfaces, with one principal curvature $k_i = 0$. Section 5.4 again is a generalization about surfaces with constant Gauß curvature. Sections 5.5 and 5.6 discuss different other energy functions, involving κ_i, κ_i^2 and $|\kappa_i|$. The last section 5.7 is about quad meshes and their characteristic, e.g. planarity.

In this chapter we use the notation (V, E, F) as before. The number of vertices, edges and facets are $|V|, |E|, |F|$. Also we define

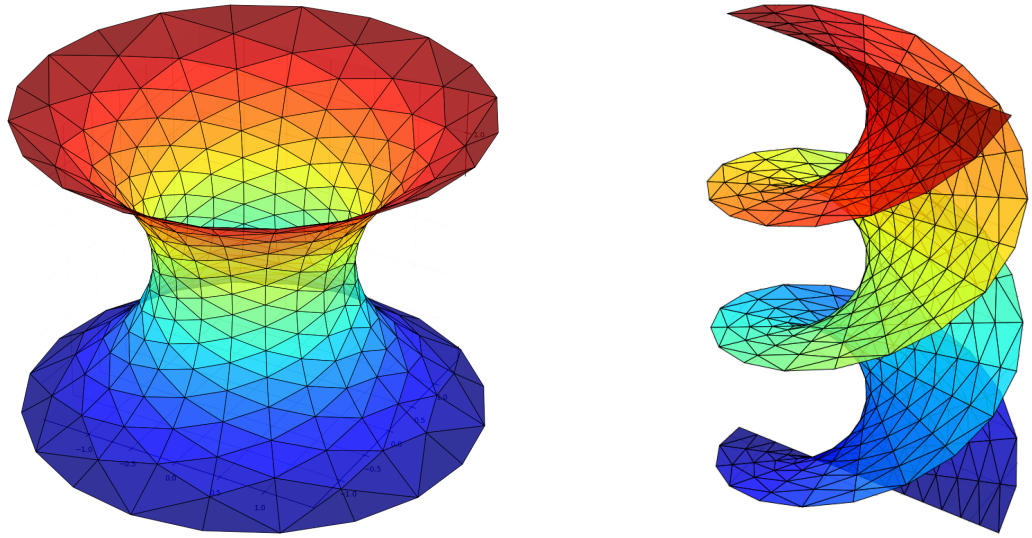
$$n := |\{v \in V | v \notin \delta S\}|$$

the number of non-boundary vertices.

Our results are in general presented in two blocks:

- A table with numerical quantities (at the beginning and the end of one computation), like
  - Total, hard constraint, fairness and target energies
  - Estimated target energy, obtained by osculating paraboloid (section 2.2.4)
  - Computation time
  - Number of iterations
  - Mesh properties like $|V|$, $|E|$, $|F|$
- A figure with graphical results, including
  - The initial condition
  - Intermediate conditions
  - The final result
  - A graphical representation of the energies during the computation

For our computations, we used a Laptop with Intel®Core™i7-6820HQ CPU (4 cores, 2 threads each, 2.7 GHz) and 16GB of RAM.



(a) Catenoid (b) Helicoid

Figure 5.1: Examples of minimal surfaces.

5.1 Minimal surfaces

The search for minimal surfaces is of great interest and a large number of publications is available [20, 9]. In section 2.1 we discussed the relation between the mean curvature H and smooth surfaces with minimal area to boundary conditions. In section 2.2 we presented different approaches to define the mean curvature in the discrete case, and following that, we will use those different definitions to state our problem.

- $\overrightarrow{H}(v) = \mathbf{0}, \quad \forall v \in V$ with $\overrightarrow{H}$ defined in 2.31 (the second version of H)
- $\kappa_1(v) + \kappa_2(v) = 0, \quad \forall v \in V$ with the eigenvalues κ_i of the discrete shape operator $\overline{W}(v)$, defined in 2.30 (corresponds to the first version of H)
- $\text{trace}(\overline{W}(v)) = 0, \quad \forall v \in V \quad \overline{W}$ also defined in 2.30

Minimal surfaces are for example the plane, catenoids and helicoids (figure 5.1).

Version $H(v)$

According to proposition 2.32 and2.33 we obtain a surface with minimal area by optimizing our mesh, such that $\mathbf{0} = \overrightarrow{H}(v) = \frac{1}{2}\sum_{v_i:v_i\sim v} J_i(v_i - v_{i+1})$ for all vertices. Our target energy at each vertex is therefore

$$E_k(\boldsymbol{x}) = A_k \sum_l (e \times n)_l,$$

with all l such that it belongs to a halfedge in a facet of v_k but does not include v_k (so it is opposite to the vertex). We have $3n$ equations, the boundary vertices are fixed via large constraint weights.

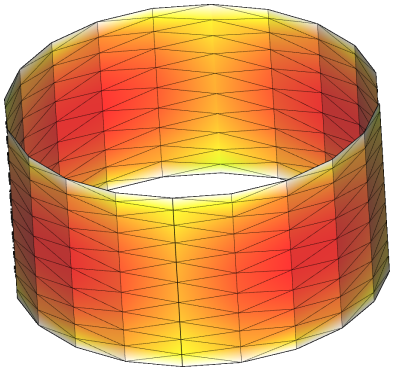
This energy, introduced in [31], is used for many applications and under constant research (see [3] and [26]).

Example figure 5.2 **Surface type**	**Original** cylinder	**Result** catenoid
Total energy	$7.64 \cdot 10^{-1}$	$1.73 \cdot 10^{-3}$
Hard constraints energy	$1.35 \cdot 10^{-13}$	$8.01 \cdot 10^{-4}$
Fairness energy	$1.34 \cdot 10^{-3}$	$7.69 \cdot 10^{-4}$
Target energy	$7.62 \cdot 10^{-1}$	$2.13 \cdot 10^{-4}$
- max. at vertex	$5.44 \cdot 10^{-3}$	$5.81 \cdot 10^{-5}$
H estimated with osculating paraboloid	$1.85 \cdot 10^{-2}$	$1.18 \cdot 10^{-3}$
Computation time:		26.1 s
Number of iterations:		8
220 vertices	620 edges	400 facets

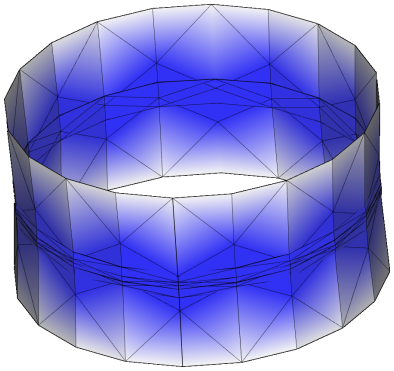
We chose the weights on the fairness energy equations relatively small. During the optimization process, this leads to increasing energy, but realigns in the end. Larger weights would lead to slower convergence, while smaller weights would end in degenerated meshes, which can not "unfold" back to a fair distribution of vertices.

The estimated target energy, which gives a measurement of the quality of our result, is of course also only an approximation. According to experiments, in this example our chosen implementation (which can be enhanced) does not entirely represent the actual mean curvature, but gives a good measure of the quality of our result.

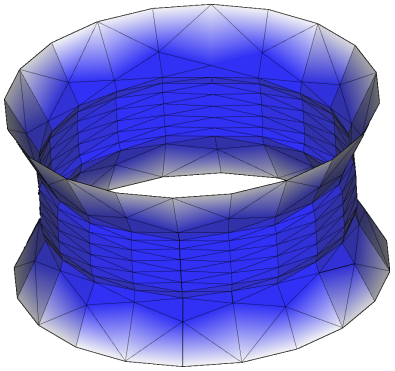
Compared to other target energy definitions, this method is relatively cheap to solve and we do not need a lot of iterations to achieve good results. This is mainly because of the few involved hard constraint equations. We can skip everything, involving W, a_1, a_2 and curvature and only rely on the introduced variables $(n \times e)_l$. Therefore this choice of target energy and setup is special among the following parts, which all involve the definition of W and use the curvature variables (but not $(n \times e)$).



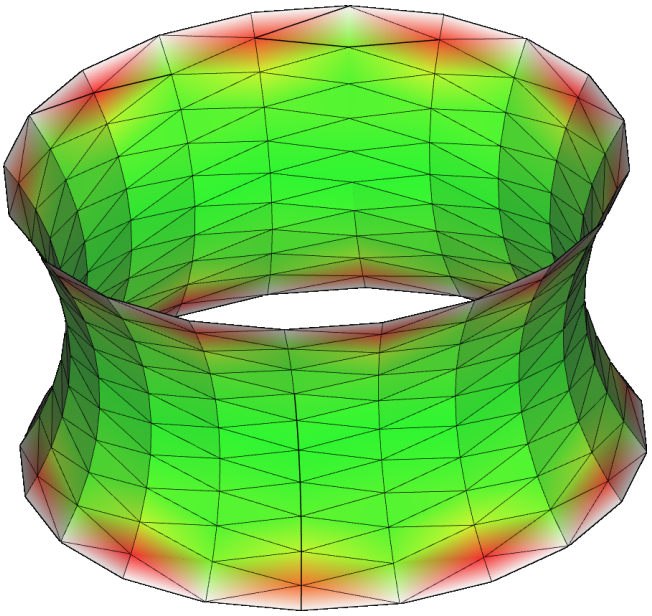
(a) Initial position. ($5.44 \cdot 10^{-3}$ max.)



(b) After 1 step. Fairness weights are chosen small.

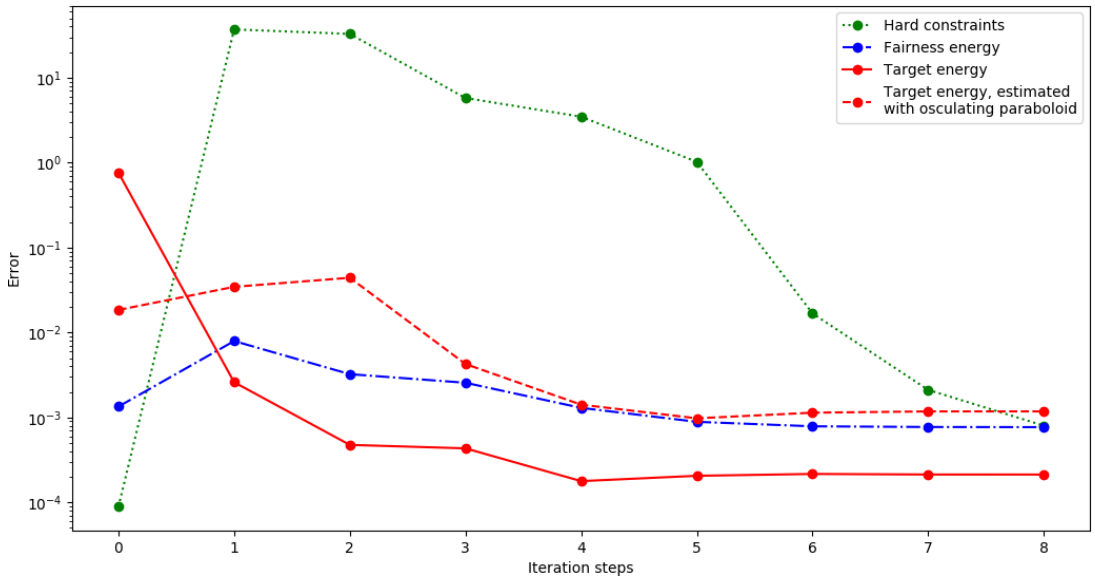


(c) After 2 iteration steps. The mesh realigns to smaller fairness energy.



(d) Result after 8 iteration steps. Target energy is close to zero. ($5.81 \cdot 10^{-5}$ max.)

Figure 5.2
Minimizing $\vec{H}(v)$ on a cylinder. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) Colors in figure (b,c) indicate the absolute change of curvature in each vertex for this step (bright: small, dark: big).



(e) Small weights for fairness energy allow fast mesh deformation (b,c). In the end fairness is restored and the target energy is close to 0. Reinitialization proofed the result to be almost a catenoid, even if the approximation with the osculating paraboloid indicates some offset.

Version $\kappa_1 + \kappa_2$

We want to define the condition of zero mean curvature directly with the two principal curvatures κ_1 and κ_2. The target energy is defined as

$$E_k(\boldsymbol{x}) = (\kappa_k^1 + \kappa_k^2)A_k$$

This leads to n target energy equations.

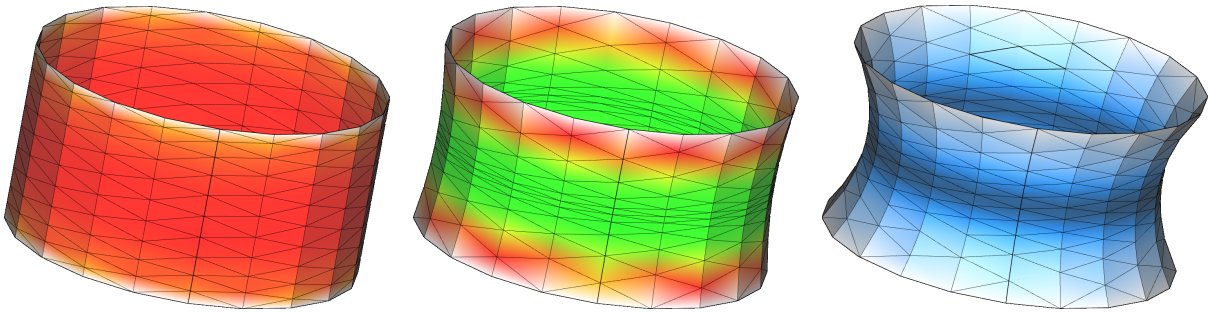
Scientific work, involving the discrete extended shape operator and its application in optimization can be found in [29]. D. Pellis and H. Pottmann use guided projection to align principal curvature directions with principal stress directions.

Since the definition of W and its application is a main part of this thesis, we will present 3 independent examples for this method.

Example figure 5.3 **Surface type**	**Original** cylinder	**Result** catenoid
Total energy	$1.34 \cdot 10^{-1}$	$1.50 \cdot 10^{-2}$
Hard constraints energy	$1.58 \cdot 10^{-13}$	$1.14 \cdot 10^{-2}$
Fairness energy	$1.46 \cdot 10^{-2}$	$9.52 \cdot 10^{-3}$
Target energy	$5.96 \cdot 10^{-2}$	$1.26 \cdot 10^{-3}$
- max. at vertex	$2.02 \cdot 10^{0}$	$1.38 \cdot 10^{-1}$
H estimated with osculating paraboloid	$1.85 \cdot 10^{-2}$	$1.30 \cdot 10^{-3}$
Computation time:		76.1 s
Number of iterations:		15
220 vertices	620 edges	400 facets

This example is relatively easy to be solved. Remarkable is the fast and stable convergence (fig 5.3e). The large jump of hard constraint energy in step one is explained as followed: The initialization is naturally exact for most of the hard constraint equations. Starting the optimization process, especially the target energy influences some of the variables to its "desired" values. In the following steps the mesh has to transform into a position, where both, the target energy and the hard constraints are fulfilled as good as possible. The fairness energy comes into account, to prevent degeneration - for example it prevents triangles in the middle of the cylinder to collapse more than they do in figure 5.3b.

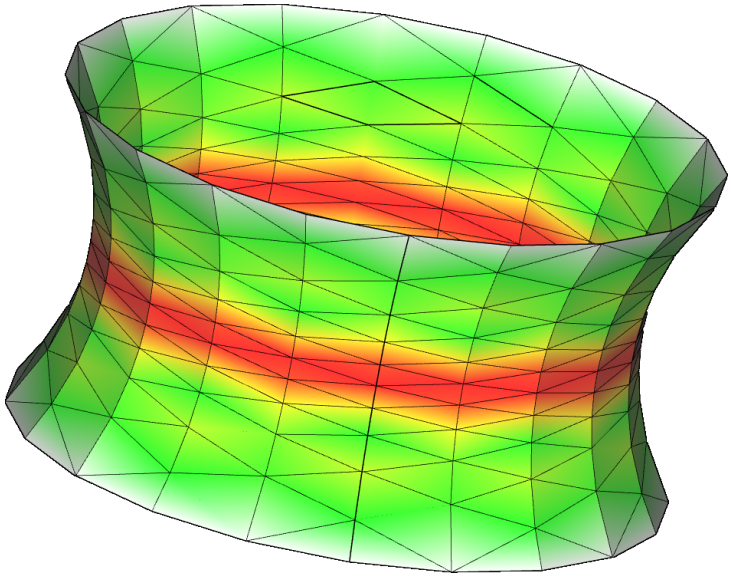
Here the important role of the weights is clearly visible. We have to keep them as loose as possible, to allow the target energy to influence the mesh, but especially the fairness weights must not be to small or large. Either the mesh collapses or they overrule the target energy. In our example we were bounded to a factor between $\frac{1}{5}$ and 5 for each weight before convergence was lost.



(a) Initial position. ($2.02 \cdot 10^{0}$ max.)

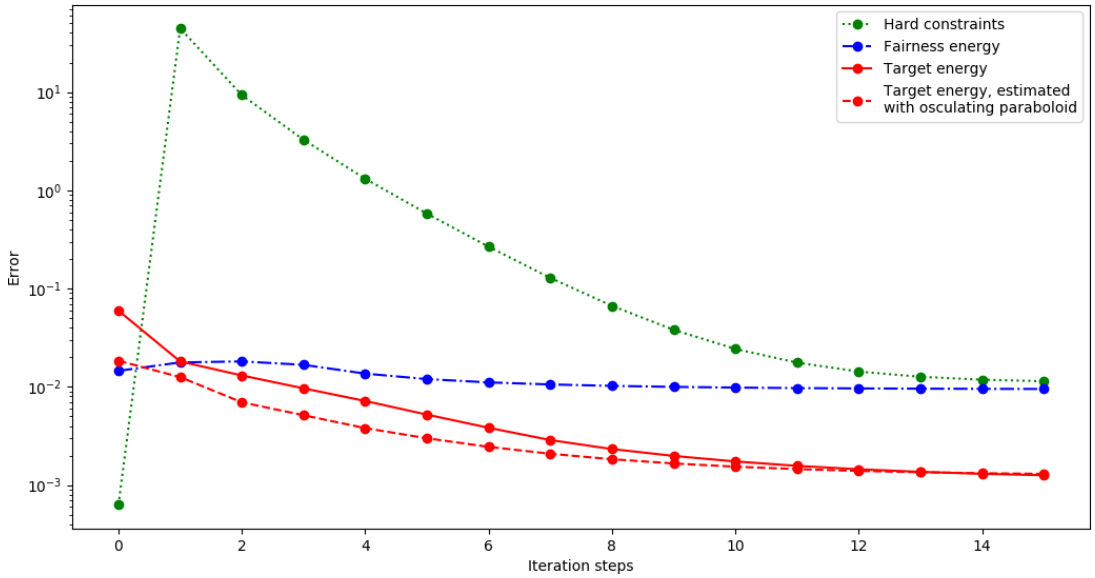
(b) 1 iteration. ($7.08 \cdot 10^{-1}$ max.)

(c) Result with changes in κ.



(d) The resulting mesh. ($1.38 \cdot 10^{-1}$ max.)

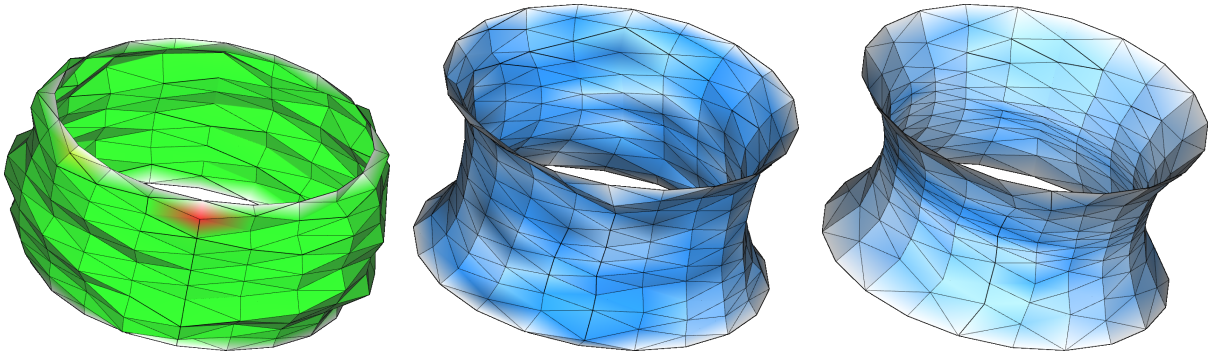
Figure 5.3
Minimizing $\kappa_1 + \kappa_2$ on a cylinder. Colors in figure (a,b,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) Colors in figure (c) indicate the change of curvature in each vertex.



(e) The energies during the optimization process.

Example figure 5.4	**Original**	**Result**
Surface type	bulgy perturbed cylinder	catenoid
Total energy	$8.60 \cdot 10^{-1}$	$1.39 \cdot 10^{-2}$
Hard constraints energy	$6.61 \cdot 10^{-2}$	$3.97 \cdot 10^{-2}$
Fairness energy	$1.28 \cdot 10^{-1}$	$9.99 \cdot 10^{-3}$
Target energy	$3.66 \cdot 10^{-1}$	$7.59 \cdot 10^{-4}$
- max. at vertex	$2.33 \cdot 10^{+1}$	$9.84 \cdot 10^{-2}$
H estimated with osculating paraboloid	$1.09 \cdot 10^{-1}$	$9.47 \cdot 10^{-4}$
Computation time:		70.8 s
Number of iterations:		13
220 vertices	620 edges	400 facets

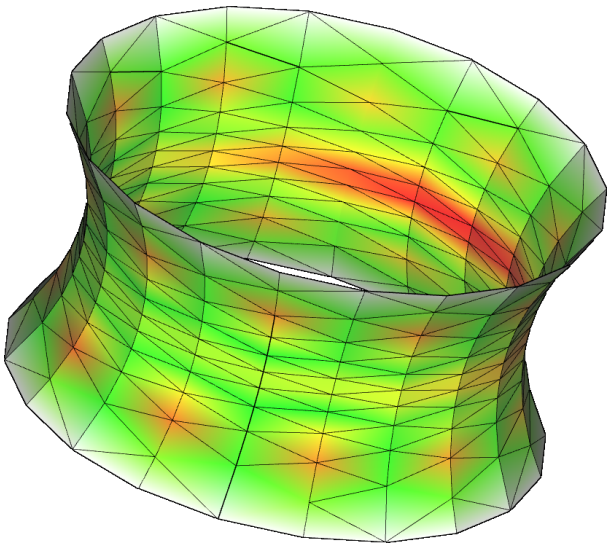
This example is similar to the previous one, with the main difference of perturbation at the initial position. To achieve convergence, larger fairness weights were necessary for the first step but could then be released to a similar value as before. Noticeable is the irregular colorization of target energy in the result, which is due to a relatively small number of only 13 iterations. Although more iteration steps would not influence the mesh in a noticeable way, they would align the variables to the hard constraint equations and lower the target energy as well, resulting in a smoother coloring.



(a) Initial position. ($2.33 \cdot 10^{+1}$ max.)

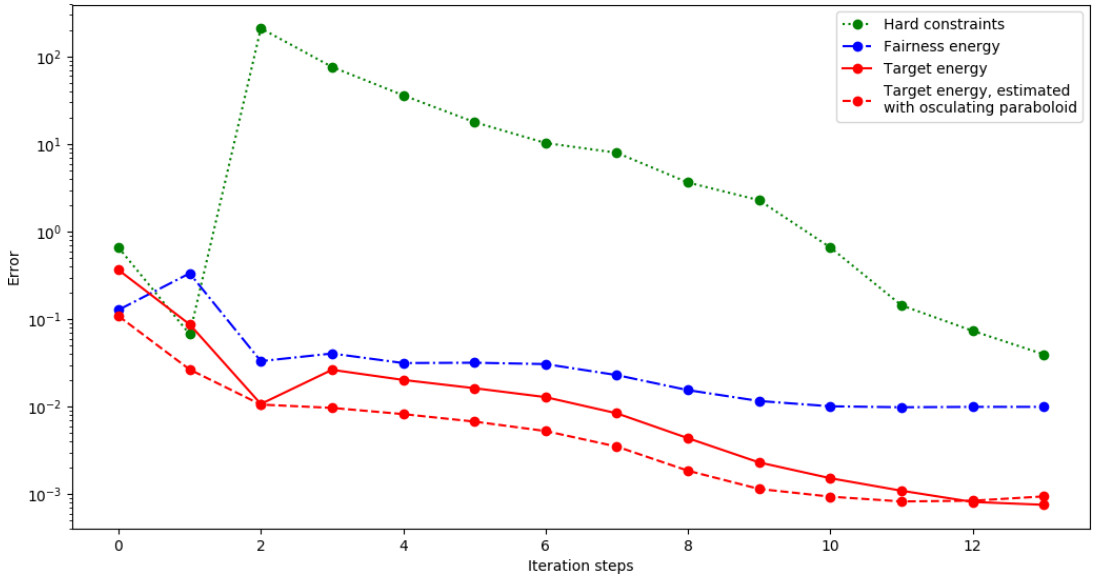
(b) 1 iteration with stronger fairness weights.

(c) 3 iterations with stronger and medium fairness weights.



(d) The resulting mesh after 13 iterations, with small fairness weights after iteration 3. ($9.84 \cdot 10^{-2}$ max.)

Figure 5.4
Minimizing $\kappa_1 + \kappa_2$ on a perturbed bulgy cylinder. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) Colors in figure (b,c) indicate the change of curvature in each vertex.



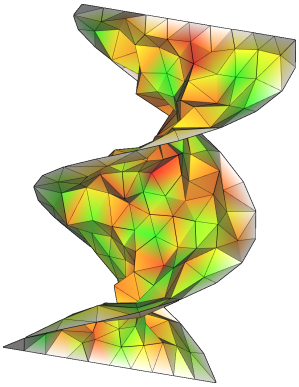
(e) The energy graph is similar to 5.3e. Increment of fairness weights in step 1 are noticeable. In step 2 they are already lowered.

Example figure 5.5 **Surface type**	**Original** perturbed helicoid	**Result** helicoid
Total energy	$1.28 \cdot 10^{0}$	$1.02 \cdot 10^{-2}$
Hard constraints energy	$4.93 \cdot 10^{-1}$	$4.63 \cdot 10^{-1}$
Fairness energy	$2.31 \cdot 10^{-2}$	$8.77 \cdot 10^{-3}$
Target energy	$6.27 \cdot 10^{-1}$	$2.78 \cdot 10^{-4}$
- max. at vertex	$6.59 \cdot 10^{0}$	$1.01 \cdot 10^{-2}$
H estimated with osculating paraboloid	$3.99 \cdot 10^{-2}$	$7.42 \cdot 10^{-5}$
Computation time:		57.6 s
Number of iterations:		14
209 vertices	568 edges	360 facets

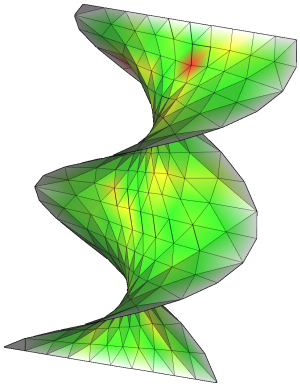
In this example, we start with a quite noisy data. Step 1 already results in a much smoother surface. However, the underlying non-visible variables (all but the vertices) are still messed up, due to the bad initialization. To speed convergence up, we re-initialize the variables, based on the current vertex positions. This results in a fast convergence after only 14 steps.

In step 7 we also lowered the fairness weights, to allow better convergence towards a small target energy.

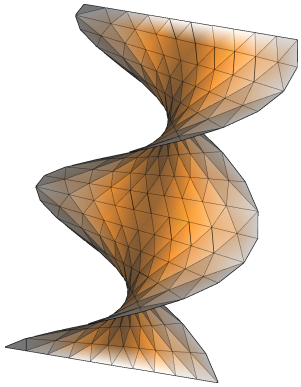
Generally, this target energy method appears to be quite stable. However the fairness terms must be weighted carefully and sometimes these weights have to be changed during the iteration process.



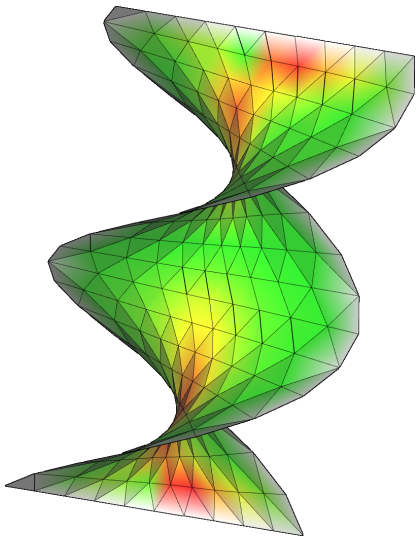
(a) Initial perturbed position. ($6.59 \cdot 10^0$ max.)



(b) After 1 iteration with stronger fairness weights and re-initialization after this step. ($2.94 \cdot 10^0$ max.)

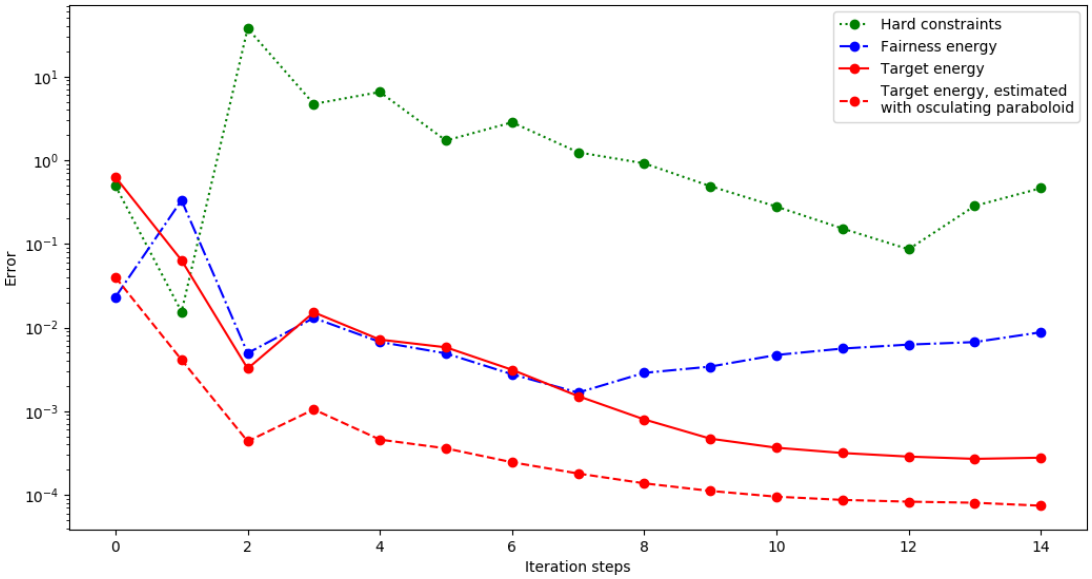


(c) The curvature $\kappa_1 \in [0.5, 1.9]$ at each vertex of the result.



(d) The resulting mesh after 14 iterations, with small fairness weights after iteration 7. ($1.01 \cdot 10^{-2}$ max.)

Figure 5.5
Minimizing $\kappa_1+\kappa_2$ on a perturbed helicoid. Colors in figure (a,b,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) Colors in figure (c) indicate the curvature κ_1 in each vertex (light: small, dark: big). As in all examples, we fix any boundary vertex, on the top, bottom and sides.



(e) At the beginning, strong fairness weights assure a useful direction of optimization. At step 7, the weights are lowered, resulting in smaller target energy without strong restriction about fairness. The resulting mesh, once in a stable position, does not diverge much from this point.

Version trace(W)

It was shown by [8], that the trace of the extended discrete shape operator converges towards the mean curvature measure. Therefore we can aim to minimize this trace, reducing the explicit equations for eigenvalues and -vectors of W. Our target energy is then

$$E_k(\boldsymbol{x}) = (W_k^{1,1} + W_k^{2,2} + W_k^{3,3})A_k$$

We have n equations.

Example figure 5.7 **Surface type**	**Original** cylinder	**Result** catenoid
Total energy	$3.52 \cdot 10^{0}$	$6.09 \cdot 10^{-4}$
Hard constraints energy	$1.33 \cdot 10^{-13}$	$1.25 \cdot 10^{-7}$
Fairness energy	$9.44 \cdot 10^{-4}$	$6.09 \cdot 10^{-4}$
Target energy	$3.52 \cdot 10^{0}$	$2.25 \cdot 10^{-8}$
- max. at vertex	$2.03 \cdot 10^{0}$	$1.5 \cdot 10^{-8}$
H estimated with osculating paraboloid	$1.85 \cdot 10^{-2}$	$1.11 \cdot 10^{-3}$
Computation time:		192.1 s
Number of iterations:		37
220 vertices	620 edges	400 facets

In this example we decided to run more iterations. Figure 5.6 shows the result after only 7 iterations, leading to an almost perfect mesh, but with worse fulfilled hard constraints. This gives the irregular coloring. The graphs in figure 5.7e show the converging results after more than 30 steps. The fairness energy halts at a certain level, because the target energy prevents further shrinking of the mesh, which fairness alone would tend to.

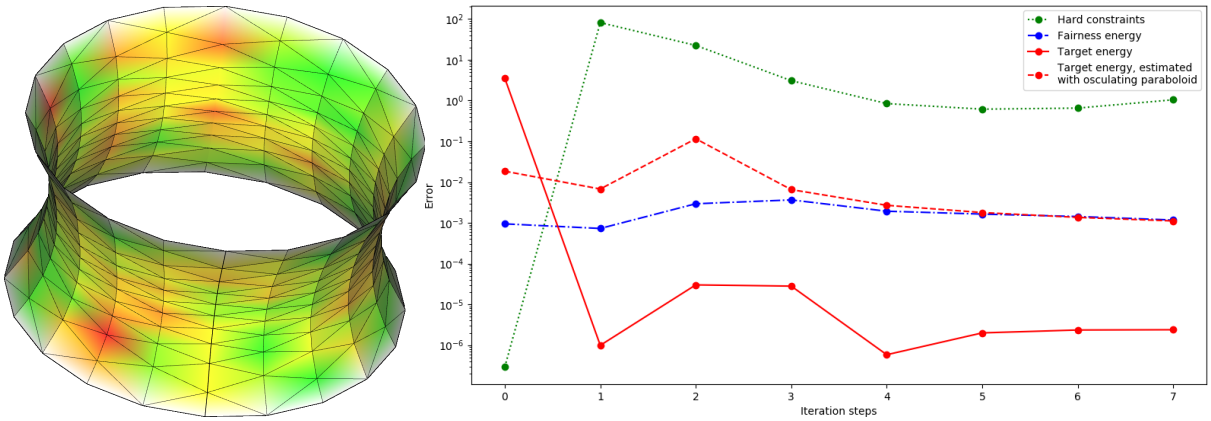
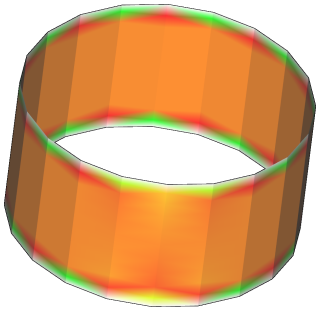
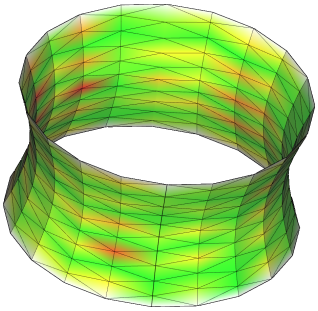


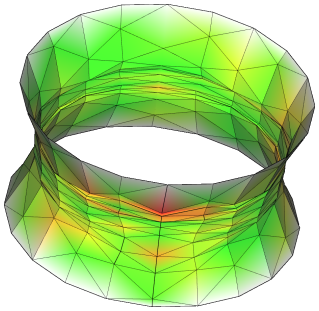
Figure 5.6: Remarks to example figure 5.7. After only 7 iteration steps, the resulting mesh is almost at its convergent final position. The colors indicate the target energy (red: $3.33 \cdot 10^{-6}$ max.). Irregular patterns hypothesize not fulfilled hard constraints at step 7. More iterations lead to a smooth solution (figure 5.7 (d) and (e)).



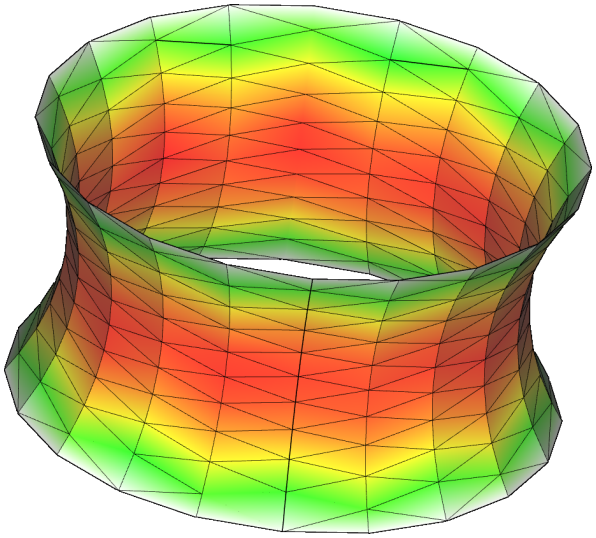
(a) Initial position. ($2.03 \cdot 10^{0}$ max.)



(b) After 1 step. Small fairness and small target energy. ($2.15 \cdot 10^{-6}$ max.)

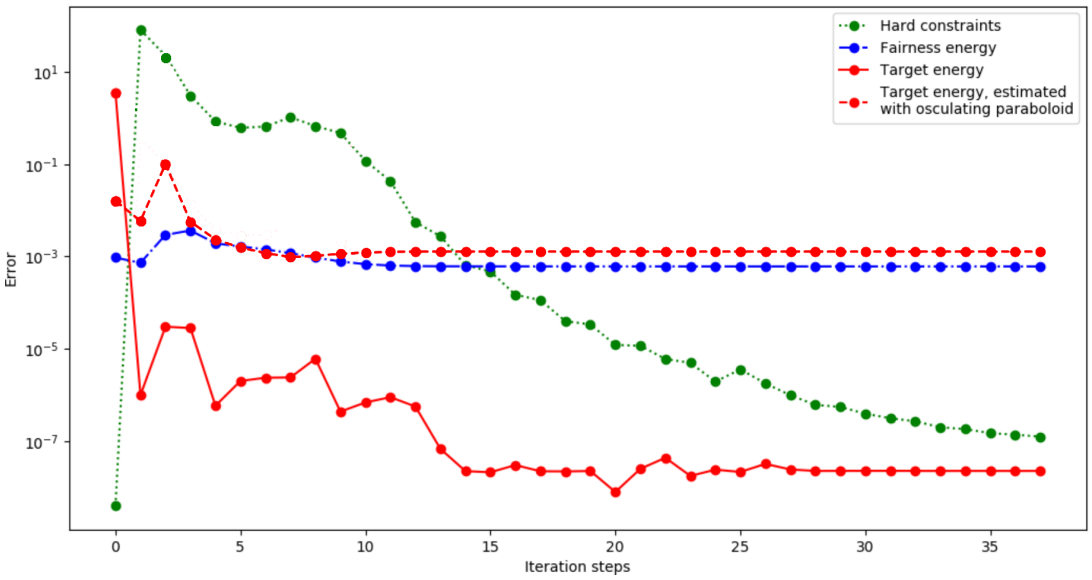


(c) The second iteration compensates the large constraint energy from step 1. ($6.03 \cdot 10^{-6}$ max.)



(d) Result after 37 iteration steps. Target energy is close to zero. ($1.5 \cdot 10^{-8}$ max.)

Figure 5.7
Minimizing trace(W) on a cylinder. Colors in figure (a-d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The final mesh is almost obtained after step 7. More iterations give convergence in constraint and target energy, but do not affect the mesh much (see also figure 5.6).



(e) Small fairness energy weights result in a divergence in the step: Fairness and target energies go down, but are not connected. More iterations give a stable solution.

5.2 Constant mean curvature surfaces

Constant mean curvature surfaces are a generalization of minimal surfaces. Their mean curvature is $H \equiv c$ with some constant scalar c. Numerical application of generating and optimizing CMC surfaces exist in a huge variety. Various different approaches were considered, for example in [28].

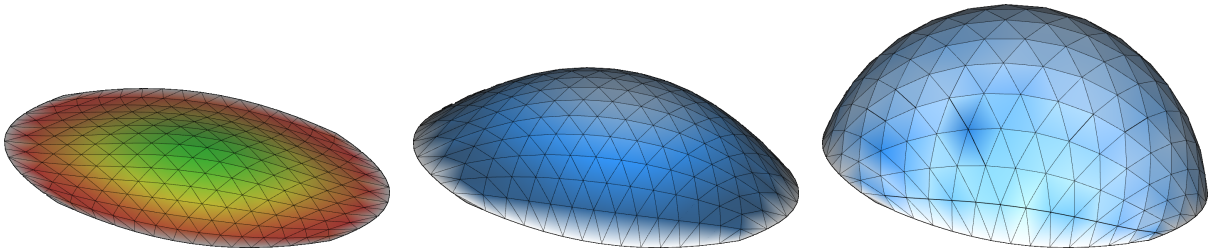
For application examples we will focus on the easiest case of (parts of) spheres. We choose the third (referring to the previous sections) definition of the mean curvature, involving the trace of W. It is a good trade-of between fewer equations while involving the complex definition of W. However, other definitions do not influence the quality of our results. Our energy is

$$E_k(\boldsymbol{x}) = (\text{trace}(W_k) - c)A_k$$

We have again n equations.

Example figure 5.8 **Surface type**	**Original** circular disc	**Result** hemisphere
Total energy	$4.12 \cdot 10^{-1}$	$1.48 \cdot 10^{0}$
Hard constraints energy	$4.80 \cdot 10^{-2}$	$1.47 \cdot 10^{0}$
Fairness energy	$2.32 \cdot 10^{-2}$	$7.38 \cdot 10^{-3}$
Target energy	$3.40 \cdot 10^{-1}$	$2.77 \cdot 10^{-4}$
- max. at vertex	$2.62 \cdot 10^{0}$	$2.35 \cdot 10^{-3}$
H estimated with osculating paraboloid	$1.70 \cdot 10^{-2}$	$3.80 \cdot 10^{-4}$
Computation time:		100.3 s
Number of iterations:		20
225 vertices	616 edges	392 facets

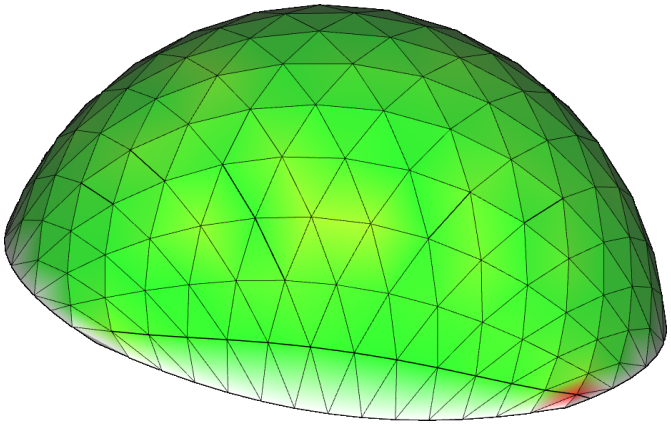
This example demonstrates the effect of negative (figure 5.9) and positive (figure 5.9) mean curvature, leading to a convex or concave surface, depending on the direction of the Gauß map. Here the target energy acts acts as a blow up on the initial surface, while the fairness tries to prohibit this effect. Because of this tension, larger constants lead to more unstable results.



(a) Initial position. ($2.62 \cdot 10^0$ max.)

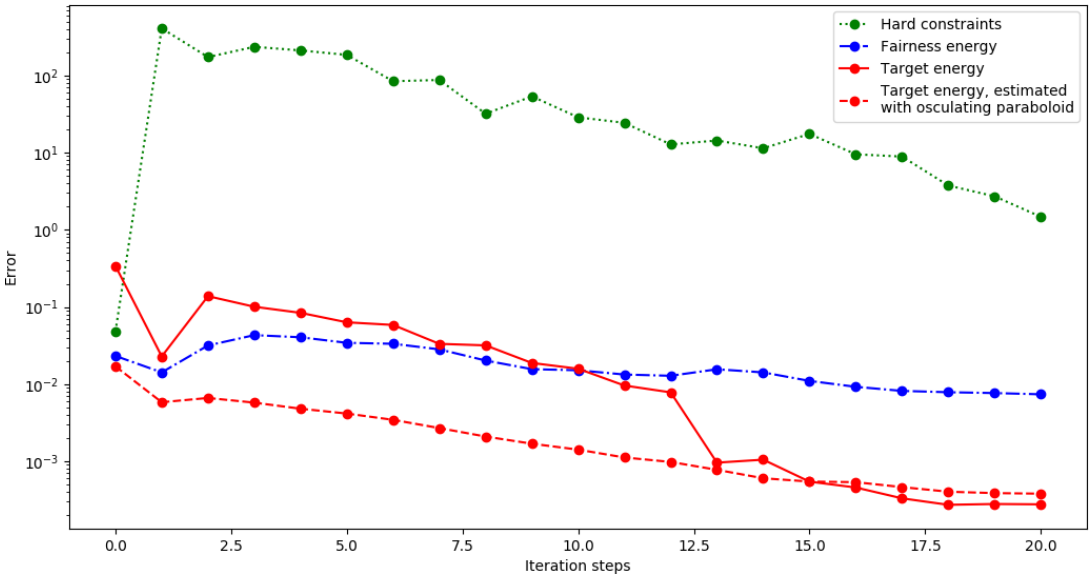
(b) After 1 iteration step.

(c) Resulting mesh with small changes in some vertices to improve hard constraint energy.



(d) The final result after 20 iteration steps. The target energy is close to zero, except for few points near the boundary. ($2.35 \cdot 10^{-3}$ max.)

Figure 5.8
Solving trace$(W) = c$ on a circular disc. Choosing $c = -1$ leads to a hemisphere. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The colors in figure (b,c) indicate the absolute change of curvature in each vertex (bright: small, dark: large).



(e) From iteration 12, the weights for the target energy were raised, leading to a better approximation.

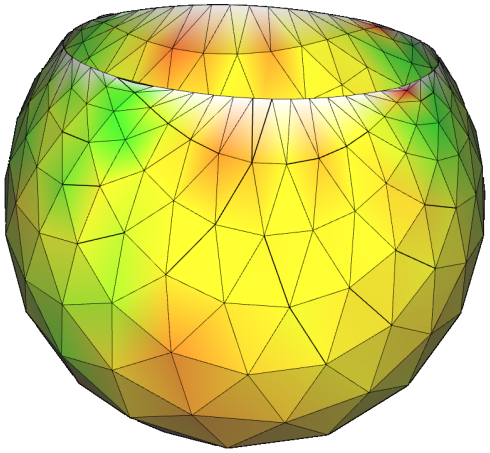


Figure 5.9: Remarks to example figure 5.8. Minimizing trace$(W) = c$ with the same initial position, but with $c > 1$. The positive constant leads to a concave solution (normals points upwards at initial, respectively inwards at resulting mesh) compared to the convex solution for a negative constant. The large constant cannot be fulfilled with the given boundary constraint, therefore the solution is not stable. (picture at iteration 20 with $1.58 \cdot 10^{-1}$ max.)

5.3 Developable surfaces

A brief study of developable surfaces is given in section 5.3. Smooth examples are planes, cylinders, cones and other solids, involving partially differentiable surfaces, gluing other developable surfaces together.

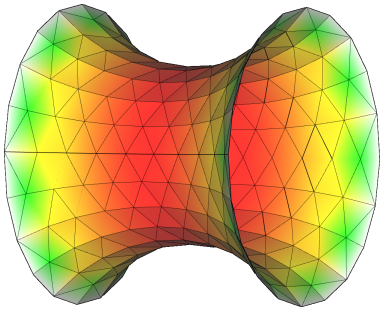
The Gaussian curvature of developable surfaces is constantly 0. We define the energy as

$$E_k(\boldsymbol{x}) = (\kappa_k^1 \kappa_k^2) A_k$$

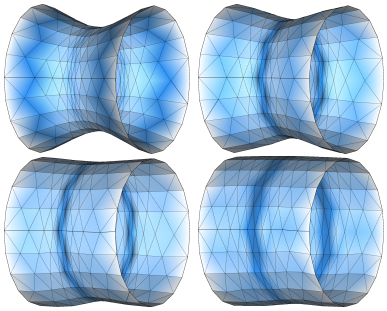
This leads to n target energy equations in our setup.

Example figure 5.10 **Surface type**	**Original** hyperboloid	**Result** cylinder
Total energy	$2.21 \cdot 10^{-1}$	$1.35 \cdot 10^{-3}$
Hard constraints energy	$3.30 \cdot 10^{-3}$	$1.87 \cdot 10^{-6}$
Fairness energy	$4.62 \cdot 10^{-2}$	$1.35 \cdot 10^{-3}$
Target energy	$1.72 \cdot 10^{-1}$	$3.02 \cdot 10^{-6}$
- max. at vertex	$5.30 \cdot 10^{0}$	$7.76 \cdot 10^{-5}$
H estimated with osculating paraboloid	$8.26 \cdot 10^{-2}$	$1.13 \cdot 10^{-6}$
Computation time:		83.2 s
Number of iterations:		16
220 vertices	620 edges	400 facets

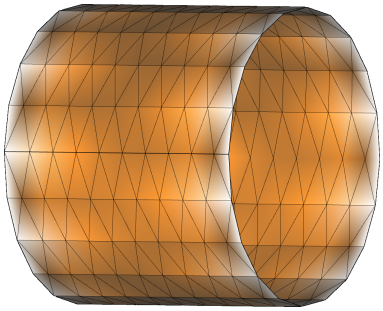
In this example of a cylinder, we can use small fairness weights, leading to a degeneration in the first steps, but are remodeled in the final shape. The osculating paraboloid fits our data better, showing a almost perfect resulting target energy after only 16 steps.



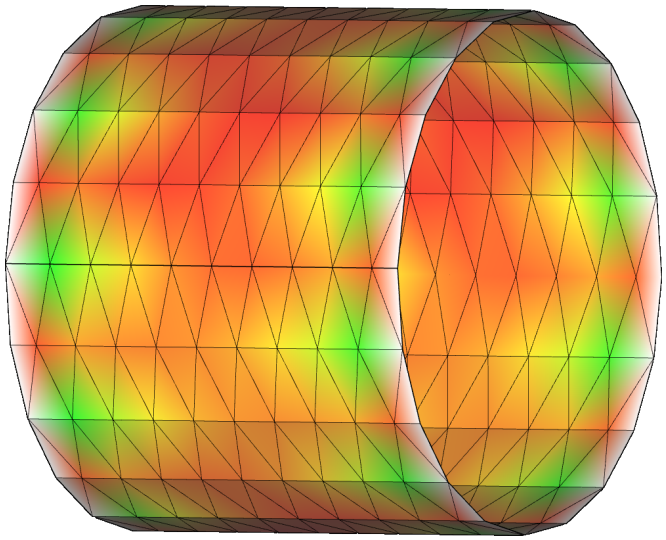
(a) Initial position. ($5.30 \cdot 10^0$ max.)



(b) Changes in the first 4 steps. Fairness is lost at the beginning, but later restored.

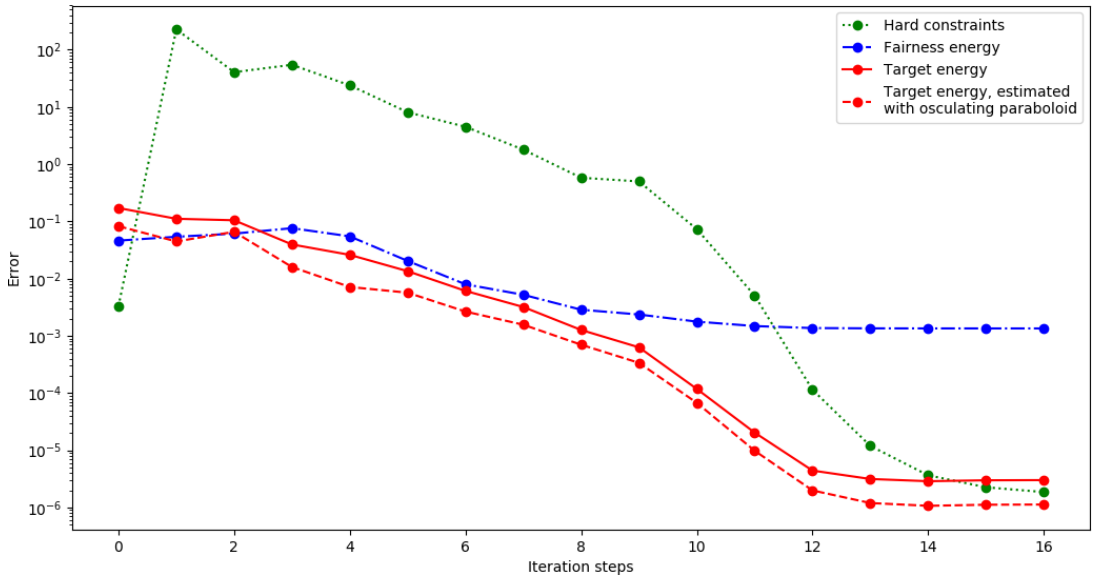


(c) At the result one of the two principal curvatures is close to 0 in each vertex. ($4.21 \cdot 10^{-5}$ max.)



(d) The final result after 16 iteration steps. The target energy is close to 0, changes in color are effectively small. ($7.76 \cdot 10^{-5}$ max.)

Figure 5.10
Minimizing $\kappa_1 \cdot \kappa_2$ on a hyperboloid. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The colors in figure (b) indicate the absolute change of curvature in each vertex and in (c) the curvature κ_1 (bright: small, dark: large).



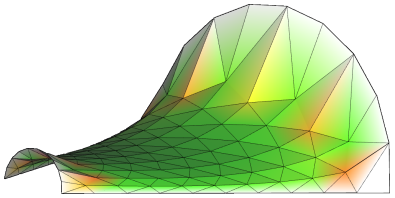
(e) The convergence of the target energy aligns with the estimated target energy from the osculating paraboloid. Small fairness weights allow degeneration in step 1-4 (figure 5.10c), but once the resulting shape is obtained, fairness terms can be fulfilled too.

Example figure 5.11 **Surface type**	**Original** flattened cone sector	**Result** cone sector
Total energy	$1.63 \cdot 10^{-1}$	$1.48 \cdot 10^{-3}$
Hard constraints energy	$9.43 \cdot 10^{-2}$	$6.33 \cdot 10^{-4}$
Fairness energy	$2.46 \cdot 10^{-2}$	$8.31 \cdot 10^{-4}$
Target energy	$4.44 \cdot 10^{-2}$	$1.97 \cdot 10^{-5}$
- max. at vertex	$1.85 \cdot 10^{0}$	$5.88 \cdot 10^{-5}$
H estimated with osculating paraboloid	$2.63 \cdot 10^{-3}$	$7.76 \cdot 10^{-5}$
Computation time:		40.1 s
Number of iterations:		30
121 vertices	320 edges	200 facets

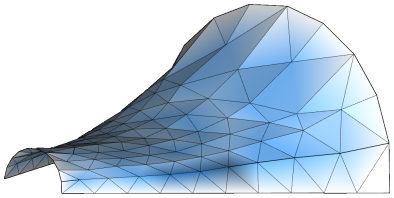
To test another example mesh, we start with a cone, flattened between its boundaries. As visible in figure 5.11a, the mesh already has a very small overall bending. This is important for initialization. Starting with planar parts (which already have zero Gaussian curvature), results in problems during the optimization. The target energy does not allow the mesh to leave its local minimum. This demonstrates the influence of the model in the whole process. Small differences in the initial position can lead to completely different outputs.

Another interesting fact in this example is the increment (!) of target weights by a factor of 10 in step 21 (figure 5.11e). This is clearly visible in the graph, but already one step later, the target energy is smaller again and converges towards a very small value, resulting in better convergence of the hard constraints.

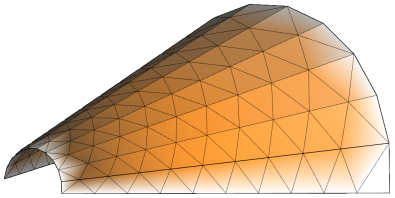
Developable surfaces are interesting to explore. To deal with more representatives of this class of surfaces, some adjustments must be made in the implementation, for example dealing with non smooth parts in the surface to represent non differentiable kinks in the smooth equivalents.



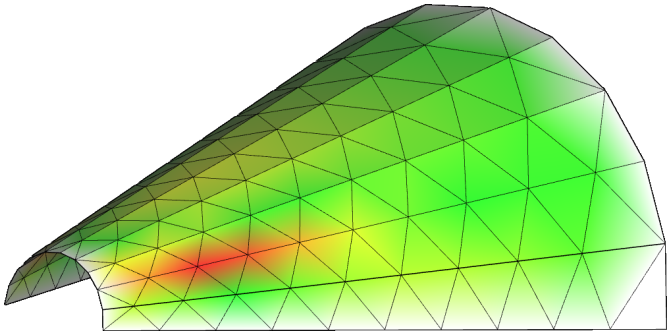
(a) Initial position. ($1.85 \cdot 10^0$ max.)



(b) After step 3.

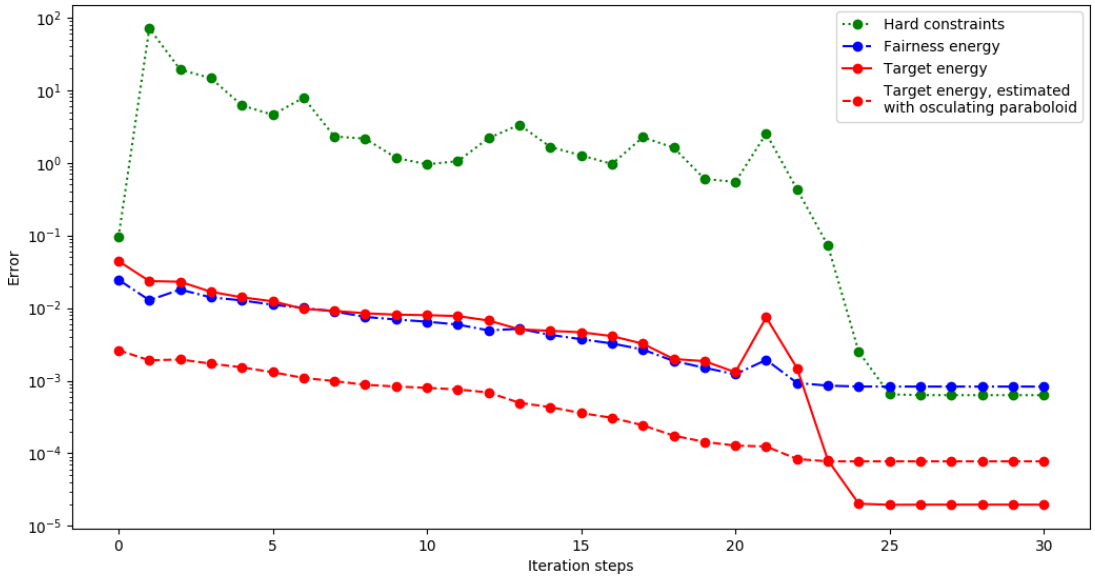


(c) The non-zero principal curvature $\kappa_2 \in [0.65, 2.24]$ of the resulting mesh.



(d) A cone sector is obtained after 30 iteration steps. ($5.88 \cdot 10^{-5}$ max.)

Figure 5.11
Minimizing $\kappa_1 \cdot \kappa_2$ on a deformed flattened cone sector. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The colors in figure (b) indicate the absolute change of curvature in each vertex and in (c) the non-zero curvature κ_2 (bright: small, dark: large).



(e) Starting from step 21 the weights for the target energy are increased by a factor of 10. This results in a better convergence of both, target energy and hard constraints.

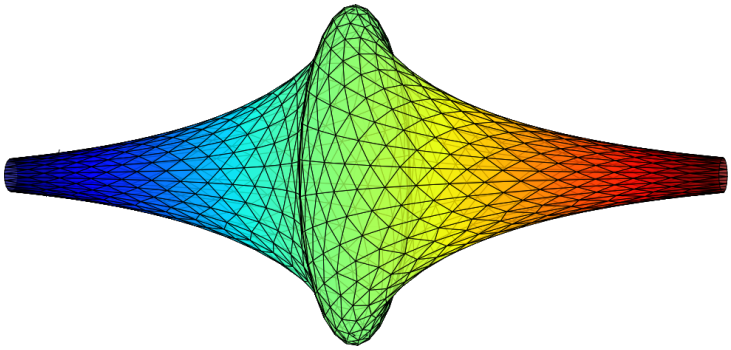


Figure 5.12: A *tractricoid*, generated by revolving a tractrix about its asymptote. This surface is also called a pseudosphere, due to the constant negative Gaussian curvature (in analogy to the constant positive Gaussian curvature of a sphere). It has a singularity at its equator. A parametrization of the tractricoid in the figure with $K = -1$ is $(u, v) \mapsto (\operatorname{sech} u \cos v, \operatorname{sech} u \sin v, u - \tanh u)$, $u \in (-\infty, \infty), v \in [0, 2\pi)$.

5.4 Constant Gaussian curvature surfaces

Similar to constant mean curvature, we generalize developable surfaces to meshes, with *constant Gaussian curvature*. A closed surface with constant positive curvature is a sphere ($K = \frac{1}{R^2}$). An example with singularities, but constant Gaussian curvature $K < 0$ is the so-called pseudosphere (figure 5.12).

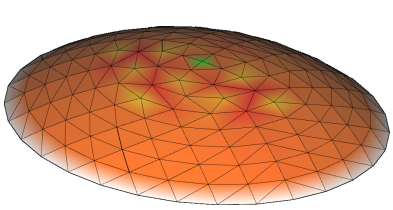
We define our energy

$$E_k(\boldsymbol{x}) = (\kappa_k^1 \kappa_k^2 - c) A_k, \quad c \in \mathbb{R}$$

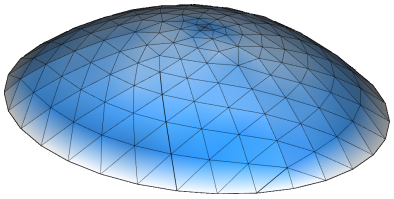
Again, this leads to n target energy equations.

Example figure 5.13 **Surface type**	**Original** sphere cap, $r = 1.5$	**Result** sphere cap, $r = 1$
Total energy	$9.00 \cdot 10^{-2}$	$7.03 \cdot 10^{-2}$
Hard constraints energy	$9.67 \cdot 10^{-4}$	$6.86 \cdot 10^{-2}$
Fairness energy	$6.04 \cdot 10^{-2}$	$1.17 \cdot 10^{-3}$
Target energy	$2.87 \cdot 10^{-2}$	$5.96 \cdot 10^{-4}$
- max. at vertex	$6.86 \cdot 10^{-1}$	$2.25 \cdot 10^{-1}$
H estimated with osculating paraboloid	$7.55 \cdot 10^{-3}$	$8.54 \cdot 10^{-4}$
Computation time:		46.6 s
Number of iterations:		16
159 vertices	444 edges	286 facets

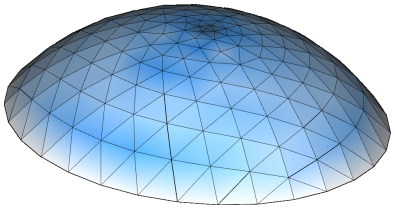
Approximating a constant Gaussian curvature $K \equiv c \neq 0$ is more difficult, due to more distinct solutions. This is due to the fact, that the Gaussian curvature is (in contrast to the mean curvature) an intrinsic measure (the famous *theorema egregium* by C.F. Gauß). For example, the dome mirrored at the xy-plane, visualizing a cup instead of a cap, has exactly the same Gaussian curvature and is therefore another stable solution. The starting position and other constraints have strong influence to the result (if there is any at all).



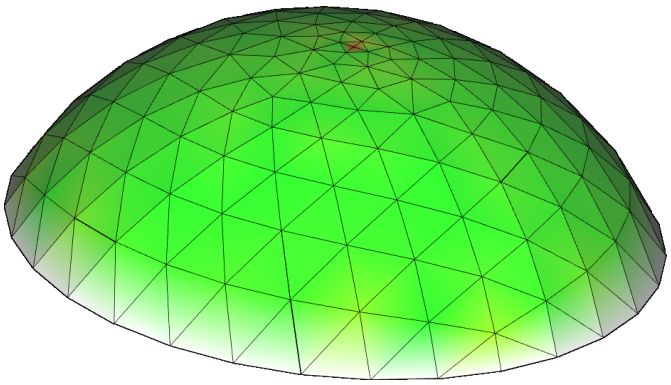
(a) Initial position. ($6.86 \cdot 10^{-1}$ max.)



(b) Step 1. Fairness energy shrinks the triangles on top of the cap.

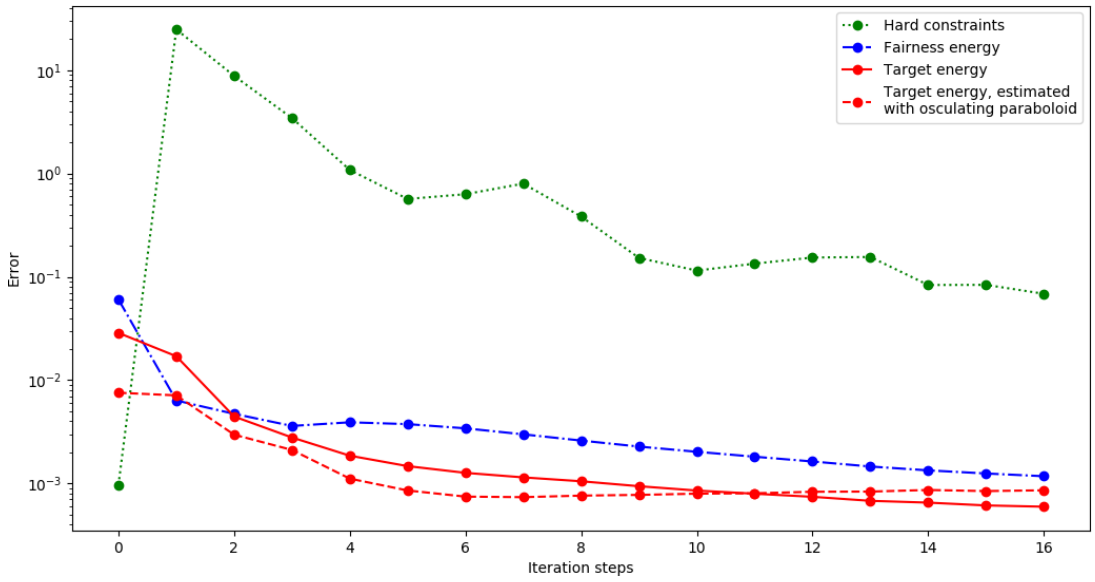


(c) Step 2. The mesh keeps blowing up to increase the Gaussian curvature.



(d) After 16 steps, a part of a sphere with radius 1 is obtained. Problems with the target energy and hard constraints occur at the top of the dome. ($2.25 \cdot 10^{-1}$ max.)

Figure 5.13: Minimizing $\kappa_1 \cdot \kappa_2 = 1$ on a sphere cap (radius of initial sphere $r = 1.5$, $K \equiv 1.44$). Due to the starting position, the mesh converges to the upper part of a sphere with a smaller radius, increasing the Gaussian curvature to $K \equiv 1$. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The colors in figure (b,c) indicate the absolute change of curvature in each vertex (bright: small, dark: large).

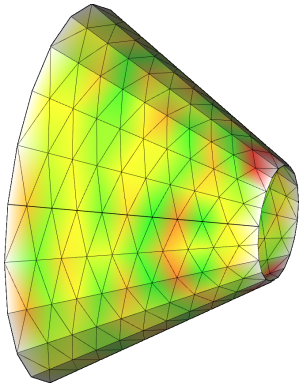


(e) Due to the mesh connectivity and fairness energy, the triangles on top shrink, making it harder for the algorithm to converge in target and hard constraints. However, the result is stable and pictures the exact solution quite well.

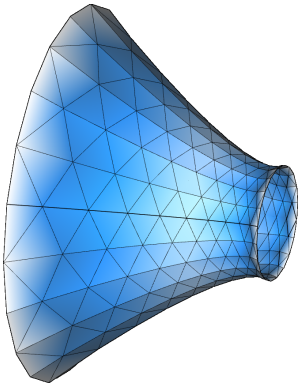
Example figure 5.14	**Original**	**Result**
Surface type	cone	tractricoid
Total energy	$1.25 \cdot 10^{-1}$	$6.83 \cdot 10^{0}$
Hard constraints energy	$9.73 \cdot 10^{-5}$	$6.81 \cdot 10^{0}$
Fairness energy	$5.03 \cdot 10^{-2}$	$4.90 \cdot 10^{-3}$
Target energy	$7.44 \cdot 10^{-2}$	$9.63 \cdot 10^{-3}$
- max. at vertex	$1.00 \cdot 10^{0}$	$7.55 \cdot 10^{-1}$
H estimated with osculating paraboloid	$1.97 \cdot 10^{-2}$	$2.25 \cdot 10^{-3}$
Computation time:		137.7 s
Number of iterations:		16
220 vertices	620 edges	400 facets

As in the previous example, the structure of the mesh is essential for the resulting surface. As visible in the pictures and energy graph, the hard constraints cannot be completely fulfilled at vertices close to the boundary next the singularity. In further application it might be possible to implement a mesh refinement method, to deal with such cases.

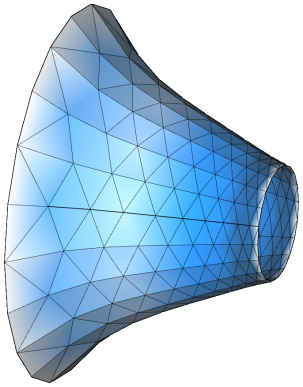
In practice, we explored some other problems with the CGC energy setup. Because of Mindings theorem, a mesh with constant Gaussian curvature is only unique up to isometry. Therefore sometimes more than one solution might be possible. Sometimes single vertices or regions of the mesh try to converge towards one solution, while the rest goes to some other. This leads to a local minimum, which cannot be left easily. For example a stable solution to one of the encountered problems, with one vertex flipped to the other side of the mesh (from convex to concave or vice versa) might be enough already, to be in that case. This problem could be addressed with special weights on single equations, or other fairness term, that treat this behavior. We did not specifically test such ideas in the framework of this thesis.



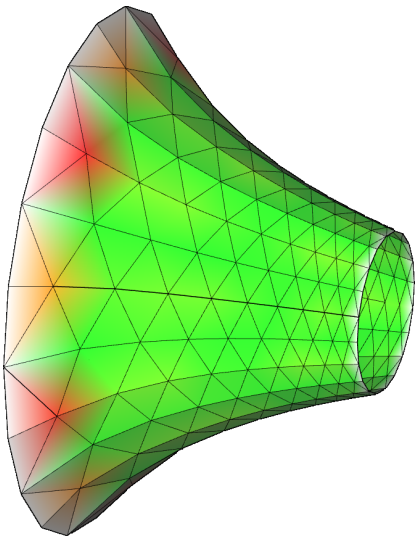
(a) Initial position. ($1.00 \cdot 10^0$ max.)



(b) The mesh after the first iteration step. Fairness energy tightens the neck.

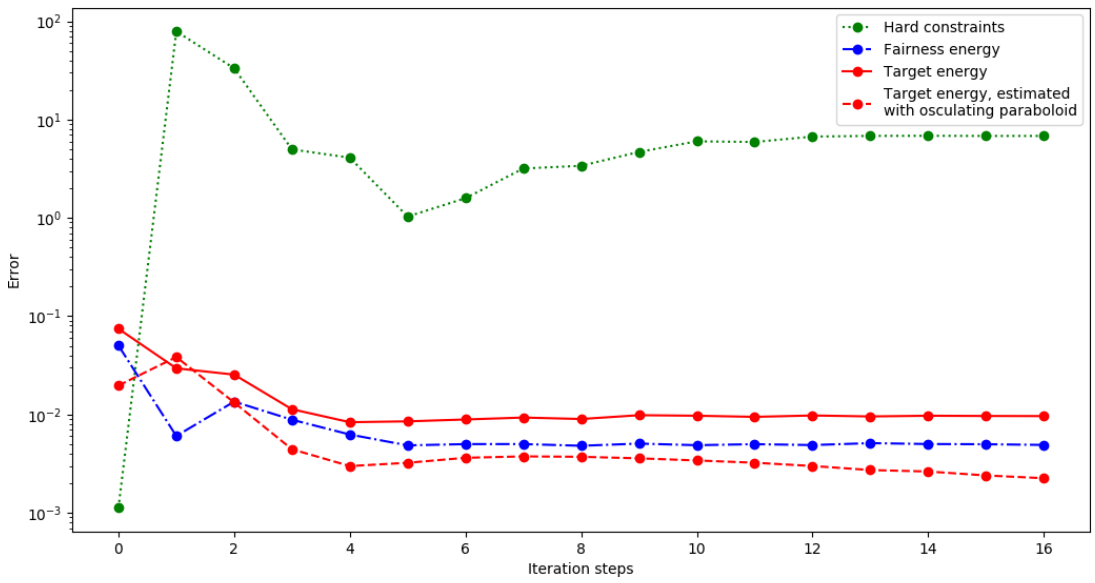


(c) A second iteration step. Target energy operates against this effect.



(d) A part of a tractricoid is obtained after 16 iteration steps. ($7.55 \cdot 10^{-1}$ max.)

Figure 5.14
Minimizing $\kappa_1 \cdot \kappa_2 = -1$ on a cone part. Colors in figure (a,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The colors in figure (b,c) indicate the absolute change of curvature in each vertex (bright: small, dark: large). Close to the singularity of the tractricoid (figure 5.12), the algorithm has difficulties to generate an optimal congruence with the hard constraints.



(e) Convergence close to the final mesh is achieved in relatively few steps. A stable solution, however, requires some more iterations. The hard constraints are not perfectly fulfilled in the end, but the target and estimated target energies are close to zero.

5.5 Willmore surfaces

Consider a compact smooth surface S. The Willmore energy is defined as

$$E(S) = \int_S (H^2 - K)dA = \int_S (\frac{\kappa_1^2}{4} + \frac{\kappa_1\kappa_2}{2} + \frac{\kappa_2^2}{4} - \kappa_1\kappa_2)dA = \frac{1}{4}\int_S (\kappa_1 - \kappa_2)^2 dA$$

This energy is always greater or equal to zero. In the set of closed compact surfaces, only spheres have zero Willmore energy. Therefore this integral measures the distance of a surface to being spherical. The Willmore energy is conformally invariant, making it important in the study of conformal geometry [39]. Many approaches on minimizing this energy in the field of discrete differential geometry were made, for example in [4].

Our target energy is stated with use of our discrete principal curvatures. The integral is approximated with a sum.

$$E(\boldsymbol{x}) = \sum_j (\kappa_j^1 - \kappa_j^2)^2 A_j$$

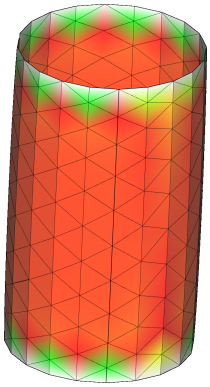
The index j iterates over all vertices of the mesh. This energy is only one single equation in our problem.

Example figure 5.15 **Surface type**	**Original** cylinder	**Result** sphere
Total energy	$6.63 \cdot 10^{-3}$	$6.84 \cdot 10^{-3}$
Hard constraints energy	$1.60 \cdot 10^{-4}$	$5.96 \cdot 10^{-3}$
Fairness energy	$2.78 \cdot 10^{-3}$	$8.75 \cdot 10^{-4}$
Target energy	$3.69 \cdot 10^{-3}$	$3.69 \cdot 10^{-6}$
- max. at vertex	$1.92 \cdot 10^{0}$	$1.13 \cdot 10^{-1}$
H estimated with osculating paraboloid	$3.69 \cdot 10^{-3}$	$3.71 \cdot 10^{-6}$
Computation time:		343.1 s
Number of iterations:		40
220 vertices	620 edges	400 facets

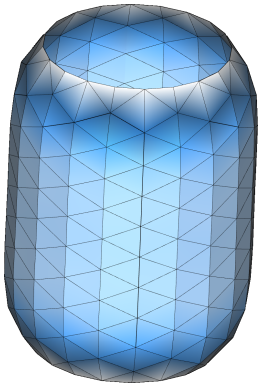
With the use of the *Gauß-Bonet theorem* one can show, that for a compact surface with fixed boundary, a minimizer of the Willmore energy is also a minimizer of *total curvature* $\int_S (\kappa_1^2 + \kappa_2^2)dA$ [4]. We give an example with a discrete version of this energy.

$$E(\boldsymbol{x}) = \sum_j ((\kappa_j^1)^2 + (\kappa_j^2)^2) A_j$$

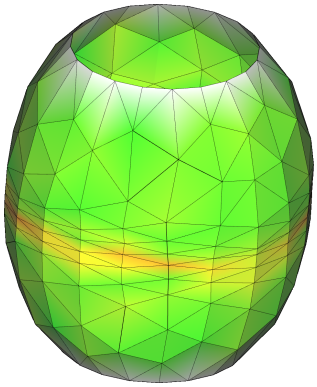
Again, this energy smooths a surface towards a sphere.



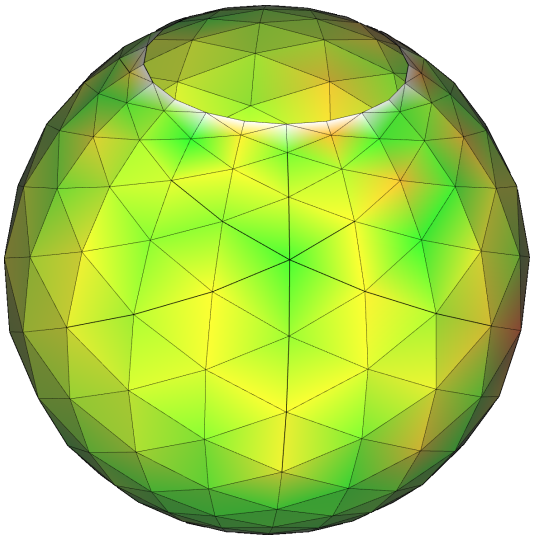
(a) Initial position. ($1.92 \cdot 10^0$ max.))



(b) The mesh after the first iteration step.

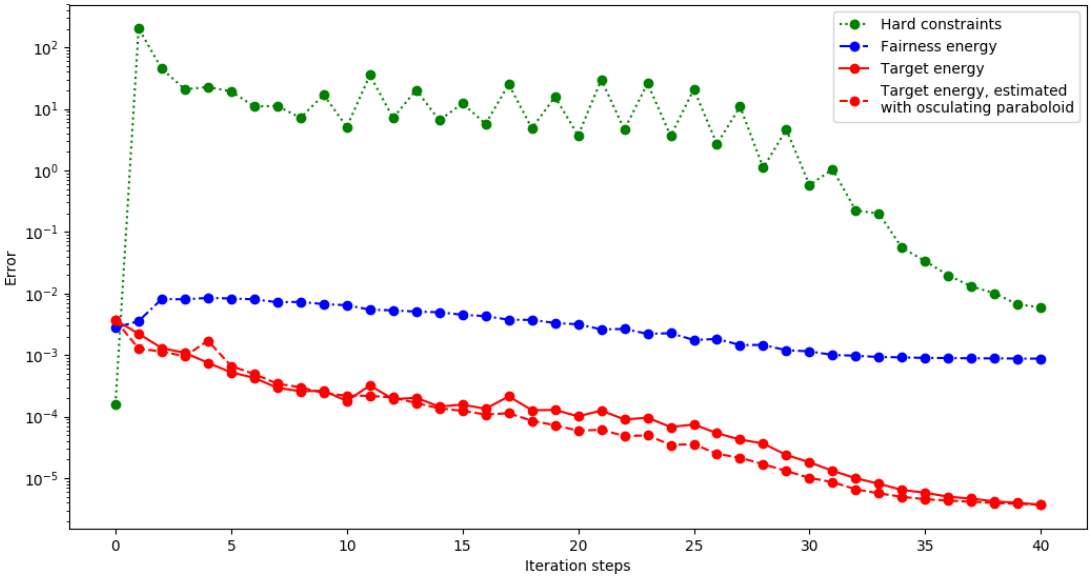


(c) 5 iteration steps. ($2.35 \cdot 10^0$ max.)



(d) The sphere is almost exactly approximated after 40 iteration steps. ($1.13 \cdot 10^{-1}$ max.)

Figure 5.15
Minimizing $\sum(\kappa_1-\kappa_2)^2$ on a slightly bulgy cylinder. Colors in figure (a,c,d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The colors in figure (b) indicate the absolute change of curvature in each vertex (bright: small, dark: large). The surface is converging towards a sphere with zero Willmore energy.



(e) During the first steps, the target energy pulls the mesh towards a sphere, violating the hard constraints. Once a spherical form is obtained, the hards constraints can start converging.

Example figure 5.17 **Surface type**	**Original** kink and buckle	**Result** round kink and planar
Total energy	$1.56 \cdot 10^{-1}$	$1.95 \cdot 10^{0}$
Hard constraints energy	$1.40 \cdot 10^{-1}$	$1.95 \cdot 10^{0}$
Fairness energy	$1.23 \cdot 10^{-2}$	$3.38 \cdot 10^{-3}$
Target energy	$3.06 \cdot 10^{-3}$	$3.91 \cdot 10^{-4}$
- max. at vertex	$6.90 \cdot 10^{+1}$	$1.75 \cdot 10^{+2}$
H estimated with osculating paraboloid	$8.53 \cdot 10^{-3}$	$8.55 \cdot 10^{-4}$
Computation time:		6118 s
Number of iterations:		50
561 vertices	1584 edges	1024 facets

This example demonstrates the behavior of the Willmore energy. The surface tends to converge locally towards a sphere (or a plane, which can be seen as a sphere with $r = \infty$). The boundary prescribes the resulting shape.

For this example we chose relatively small fairness and target weights. One reason, is the definition of the target energy: $\sum \kappa_1^2 + \kappa_2^2$. Only in the special case of a plane, this energy is equal to zero. In other cases it can have a local minimum larger than zero. Big target weights would therefore dictate curvature values, that can never be achieved by a uniform surface, resulting in divergence. On the other hand, the fairness weights must correlate to prevent influence in the shape of the output, but stabilize the result. Together the iteration process is converging slowly but stable.

To verify the quality of the solution, we compare it to the result after 50 iterations without a target energy, but unchanged weights for all other equations. The fairness terms themselves do not influence the resulting shape significantly (figure 5.16).

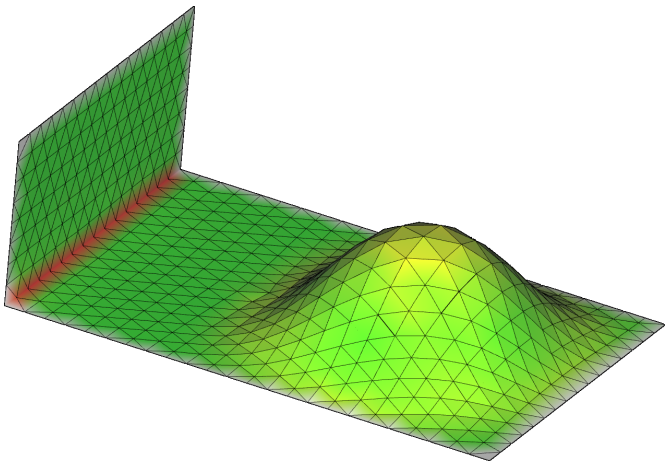
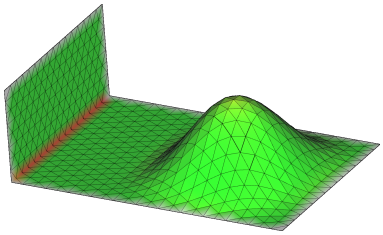
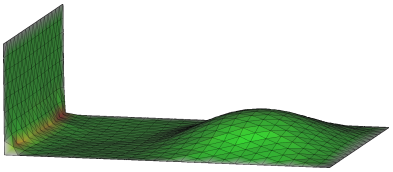


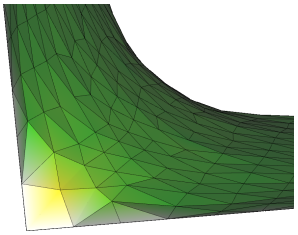
Figure 5.16: With the parameters from example figure 5.17, but without target energy, the algorithm rarely affects the mesh after 50 steps of iteration.



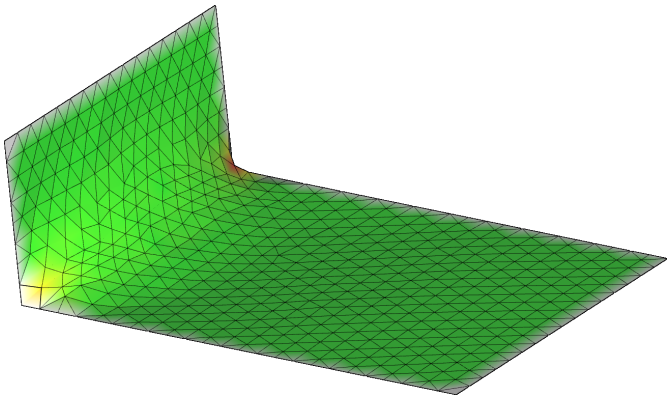
(a) Initial position. ($6.90 \cdot 10^{+1}$ max.)



(b) After 10 iterations, the buckle is already much smaller and some changes in the kink are visible. ($2.68 \cdot 10^{+2}$ max.)

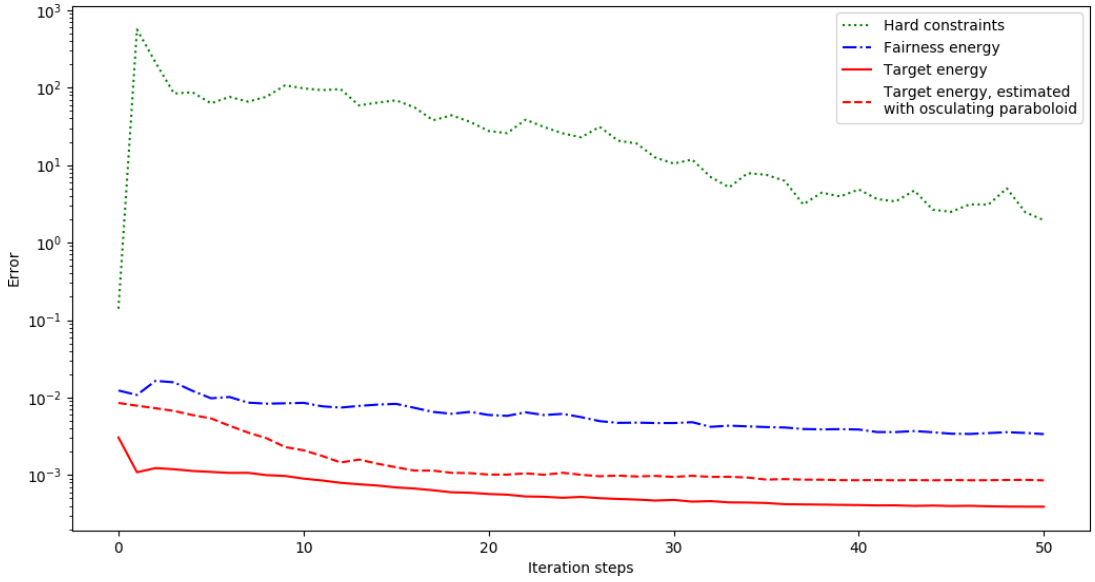


(c) The result after 50 iteration steps shows a smooth surface at the kink apart from the 2 corners at the sides.



(d) After 50 iteration steps, the surface is at is final position. The buckle is completely flattened and the kink round. The only remaining vertices with large target energy are next to the corners, which are stationary. ($1.75 \cdot 10^{+2}$ max.)

Figure 5.17
Minimizing $\sum \kappa_1^2 + \kappa_2^2$ on a kinked surface with a buckle. Colors in figure (a-d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) The behavior is as expected. The buckle is smoothened to a flat piece, while the kink is rounded, although the corners of the boundary make this more complex.



(e) The first iteration step lowers the target energy significantly, at the cost of the hard constraints. During the rest of the process, hard constraints are corrected and fairness improved, resulting in the stationary solution, pictured above.

5.6 Surfaces with minimal total absolute curvature

For a given surface S the *total absolute curvature* energy is defined as

$$\int_S (|\kappa_1| + |\kappa_2|) dA$$

Compared to the Willmore energy and its equivalent, the total energy, it is far less studied. One reason is the difficulty of building derivatives of absolute values. Potential application is possible in building construction and architectural design. In [23] M. Kilian and D. Pellis showed, that total absolute curvature plays an important role in combined form and stress optimization of freeform structures. They used a discrete 2×2 shape operator to derive the principal curvatures and guided projection for optimization.

With our setup, we can easily define our energy functional

$$E_k(\boldsymbol{x}) = \sum_j (|\kappa_j^1| + |\kappa_j^2|) A_j$$

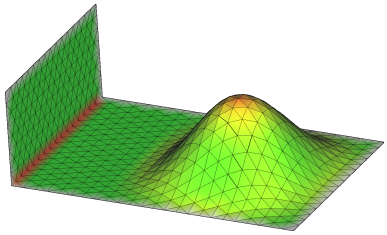
Like in the previous chapter, we have one single target equation. The total absolute curvature behaves also like a smoother, but with different effect on kinks. The following example will demonstrate this.

Example figure 5.18 **Surface type**	**Original** kink and buckle	**Result** round kink and planar
Total energy	$1.53 \cdot 10^{-1}$	$1.31 \cdot 10^{-1}$
Hard constraints energy	$1.40 \cdot 10^{-1}$	$1.30 \cdot 10^{-1}$
Fairness energy	$1.23 \cdot 10^{-2}$	$1.31 \cdot 10^{-3}$
Target energy	$3.53 \cdot 10^{-4}$	$1.16 \cdot 10^{-4}$
- max. at vertex	$1.66 \cdot 10^{+1}$	$1.65 \cdot 10^{+1}$
H estimated with osculating paraboloid	$1.70 \cdot 10^{-3}$	$7.49 \cdot 10^{-4}$
Computation time:		12060 s
Number of iterations:		90
561 vertices	1584 edges	1024 facets

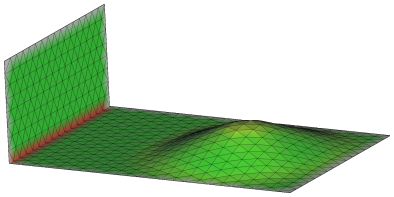
This example demonstrates the behavior of the total absolute curvature energy. The sharp kink is not decreased while the bulge is flattened. We explain this behavior as follows. Of course zero principal curvature at each vertex is optimal for the functional and therefore every bulge is "worse" than a planar surface. However the kink in the boundary does not allow the surface to bend towards a plane overall. For small values, the sum of squares is less than the square of the sum

$$\sum_i |x_i|^2 < (\sum_i |x_i|)^2, \quad x_i < 1$$

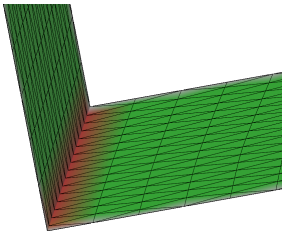
and the Willmore energy "spreads" the overall curvature which was first concentrated in few vertices over adjacent neighbors. The linear total absolute curvature however, does not change when curvature is distributed over different vertices and therefore the kink already has (constant) minimal total absolute curvature.



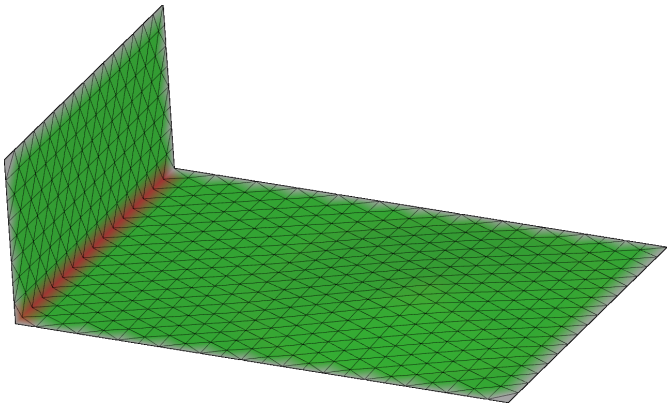
(a) Initial position. ($1.66 \cdot 10^{+1}$ max.)



(b) After 50 iterations with a small target weight, the buckle is smaller, but convergence is slow. ($1.65 \cdot 10^{+1}$ max.)

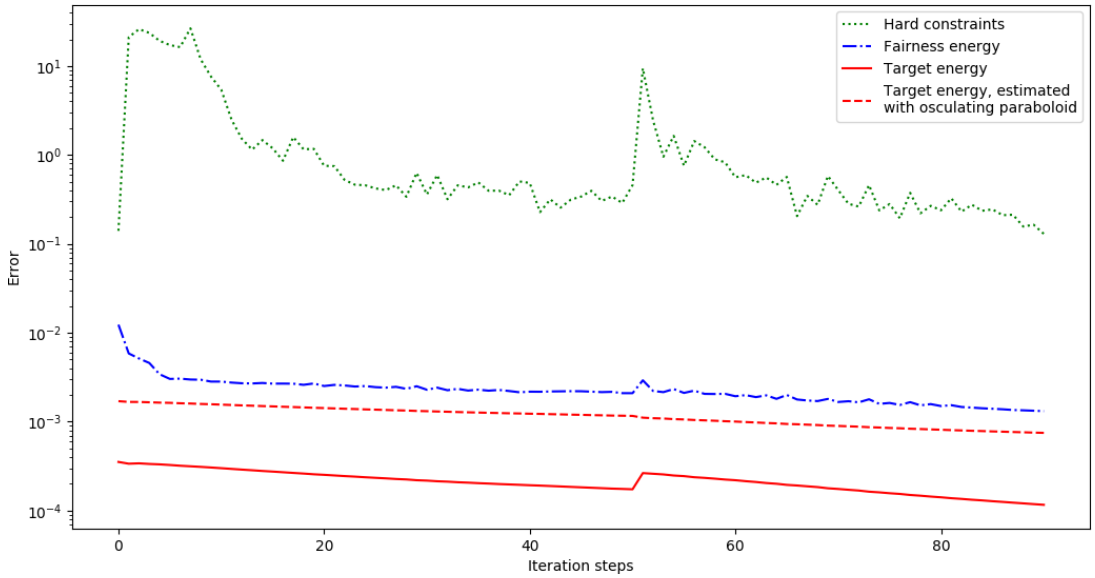


(c) The result after 90 iteration steps shows almost no changes in the kink.



(d) A stationary solution is obtained after 90 iteration steps. The buckle is flat and the kink unchanged (the maximum energy in one vertex is still $1.65 \cdot 10^{+1}$)

Figure 5.18
Minimizing $\sum |\kappa_1| + |\kappa_2|$ on a kinked surface with a buckle. Colors in figure (a-d) indicate the target energy in each vertex (green: low; red: high. Maximum of scale in image caption.) With weights of the same magnitude, the algorithm performs similar to the Willmore energy case, except for the kink. Changes would not decrease the total absolute curvature.



(e) After 50 iteration steps with a relatively low target weight, we increased it and observed faster convergence with stable results after 90 iteration steps.

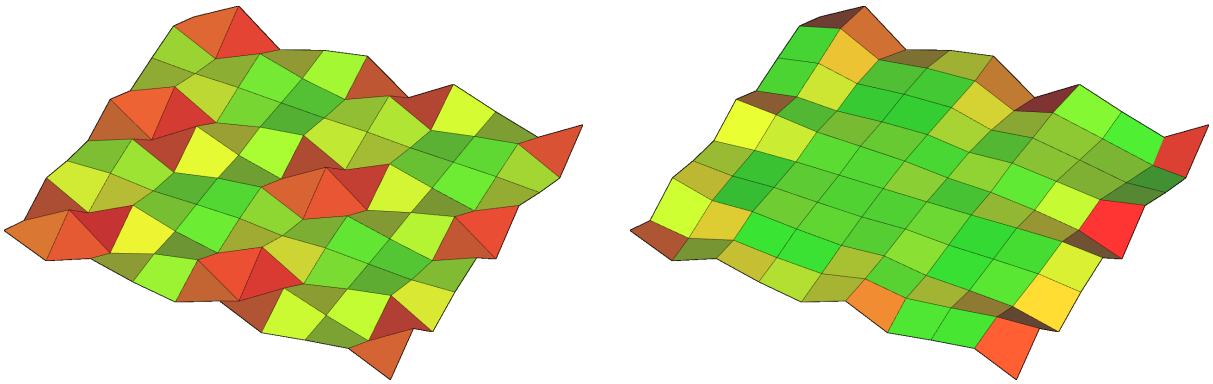


Figure 5.19: A mesh with highly non planar quad facets on the left (color indicates energy at each facet. Red: high, green: low. $1.56 \cdot 10^{-2}$ max.). After 10 iteration steps, the constraints on the facet normals together with fairness terms converged towards a mesh with planar facets, with only relatively large errors next the fixed boundary ($5.36 \cdot 10^{-3}$ max.).

5.7 Quad meshes

In this last chapter we want to an example of our guided projection framework applied on a quad mesh. The hard constraints, presented in chapter 4, are initially fulfilled for triangle meshes and converge towards zero energy for a stable resulting mesh. For an arbitrary mesh with four vertices per facet, the face normal is not well defined. The introduced hard constraints deal with this problem. Orthogonality of the facet normals with each edge of the facet bring the mesh close to a *planar quad mesh*, this is vertices of each facet lying in one plane. Planar facets are used in architectural application, for example in [36].

To demonstrate the effect, we reduce our constraints only to the ones, necessary for the facet normals and add a simply fairing term. The result is shown in figure 5.19.

6 Conclusion

In this thesis we examined basics of discrete differential operators, in particular the extended shape operator for triangular meshes based on normal cycles. Principal curvature and its relatives were built on this fundamentals. The main goal was the finding of a method to optimize meshes under constraints involving these quantities. With use of guided projection, this was achieved and demonstrated on various examples.

Specifically more variables were introduced in addition to the mesh vertices and linked via hard-constraints. During the iteration process, these constraint equations were violated, but as seen in the graphics, later restored, once a stable mesh position was found. Similarly the softer fairness terms converged (after diverging from the initial position in the first steps) towards a fair result. Our main target, the energy functional defining our specific constraint was not perfectly fulfilled at the beginning and defined the primary changes in variables.

In general the first iteration step performed significantly different from the rest. Initially the hard constraints are nearly perfectly fulfilled and the fairness and target energy terms made major changes on the visible mesh. The hard constraints therefore degraded strongly. In the second step, this effect was partially dulled, in many cases resulting in a position even closer to the initial, compared to the result after the first step. More steps led to convergence towards more convergence in all three energies, hard constraints, fairness and target. In some applications, the hard constraints were not possible to be fulfilled steadily, before a stable mesh with low target energy was achieved.

In our examples we studied minimal surfaces with different approaches of target energy. All of them showed good convergence. Methods with fewer equations and variables involved naturally performed faster. Developable surfaces were easily achieved, unless the original mesh was in a position, close to a local but not global minimum. We studied Willmore energy functionals and demonstrated its behavior. Noticeable the equivalent total curvature functional performed worse on complex examples. This was mainly because of an optimal solution with large positive energy and the fact, that the vertex area A_j is constant in each iteration step, misleading the algorithm under some circumstances. The total absolute curvature happened to have similar problems. On the other hand the approach with additional variables for absolute values of curvature did not fail to impress, assumed the initial mesh was useful.

Main problems in the implementation were the time consuming computations for complex problems. Due to many equations and a ill conditioned sparse matrix the linear system took a lot of time to be iteratively solved. For application a more efficient solver is essential. Depending of the size of the problem, a Cholesky factorization attempt might be more promising, since it can use the structure of the sparse matrix, which does not change for individual steps. Only values within this matrix must be adapted. Another important restriction is the influence of weight parameters. We noticed, that even small changes can make the difference between convergence and not. Theory about their magnitude might be helpful but is very difficult due to many distinct equations and dependencies of initial data.

The broad field of application with the presented and not mentioned classes and side conditions makes it generally difficult to find optimal values during iteration and stating explicit rules about the quality of results. Adaptions for individual practices are necessary. The presented method also lacks a theory about the general existence of a convergent solution. We illustrated examples with useful results, but as mentioned, the weights and original mesh have a huge impact and the algorithm may fail to present an expected result.

For further work we highlight the importance of more testing and experimenting with different examples in combination with a more powerful solver to address complex data. Also more than one target energy, and equations respectively, can be joint to achieve a combination of surface classes, resulting in a huge space of possibilities. More methods, using principal curvatures as well as principal directions can be explored and implemented. The setup indicates an easy integration of more equations based on the existing variables.

Special attention can be brought to quadrilateral meshes and polyhedral meshes. Theoretical study of the presented operators in the case of non-triangular meshes are of interest and might give promising results. The implementation itself is not restricted to any category of meshes, but the definitions have to be studied prior to usage.

One adaption of interest would be the implementation and study of more complex boundary restrictions like boundary curves or even reference shapes (compare [36]). Interactive design with real-time calculation can be very interesting, provided the computational time can be sufficiently decreased. Also adaptive meshes which may change connectivity (for example by inserting or deleting vertices) can prove useful to approximate desired surfaces even better.

In summary the presented framework provides many possibilities for further research and additional application.

Bibliography

[1] Nina Amenta and Marshall Bern. “Surface reconstruction by Voronoi filtering”. In: *Discrete Computat. Geom.* 1999, pp. 481–504.

[2] Thomas Banchoff and Stephen Lovett. *Differential geometry of curves and surfaces*. A K Peters, Ltd., 2010.

[3] Alexander I. Bobenko, Helmut Pottmann, and Johannes Wallner. “A curvature theory for discrete surfaces based on mesh parallelity”. In: *Math. Annalen* 348.124 (2010).

[4] Alexander Bobenko and Peter Schröder. “Discrete Willmore flow”. In: *Proceedings of the third Eurographics symposium on Geometry processing*. 101. 2005.

[5] Mario Botsch et al. “OpenMesh - a generic and efficient polygon mesh data structure”. In: *1st OpenSG Symposium*. 2002.

[6] Manfredo P. do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall, Inc., 1976.

[7] Rick Chartrand and Wotao Yin. “Iterively reweighted algorithms for compessive sensing”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008.

[8] David Cohen-Steiner and Jean-Marie Morvan. “Restricted Delaunay Triangulations and Normal Cycle”. In: *Proceedings of the nineteenth annual symposium on computational geometry*, pp. 312–321.

[9] Todias Holck Colding and William P. Minicozzi II. *A Course in Minimal Surfaces*. American Mathematical Society, 2011.

[10] B. Delaunay. “Sur la sphère vide. A la mémoire de Georges Voronoi”. In: *Bulletin de l’Académie des Sciences de l’URSS. Classe des sciences mathématiques et na* (1934), pp. 793–800.

[11] R. Fletcher. “Conjugate gradient methods for indefinite systems”. In: *Numerical Analysis*. Vol. 506. Springer, 1976, pp. 73–98.

[12] Carl Geiger and Christian Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer, 1999.

[13] Carl Geiger and Christian Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, 2002.

[14] B. Guan and J. Spruck. “The existence of hypersurfaces of constant Gauss curvature with prescribed boundary”. In: *J. Differential Geom.* 62 (2002), pp. 259–287.

[15] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[16] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*. Springer, 1994.

[17] B. Harmann. “Curvature Approximation for Triangulated Surfaces”. In: *Geometric Modelling. Computing Supplementum*. Ed. by G. Farin et al. Vol. 8. Springer.

[18] Daniel Herold. “Grundlagen der Sphärischen Geometrie”. Fachbereichsarbeit. Wiener Neustadt: BRG Gröhrmühlgasse, 2012.

[19] Klaus Hildebrandt and Konrad Polthier. “Constraint-based fairing of surface meshes”. In: *Eurographics Symposium on Geometry Processing*. Ed. by Alexander Belyaev and Michael Garland. 2007.

[20] William H. Meeks III and Joaquin Perez. “The classical theory of minimal surfaces”. In: *Bull. Amer. Math. Soc.* (2011).

[21] C. Kanzow. *Numerik linearer Gleichungssysteme: Direkte und iterative Verfahren*. Springer, 2005.

[22] Martin Kilian et al. “Curved Folding”. In: *ACM Trans Graph.* 27.3 (Jan. 2008).

[23] Martin Kilian et al. “Material-minimizing forms and structures”. In: *ACM Trans. Graph. 36* 6 (2017).

[24] Benno Klotzek. *Einführung in die Differentialgeometrie*. Verlag Harri Deutsch, 1997.

[25] Wolfgang Kühnel. *Differentialgeometrie. Kurven - Flächen - Mannigfaltigkeiten*. Vieweg & Sohn Verlag/GWV Fachverlag GmbH, 2005.

[26] Jean-Marie Morvan. *Generalized curvatures*. Springer, 2008.

[27] Christopher C. Paige and Michael A. Saunders. “LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares”. In: *ACM Transactions on Mathematical Software* 8.1 (1982), pp. 43–71.

[28] Hao Pan et al. “Robust Modeling of Constant Mean Curvature Surfaces”. In: *ACM Transactions on Graphics - TOG* 31 (July 2012).

[29] Davide Pellis and Helmut Pottmann. “Aligning principal stress and curvature directions”. In: *Advances in Architectural Geometry*. 2018.

[30] Chi-Han Peng, Helmut Pottmann, and Peter Wonka. “Designing Patterns using Triangle-Quad Hybrid Meshes”. In: *ACM Trans. Graph.* 37.4 (2018).

[31] Ulrich Pinkall and Konrad Polthier. “Computing Discrete Minimal Surfaces and Their Conjugates”. In: *Experimental Mathematics* 2.1 (1993), pp. 15–36.

[32] Heinz Pottmann et al. *EvoluteTools*. http://evolute.at.

[33] H. Rosenberg and J. Spruck. “On the existence of convex hypersurfaces of constant Gauss curvature in hyperbolic space”. In: *J. Differential Geom* 40 (1994), pp. 379–409.

[34] P. T. Sanders and S. W. Zucker. “Inferring surface trace and differential structure from 3d images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.9 (1990), pp. 833–854.

[35] Jacob Steiner. *Jacob Steiner’s gesammelte Werke : 2*. Berlin: Karl Weierstrass, 1882.

[36] Chengcheng Tang et al. “Form-finding with polyhedral meshes made simple”. In: *ACM Transactions on Graphics* 33.70 (2014).

[37] Chengcheng Tang et al. “Interactive design of developable surfaces”. In: *ACM Trans Graph.* 35.2 (2016).

[38] G. Taubin. “Estimating the tensor of curvature of a surface from a polyhedral approximation”. In: *ICCV ’95 Proceeding of the Fifth International Conference on Computer Vision* (1995).

[39] T.J.Willmore. “Surfaces in Conformal Geometry”. In: *Annals of Global Analysis and Geometry 18* (2000), pp. 255–264.

[40] Max Wardetzky et al. “Discrete laplace operators: No free lunch”. In: *Proceedings of the Fifth Eurographics Symposium on Geometry Proceedings*. Eurographics Association. Aire-la-Ville, Switzerland, 2007, pp. 33–37.

[41] Henry C. Wente. “Counterexample to a conjecture of H. Hopf”. In: *Pacific J. Math.* 121 (1986), pp. 193–243.

[42] P. Wintgen. *Normal cycle and integral curvature for polyhedra in Riemmannian manifolds*. Ed. by Gy. Soos and J. Szenthe. 1982.

[43] M. Zähle. *Integral and current representations of Federer’s curvature measure*. Basel: Arch. Math., 1986, pp. 557–567.