TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Rationalization and Design with Cylindrical Strip Models

ausgeführt am Institut für

Diskrete Mathematik und Geometrie

der Technischen Universität Wien

unter der Anleitung von

O.Univ.Prof. Mag.rer.nat. Dr.techn. Helmut POTTMANN

durch

Hannes Limbeck, B.Sc.
Schönbrunnerstrae 179/1/4
1120 Wien

Contents

1 Introduction

1.1 Overview

Dealing with the approximation and design using general cylinders, the motivation for this thesis originates from architecture. While façades tend more and more towards free-form shapes, the latter initially call for a vast amount of purpose-built items and thus increase the costs dramatically.

Rationalization of those façades using ruled and particularly developable surfaces provides a powerful method to overcome this problem. This thesis introduces a complete pipeline for the initialization of a surface covering with cylinder strips – which are a subset of developable surfaces – using contour generators. Based on this initialization and the work [Pot+08], the ideas and methods of the latter are extended to a rationalization technique using cylinder strips.

Similarly, the work [Liu+06] is extended to enable the modeling with cylindrical meshes.

Additionally to the mathematical formulation, details on some algorithms are given. In order to test the proposed methods, they were implemented in C++ using the Evolute SDK allowing to present many of the examples.

In **chapter 2** all basic notions are given for further progressing without the need of breaks to state the foundation. The first section describes differential geometric properties, the second section transfers some of the statements from surfaces to meshes and introduces cylinder strips. In the last section of the chapter the notions and solution methods for selected optimization problems are given.

Chapter 3 treats the rationalization of surfaces in two sections. The first section is the initialization of directrices along which the cylinders shall be initially aligned. The individual methods are stated along with their theory to give a step by step description of the whole process. Afterwards, the second section deals with the actual approximation of a given mesh using cylinder strips by formulating an optimization problem.

Chapter 4 describes the process of generating a strip-wise cylindrical mesh through successive alteration between subdividing and optimizing, by describing the subdivision step and the optimization phase separately and then combining them.

Both, chapter 3 and chapter 4 provide examples and conclusions at the end, showing the possible applications and limitations of the described methods.

Finally, **chapter 5** starts with some details on the implementation, sharing some details on the algorithms and complexity to conclude the thesis.

1.2 Prior Work

In order to initialize a cylindrical strip-model for the rationalization, the contour generator of the mesh is of interest – this is the shadow border on the mesh arising from (parallel) illumination – and the idea originates from the reconstruction of a surface from a sequence of images ([Lau94]). Computing the contour generators from different positions and using their connectivity, the original surface can be recovered approximately ([LBP01]). In doing so, one relies on results of the analysis of apparent contours, i.e. the projected contour generators in the image space, as information about the contour generators like sign of the Gaussian curvature, depth and convexity can be retrieved through the image sequence and viewing motion ([Koe84], [CB90]). An overview of those properties and methods for the parallel case, which is important in the following, as well as perspective case is given in [CG00].

In practice and for this thesis, not only the analysis of smooth surfaces is of interest, therefore discrete methods need to be developed. Extraction of those edges of a mesh, which belong to the contour generator, is fast ([BE99]), but is not very accurate. Describing the contour generator as a piece-wise linear curve through the mesh triangles is more precise and can be accelerated using clustering ([Wan+08]). Other approaches work in the image space or are hybrid methods combining object and image space information ([Ise+03]). These algorithms are also extensively used for non-photorealistic rendering, where feature lines like the silhouette of an object shall be emphasized ([Mar+97]), this also includes technical drawings ([Goo+98]).

The illumination directions and the contour generator tangents are conjugate, which has many important implications for the surface construction along the contour generators – in the parallel case forming cylindrical strips – and the relation between neighboring generators. Conjugate curve networks can be seen as the smooth counterpart of planar quad-meshes, so-called PQ-meshes ([Sau70]), which in turn can be disassembled into developable strips. Hence, discrete conjugate nets are of interest too ([BS08]).

Although not the only geometric objects ([Pot+07]), ruled ([FP10]) and developable strips play an important role in architecture, because they are cost-efficient to produce, since they are generated by moving lines. Surface approximation using developable strips was already treated extensively, especially for the use in architecture in [Pot+08] in form of circular and conical models. Although that work does not rely on conjugate nets arising from parallel illumination, the approximation method used in this thesis is an adaption of the method presented in that paper. The representation of the cylindrical strips is chosen to be B-Spline surfaces, because smooth cylinders can be seen as the limit case of discrete cylinders under subdivision processes ([Wal]) and are a well-studied ([FHK02]). There clearly are other rationalization methods as well, for example in case of surfaces which can be approximated well with only one cylindrical strip and have a suitable shape, it is possible to treat the rationalization problem in image space through correct projection ([Ran98]).

In the design section, the presented method is an adaption of an existing algorithm again. Based on the PQ-perturbation showcased in [Liu+06] for conical meshes, a

quad-mesh is alternately perturbed, i.e. optimized towards a developable strip model with parallel rulings, and subdivided to get a refined strip-wise cylindrical mesh. Similar approaches are undertaken in the area of paper-craft toys, where likewise strips need to be extracted and subsequently optimized ([MGE08], [MS04]).

Additionally, much of the foundation concerning surfaces and their representations is covered by [Car76] and [HL92], those dealing with optimization and general surface approximation by [MNT04], [Kel04], [GK02] and [Pot].

2 Preliminaries

There are some notational conventions used in this thesis. If not stated differently, scalars are denoted with lower case Greek letters (e.g. $\lambda \in \mathbb{R}$) or, if used as parameters, with lower case Latin letters (e.g. $t \in \mathbb{R}$), elements of a vector space are denoted with lower case Latin letters in the form $\overrightarrow{d} \in \mathbb{R}^n$ and matrices with upper case Latin letters (e.g. $\mathrm{A} \in \mathbb{R}^{m \times n}$). To identify maps, they are usually written with their parameters, for example $c(t)$, $\mathbf{x}(u, v)$.

2.1 Differential Geometry

2.1.1 Curves and Surfaces

DEFINITION 2.1 *Let $I = (a, b) \subset \mathbb{R}$ be an open interval. A map $c : I \to \mathbb{R}^n : t \mapsto c(t)$ is called a **differentiable parametrized curve** if $c(t)$ is differentiable. For a point $c(t_0) = (c_1(t_0), c_2(t_0), \ldots, c_n(t_0))$ of the curve at time t_0, $c'(t_0) = (c_1'(t_0), c_2'(t_0), \ldots, c_n'(t_0))$ denotes the derivative with respect to the parameter t, called the **tangent vector** of $c(t)$ at t_0.*

In the following, a curve $c(t)$ will be a differentiable map $c : I \to \mathbb{R}^3$. A point $c(t_0)$ is called a **regular point** if $c'(t_0) \neq 0$ and a **singular point** if $c'(t_0) = 0$. Further, a straight line $\ell \subset \mathbb{R}^3$ is said to be **tangent** to the curve at a regular point $c(t_0)$ if $c(t_0) \in \ell$ and $c'(t_0)$ is parallel to ℓ.

Regular curves – curves consisting of regular points only – can be re-parametrized. A useful parameter is the **arc length** $s(t)$ of the curve, which is given by

$$s(t) := \int_a^t ||\dot{c}(u)|| \mathrm{du}$$

and yields $\mathrm{d}s = ||\dot{c}|| \mathrm{d}t$.

Let $c : I \to \mathbb{R}^3$ be a curve parametrized by arc length, then $\kappa(s) := ||c''(s)||$ is called the **curvature** of the curve at s. Using an arbitrary parameter t, the curvature is given by $\kappa(t) := \frac{\|c''(t) \wedge c'(t)\|}{\|c'(t)\|^3}$. The concept of curvature can be developed for surfaces too, which is done subsequently.

DEFINITION 2.2 *A set $S \subset \mathbb{R}^3$ is called a **regular surface**, if for each $p \in S$ there exists a neighborhood $V \subset \mathbb{R}^3$ of p and a subset $U \subset \mathbb{R}^2$ so that there is a map $\mathbf{x} : U \to V \cap S : (u, v) \mapsto (x(u, v), y(u, v), z(u, v))$ with the following properties:*

1. *$\mathbf{x}$ is differentiable: $x(u, v)$, $y(u, v)$, $z(u, v)$ have continuous partial derivatives of all orders in U.*

2. $\mathbf{x}$ *is a homeomorphism, which means the continuous map* $\mathbf{x}$ *also has a continuous inverse* $\mathbf{x}^{-1} : V \cap S \to U$ *which is the restriction of a continuous map* $\mathbf{X} : W \to \mathbf{R}^2$, $W \supset V \cap S$.

3. *For each* $q \in U$, *the differential* $\mathrm{d}\mathbf{x}_q : \mathbb{R}^2 \to \mathbb{R}^3$ *is one-to-one, that is* $\frac{\partial \mathbf{x}}{\partial u}$ *and* $\frac{\partial \mathbf{x}}{\partial v}$ *are linearly independent for each* q.

The map $\mathbf{x}$ in definition 2.2 yields local coordinates for the surface S in the neighborhood $V \cap S$, thus it is called a local coordinate chart (or parametrization).

Similar to curves, a **parametrized surface** is a differentiable map $\mathbf{x} : U \subset \mathbb{R}^2 \to \mathbb{R}^3$ and points $q \in U$ where $\mathrm{d}\mathbf{x}_q$ is one-to-one are **regular points**, those where this does not hold are called **singular points**.

A vector $\overrightarrow{v} \in \mathbb{R}^3$ is a **tangent vector** to a surface S at a point $p \in S$ if there is a regular curve $c : (-\varepsilon, \varepsilon) \to S$ with $c(0) = p$ and $c'(0) = \overrightarrow{v}$. The set of all tangent vectors at the regular point p is denoted T_pS and with $p = \mathbf{x}(q)$ for $q \in U$ holds

$$\mathrm{d}\mathbf{x}_q(\mathbb{R}^2) = T_pS \ .$$

The set of tangent vectors T_pS at a point p is a 2-dimensional vector space with basis $\{\mathbf{x}_u := \frac{\partial \mathbf{x}}{\partial u}, \mathbf{x}_v := \frac{\partial \mathbf{x}}{\partial v}\}$ associated with $\mathbf{x}$.

2.1.2 First Fundamental Form

Given $S \subset \mathbb{R}^3$, for every regular point $p \in S$ the vector space $T_pS \subset \mathbb{R}^3$ can also be seen as a subset of the Euclidean Space $\mathbb{E}^3$, that is $\mathbb{R}^3$ endowed with the Euclidean inner product

$$< (x_1, y_1, z_1), (x_2, y_2, z_2) >= x_1x_2 + y_1y_2 + z_1z_2 \ .$$

Hence, one can restrict the inner product of the ambient space $\mathbb{E}^3$ to T_pS.

Let $p \in S$ be a regular point of a surface $S \subset \mathbb{R}^3$ and $< ., . >$ the Euclidean inner product. Then the map

$$< ., . >_p : T_pS \times T_pS \to \mathbb{R} : (\overrightarrow{v_1}, \overrightarrow{v_2}) \ \mapsto < \overrightarrow{v_1}, \overrightarrow{v_2} >_p := < \overrightarrow{v_1}, \overrightarrow{v_2} > \qquad (2.1)$$

is a symmetric bilinear form on T_pS, namely the restriction of the Euclidean inner product to T_pS.

DEFINITION 2.3 *With the notation from* (2.1), *the quadratic form*

$$\mathcal{I}_p : T_pS \to \mathbb{R}^+ : \overrightarrow{v} \ \mapsto < \overrightarrow{v}, \overrightarrow{v} >_p \qquad (2.2)$$

is called ***first fundamental form*** *of the regular surface* S *at* $p \in S$.

PROPOSITION 2.4 *Let* $c(t) = \mathbf{x}(u(t), v(t)) \subset S$ *be a regular curve on a regular surface* S, *parametrized via a map* $\mathbf{x}(u, v)$, *then there exist functions* $E(u, v)$, $F(u, v)$, $G(u, v)$ *so that for* $p = c(t_0) = \mathbf{x}(u_0, v_0)$ *holds*

$$\mathcal{I}_p(c'(t_0)) = E(u_0, v_0)(u'(t_0))^2 + 2F(u_0, v_0)u'(t_0)v'(t_0) + G(u_0, v_0)(v'(t_0))^2 \ .$$

Proof Since S is regular, one can write $T_pS = \text{span}\{\mathbf{x}_u, \mathbf{x}_v\}$ and $c'(t_0) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(u(t_0), v(t_0)) = \mathbf{x}_u u' + \mathbf{x}_v v'$. To shorten the notation, $\mathbf{x}_u := \mathbf{x}_u(u_0, v_0)$, other maps respectively. Hence

$$\begin{aligned}
\mathcal{I}_p(c'(t_0)) &= <\mathbf{x}_u u' + \mathbf{x}_v v' \ , \ \mathbf{x}_u u' + \mathbf{x}_v v' >_p \\
&= <\mathbf{x}_u u' \ , \ \mathbf{x}_u u' + \mathbf{x}_v v' >_p + <\mathbf{x}_v v' \ , \ \mathbf{x}_u u' + \mathbf{x}_v v' >_p \\
&= <\mathbf{x}_u u' \ , \ \mathbf{x}_u u' >_p + 2 <\mathbf{x}_u u' \ , \ \mathbf{x}_v v' >_p + <\mathbf{x}_v v' \ , \ \mathbf{x}_v v' >_p \\
&= <\mathbf{x}_u, \mathbf{x}_u >_p (u')^2 + 2 <\mathbf{x}_u, \mathbf{x}_v >_p u'v' + <\mathbf{x}_v, \mathbf{x}_v >_p (v')^2
\end{aligned}$$

and defining

$$\begin{aligned}
E(u,v) &:= <\mathbf{x}_u, \mathbf{x}_u >_p \\
F(u,v) &:= <\mathbf{x}_u, \mathbf{x}_v >_p \\
G(u,v) &:= <\mathbf{x}_v, \mathbf{x}_v >_p
\end{aligned} \tag{2.3}$$

for $\mathbf{x}(u,v) = p$ yields the desired functions. ■
The inner product of two elements $<\overrightarrow{v_1}, \overrightarrow{v_2}>_p$ will also be denoted $\overrightarrow{v_1} \cdot \overrightarrow{v_2}$, suppressing the point p.

2.1.3 Gauss Map

In order to define the second fundamental form, which plays an important role in the next chapter, one has to examine the Gauss map and its differential.

For a surface $S \subset \mathbb{R}^3$, a **vector field** is a map $V : S \to \mathbb{R}^3$. Clearly, as this is a very general concept, not all vector fields are of interest. Assume $S \subset \mathbb{R}^3$ is a regular surface. A map $V : S \to \mathbb{R}^3$ is called **tangential vector field** if $V(p) \in T_pS$ for every $p \in S$ and **normal vector field** if $V(p) \cdot \overrightarrow{v_p} = 0$ for every $p \in S$ and $\overrightarrow{v_p} \in T_pS$. In the following, they are assumed to be normalized.

Every normalized vector has length 1, hence one can think of it as lying on the unit sphere, so $\overrightarrow{v} \in \mathbb{R}^3$ with $\|\overrightarrow{v}\| = 1$ can be seen as $\overrightarrow{v} \in \mathcal{S}^2 := \{(x,y,z) \in \mathbb{R}^3 | x^2 + y^2 + z^2 = 1\}$.

DEFINITION 2.5 *Given a parametrization $\mathbf{x}(u,v)$ in a neighborhood U of $p \in S$, the map*

$$\mathrm{N} : \mathbf{x}(U) \to \mathcal{S}^2 : q \mapsto \frac{\mathbf{x}_u \wedge \mathbf{x}_v}{\|\mathbf{x}_u \wedge \mathbf{x}_v\|}(q) \ , \tag{2.4}$$

*is a normalized normal vector field on $\mathbf{x}(U)$. If it can be extended to a differentiable unit normal vector field on the whole surface S it is called the **Gauss map** of S.*

The extendability of (2.4) means, that locally there always exists a parametrization $\mathbf{x}(u,v)$ so that $<\mathbf{x}_u \wedge \mathbf{x}_v, \mathrm{N}>$ has the same sign for every point $p \in S$. In this case, the surface is called *orientable*.

PROPOSITION 2.6 *The Gauss map is differentiable. Its differential, also called* ***Weingarten map*** *or* ***Shape operator****, in a point $p \in S$ is given by*

$$\mathrm{dN}_p : T_pS \to T_{N(p)}\mathcal{S}^2 : \overrightarrow{v} \mapsto \mathrm{dN}_p(\overrightarrow{v}) \tag{2.5}$$

and is a self-adjoint linear map.

Proof It is obvious, that the Gauss map is differentiable and has a linear differential of the form (2.5). The property that it is self-adjoint, which means $< \mathrm{dN}_p(\overrightarrow{v}), \overrightarrow{w} >= < \overrightarrow{v}, \mathrm{dN}_p(\overrightarrow{w}) >$, is developed in the subsequent paragraphs. ■

Given a parametrized surface $S = \mathbf{x}(u, v)$ and a point $p \in S$, one can express every element of T_pS as a linear combination of its basis $\{\mathbf{x}_u, \mathbf{x}_v\}$. Along a curve on the surface through the point p, the normal field can be written as $\mathrm{N}(u(t), v(t))$. Then, as dN_p can also be seen as a map to T_pS, there are two possibilities to denote $\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u(t), v(t))$, whereupon the second is interesting because it allows an expression in terms of the tangential plane;

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u,v) &= \mathrm{N}_u u' + \mathrm{N}_v v' \\ &= \zeta \mathbf{x}_u + \eta \mathbf{x}_v \quad , \ \zeta, \eta \in \mathbb{R} \ . \end{aligned}$$

Consider the inner product of $\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u, v)$ with the basis elements

$$\begin{aligned} < \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u,v), \mathbf{x}_u > &= < \mathrm{N}_u u' + \mathrm{N}_v v', \mathbf{x}_u > \\ &= < \mathrm{N}_u, \mathbf{x}_u > u' + < \mathrm{N}_v, \mathbf{x}_u > v' \ , \\ < \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u,v), \mathbf{x}_u > &= < \zeta \mathbf{x}_u + \eta \mathbf{x}_v, \mathbf{x}_u > \\ &= \zeta < \mathbf{x}_u, \mathbf{x}_u > + \eta < \mathbf{x}_v, \mathbf{x}_u > \\ &\overset{(2.3)}{=} \zeta E + \eta F \end{aligned}$$

and

$$\begin{aligned} < \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u,v), \mathbf{x}_v > &= < \mathrm{N}_u u' + \mathrm{N}_v v', \mathbf{x}_v > \\ &= < \mathrm{N}_u, \mathbf{x}_v > u' + < \mathrm{N}_v, \mathbf{x}_v > v' \ , \\ < \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{N}(u,v), \mathbf{x}_v > &= < \zeta \mathbf{x}_u + \eta \mathbf{x}_v, \mathbf{x}_v > \\ &= \zeta < \mathbf{x}_u, \mathbf{x}_v > + \eta < \mathbf{x}_v, \mathbf{x}_v > \\ &\overset{(2.3)}{=} \zeta F + \eta G \end{aligned}$$

which yields

$$\begin{pmatrix} < \mathrm{N}_u, \mathbf{x}_u > & < \mathrm{N}_v, \mathbf{x}_u > \\ < \mathrm{N}_u, \mathbf{x}_v > & < \mathrm{N}_v, \mathbf{x}_v > \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} \zeta \\ \eta \end{pmatrix} \ .$$

Because of $< \mathrm{N}, \mathbf{x}_u >= 0$, deriving with respect to u leads to $< \mathrm{N}_u, \mathbf{x}_u > + < \mathrm{N}, \mathbf{x}_{uu} >= 0$ and thus $< \mathrm{N}_u, \mathbf{x}_u > = - < \mathrm{N}, \mathbf{x}_{uu} >$. Using the same argument for the parameter v, the above equation becomes

$$-\begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1} \begin{pmatrix} < \mathrm{N}, \mathbf{x}_{uu} > & < \mathrm{N}, \mathbf{x}_{uv} > \\ < \mathrm{N}, \mathbf{x}_{vu} > & < \mathrm{N}, \mathbf{x}_{vv} > \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \zeta \\ \eta \end{pmatrix}$$

or using

$$\begin{aligned} L &:= < \mathrm{N}, \mathbf{x}_{uu} > , \\ M &:= < \mathrm{N}, \mathbf{x}_{uv} >=< \mathrm{N}, \mathbf{x}_{vu} > , \\ N &:= < \mathrm{N}, \mathbf{x}_{vv} > \end{aligned} \tag{2.6}$$

one has

$$-\underbrace{\begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1}}_{\mathrm{I}^{-1}} \underbrace{\begin{pmatrix} L & M \\ M & N \end{pmatrix}}_{\mathrm{II}} \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \zeta \\ \eta \end{pmatrix} . \tag{2.7}$$

Equation (2.7) shows how to compute the unknown coefficients ζ, η of the derivative of the normal vector field N along a curve on the surface with the knowledge of the second partial derivatives of the parametrization and the velocity vector (u', v') of the curve.

Note that I^{-1} always exists in these cases because $\mathbf{x}_u = 0$, $\mathbf{x}_v = 0$ or $< \mathbf{x}_u, \mathbf{x}_u > \cdot < \mathbf{x}_v, \mathbf{x}_v > = < \mathbf{x}_u, \mathbf{x}_v >^2$ all contradict the regularity.

COROLLARY 2.7 *The differential of the Gauss map can be computed as*

$$\mathrm{dN}_p(\overrightarrow{v}) = -\mathrm{I}^{-1}\mathrm{II} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} ,$$

where $\overrightarrow{v} = \alpha \mathbf{x}_u + \beta \mathbf{x}_v \in T_p S$.

Taking tangent elements $\overrightarrow{v}_1 = \zeta \mathbf{x}_u + \eta \mathbf{x}_v \in T_p S$ and $\overrightarrow{v}_2 = \hat{\zeta} \mathbf{x}_u + \hat{\eta} \mathbf{x}_v \in T_p S$, one can define the bilinear map

$$II(\overrightarrow{v}_1, \overrightarrow{v}_2) = (\zeta, \eta)\mathrm{II}(\hat{\zeta}, \hat{\eta})^T \tag{2.8}$$

which fulfills $II(\overrightarrow{v}_1, \overrightarrow{v}_2) = II(\overrightarrow{v}_2, \overrightarrow{v}_1)$ because the matrix II is symmetric and further

$$\begin{aligned} II(\mathbf{x}_u, \mathbf{x}_u) &= (1,0)\mathrm{II}(1,0)^T &&= L , \\ II(\mathbf{x}_u, \mathbf{x}_v) &= (1,0)\mathrm{II}(0,1)^T &&= M = II(\mathbf{x}_v, \mathbf{x}_u) , \\ II(\mathbf{x}_v, \mathbf{x}_v) &= (0,1)\mathrm{II}(0,1)^T &&= N . \end{aligned}$$

2.1.4 Second Fundamental Form

DEFINITION 2.8 *The bilinear form*

$$\mathcal{II}_p : T_pS \times T_pS \to \mathbb{R} : (\overrightarrow{v}, \overrightarrow{w}) \mapsto - < \mathrm{dN}_p(\overrightarrow{v}), \overrightarrow{w} > \tag{2.9}$$

is called ***second fundamental form****.*

PROPOSITION 2.9 *Let $c(t) = (v(t), w(t))$ be a curve on S through the point $p = c(0) \in S$ with $c'(0) = \overrightarrow{v} \in T_pS$, $\overrightarrow{w} \in T_pS$ arbitrary. Then the second fundamental form* (2.9) *at p can be computed by* (2.8) *via*

$$\mathcal{II}_p(\overrightarrow{v}, \overrightarrow{w}) = II(\overrightarrow{v}, \overrightarrow{w}) \ .$$

Proof First consider the coefficients L, M, N. As $< \mathrm{N}, \mathbf{x}_u >= 0$, differentiating with respect to u yields $< \mathrm{N}_u, \mathbf{x}_u > + < \mathrm{N}, \mathbf{x}_{uu} >= 0$ and thus $< \mathrm{N}, \mathbf{x}_{uu} = - < \mathrm{N}_u, \mathbf{x}_u >$. Applying this argument to $< \mathrm{N}, \mathbf{x}_v >= 0$ too and differentiating also with respect to v then helps expressing L, M, N as

$$\begin{aligned} L &= - < \mathrm{N}_u, \mathbf{x}_u > , \\ M &= - < \mathrm{N}_u, \mathbf{x}_v >= - < \mathrm{N}_v, \mathbf{x}_u > , \\ N &= - < \mathrm{N}_v, \mathbf{x}_v > \ . \end{aligned} \tag{2.10}$$

Given $\mathcal{II}_p(\overrightarrow{v}, \overrightarrow{w}) = - < \mathrm{dN}(\overrightarrow{v}), \overrightarrow{w} >$, $\mathrm{dN}(\overrightarrow{v})$ can be written as $\mathrm{dN}(\overrightarrow{v}) = \mathrm{N}_u u' + \mathrm{N}_v v'$ and $\overrightarrow{w}$ as $\overrightarrow{w} = \alpha \mathbf{x}_u + \beta \mathbf{x}_v$. Hence

$$\begin{aligned} - < \mathrm{dN}(\overrightarrow{v}), \overrightarrow{w} > &= - < \mathrm{N}_u u' + \mathrm{N}_v v', \alpha \mathbf{x}_u + \beta \mathbf{x}_v > \\ &= (u', v') \begin{pmatrix} - < \mathrm{N}_u, \mathbf{x}_u > & - < \mathrm{N}_u, \mathbf{x}_v > \\ - < \mathrm{N}_v, \mathbf{x}_u > & - < \mathrm{N}_v, \mathbf{x}_v > \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \\ &= (u', v') \begin{pmatrix} L & M \\ M & N \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \\ &= II(\overrightarrow{v}, \overrightarrow{w}) \ . \end{aligned}$$

■

With all this preparatory work, one can define one of the most important properties for this thesis, which is used for the initialization in the next chapter.

DEFINITION 2.10 *Assume $\overrightarrow{v}, \overrightarrow{w} \in T_pS$, $\overrightarrow{v} \neq 0$, $\overrightarrow{w} \neq 0$, then $\overrightarrow{v}$ and $\overrightarrow{w}$ are said to be* ***conjugate*** *if $\mathcal{II}(\overrightarrow{v}, \overrightarrow{w}) = 0$.*

2.1.5 Curvature

The concept of curvature can now be extended to surfaces.

DEFINITION 2.11 *For $p = \mathbf{x}(u, v) \in S$ and $\overrightarrow{v} \in T_pS$, the* ***normal curvature*** *of S at p in the direction $\overrightarrow{v}$ is defined by*

$$k_n = \frac{\mathcal{II}_p(\overrightarrow{v}, \overrightarrow{v})}{\mathcal{I}_p(\overrightarrow{v})} = - \frac{< \mathrm{dN}_p(\overrightarrow{v}), \overrightarrow{v} >}{||\overrightarrow{v}||^2} \ .$$

THEOREM 2.12 [Car76] *Let $\mathbb{V}$ be a 2-dimensional linear vector space and $A : \mathbb{V} \to \mathbb{V}$ a self-adjoint linear map. Then there exists an orthonormal basis $\{\overrightarrow{e}_1, \overrightarrow{e}_2\}$ of $\mathbb{V}$ such that $A(\overrightarrow{e}_1) = \lambda_1 \overrightarrow{e}_1$, $A(\overrightarrow{e}_2) = \lambda_2 \overrightarrow{e}_2$ (that is, $\overrightarrow{e}_1$ and $\overrightarrow{e}_2$ are eigenvectors, and λ_1, λ_2 are eigenvalues of A). In the basis $\{\overrightarrow{e}_1, \overrightarrow{e}_2\}$, the matrix A is clearly diagonal and the elements λ_1, λ_2, $\lambda_1 \geq \lambda_2$, on the diagonal are the maximum and minimum, respectively, of the quadratic form $Q(\overrightarrow{v}) =< A\overrightarrow{v}, \overrightarrow{v} >$ on the unit circle of $\mathbb{V}$.*

So, for all $p \in S$ exists an orthonormal basis $\{e_1, e_2\}$ of $T_p\mathcal{S}^1$ such that

$$\begin{aligned} \mathrm{dN}_p(e_1) &= -\kappa_1 e_1 \\ \mathrm{dN}_p(e_2) &= -\kappa_2 e_2 \end{aligned}$$

where κ_1, κ_2 are the minimum and maximum of the second fundamental form on the unit circle of $T_p\mathcal{S}^1$, respectively, and thus the extreme values of k_n at p, following the last theorem 2.12.

DEFINITION 2.13 *The maximum κ_1 and the minimum κ_2 of the normal curvature k_n are called **principal curvatures** at $p \in S$. The corresponding directions $\overrightarrow{e}_1$, $\overrightarrow{e}_2$ are the **principal directions**.*

Using theorem 2.12, on sees also sees that $\det \mathrm{dN} = \kappa_1\kappa_2$ and $\mathrm{tr}\ \mathrm{dN} = \kappa_1 + \kappa_2$.

DEFINITION 2.14 *The value $K = \det \mathrm{dN}_p$ is called **Gaussian curvature** in $p \in S$ and the value $H = -\frac{1}{2}\mathrm{tr}\ \mathrm{dN}_p$ is the **Mean curvature** in $p \in S$.*

The Gaussian curvature is crucial for the following tasks, while the Mean curvature will not play an important role. Thus, despite the definition above, it is not considered anymore. The Gaussian curvature can be computed by

$$\begin{aligned} K &= \det \mathrm{dN}_p = \left(\det \begin{pmatrix} E & F \\ F & G \end{pmatrix}\right)^{-1} \det \begin{pmatrix} -L & -M \\ -M & -N \end{pmatrix} \\ &= \frac{LN - M^2}{EG - F^2}\ . \end{aligned}$$

The Gaussian curvature is used to classify the points of a surface. A point $p \in S$ is called **elliptic** if $K > 0$, **parabolic** if $K = 0$ and in case $K < 0$ it is called **hyperbolic**. Additionally, if p is elliptic and $\kappa_1 = \kappa_2$ then it is also called an **umbilical** point and if p is parabolic with $\mathrm{dN}_p = 0$ it is said to be planar. If a connected (subset of a) surface consists of umbilical points only, it is either planar or contained in a sphere. A special case of conjugacy occurs for $\overrightarrow{v} \in T_pS \backslash \{\overrightarrow{0}\}$ with $\mathcal{II}(\overrightarrow{v}, \overrightarrow{v}) = 0$. These elements are called self-conjugate or **asymptotic directions** and $k_n = 0$ holds at p in direction $\overrightarrow{v}$.

2.2 Cylindrical Strip Models

With help of the notations of the previous section, one can now define cylindrical strip models.

2.2.1 Ruled and Developable Surfaces

DEFINITION 2.15 *A surface S is called a **ruled surface** if there is a parameterization*

$$\mathbf{x}(u,v) = c(u) + vd(u) \tag{2.11}$$

of the surface, where $c(u)$ is an arbitrary curve, called directrix, $d(u)$ defines a direction for every u, called generating direction, and the straight lines $L_u(v) = \mathbf{x}(u,v)$ are called rulings.

As set earlier, $c(u)$ as a curve is smooth. The direction $d(u)$ in the definition above is required to be at least continuously changing in order to generate a useful surface, nevertheless a ruled surface is allowed to have singular points per definition. For every ruled surface defined via (2.11), there exists a parametrization $\tilde{\mathbf{x}}(u,v) = c(u) + v\tilde{d}(u)$ with $||\tilde{d}(u)|| = 1$ which generates the same surface. So, without loss of generality, $d(u)$ is always considered to be normalized.

Often of interest are developable surfaces, i.e. surfaces that can be mapped isometrically into the plane, which means they can be bend and folded without stretching or shearing to a planar surface.

DEFINITION 2.16 *A surface S is called **developable surface** if for every point $p \in S$ holds $K = 0$.*

In $\mathbb{R}^3$ developable surfaces are ruled surfaces, so equivalently a surface with parametrization (2.11) is developable if $\det(c'(u), d(u), d'(u)) = 0$. To see this, look at

$$\begin{aligned} \mathbf{x}_{uu} &= c''(u) + vd''(u), \\ \mathbf{x}_{uv} &= d'(u), \\ \mathbf{x}_{vv} &= 0 \end{aligned}$$

which leads to $K = 0$ exactly if $M =< \mathrm{N}, \mathbf{x}_{uv} >= 0$.

With $(||d(u)|| = 1 \Rightarrow d(u) \wedge d'(u) = 0)$, follows $\mathbf{x}_u \wedge \mathbf{x}_v = (c'(u) + vd'(u)) \wedge d(u) = c'(u) \wedge d(u)$, hence $< \mathbf{x}_u \wedge \mathbf{x}_v, d'(u) >=< c'(u) \wedge d(u), d'(u) >= \det(c'(u), d(u), d'(u))$ and finally

$$K = 0 \Leftrightarrow \det(c'(u), d(u), d'(u)) = 0 \ .$$

DEFINITION 2.17 *A surface S is called a **cylinder** or **cylindrical surface** if there is a parametrization*

$$\mathbf{x}(u,v) = c(u) + v\overrightarrow{d} \tag{2.12}$$

of the surface where $c(u)$ is an arbitrary curve, called directrix, and $\overrightarrow{d}$ is an arbitrary direction vector, called generating direction.

The case of a ruled surface (2.11) with $d(u)$ not constant is also called *non-cylindrical surface* and the special case where $c(u)$ is a circle and $\overrightarrow{d}$ is normal to the plane in which the circle lies, is called right circular cylinder and complies with the surface

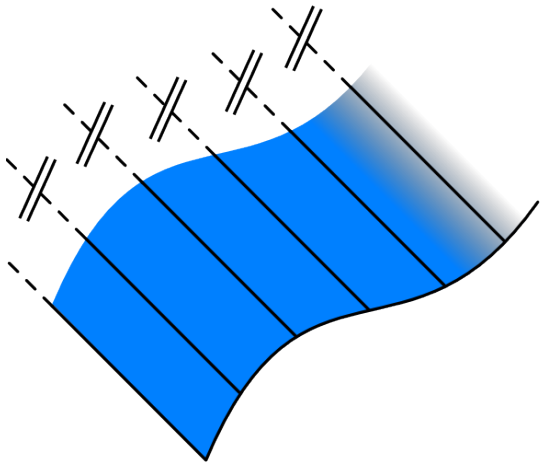


Figure 2.1: Example of a cylinder strip.

most people usually think of first when they hear the term "cylinder". Fig.(2.1) shows an example of a cylinder strip.

For a cylinder given by (2.12), one can compute the second derivatives $\mathbf{x}_{uu} = c''(u)$, $\mathbf{x}_{uv} = 0$, $\mathbf{x}_{vv} = 0$ and thus via (2.6) the coefficients of the matrix of the second fundamental form are, for the normal field given via (2.4),

$$\begin{aligned} L &= <\mathrm{N}, \mathbf{x}_{uu}> &= <\mathrm{N}, c''> , \\ M &= <\mathrm{N}, \mathbf{x}_{uv}> &= <\mathrm{N}, 0> = 0 , \\ N &= <\mathrm{N}, \mathbf{x}_{vv}> &= <\mathrm{N}, 0> = 0 . \end{aligned}$$

That means for $K = \det \mathrm{dN}_p$ with numerator $LN - M^2$, that the Gaussian curvature fulfills $K = 0$, independently of the considered point. Therefore cylindrical surfaces are developable and thus consist of parabolic (or planar) points only.

For computational purposes an useful representation of cylindrical surfaces is needed. As B-Spline surfaces provide a theory-rich method and nice properties, they are used to parametrize cylinder strips in the smooth case, while their control nets represent discrete cylinder strips, as shown in the subsequent sections.

2.2.2 Mesh representation

A map

$$\mathbf{x} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}^3 : (i, j) \mapsto x_{i,j} := \mathbf{x}(i, j) \tag{2.13}$$

is called 2-dimensional net. As only finite objects are treated in the following, (2.13) will be used as

$$\mathbf{x} : U \subset \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}^3 \ , \tag{2.14}$$

with U finite and connected. Further, the triple (V, E, F) containing discrete finite sets of vertices V, halfedges E and faces F with

$$\begin{aligned} V &= \{v_k | v_k = x_{i,j}\} \ , \\ E &= \{(x_{ij}, x_{k\ell}) | (i,j) \neq (k,\ell) \text{ and } (k = i + i_k \vee \ell = j + j_\ell \text{ with } i_k, j_\ell \in \{-1,0,1\})\}, \\ F &= \{(v_i, v_j, v_k) | (v_i, v_j), (v_j, v_k), (v_k, v_i) \in E\} \text{ or} \\ F &= \{(v_i, v_j, v_k, v_\ell) | (v_i, v_j), (v_j, v_k), (v_k, v_\ell), (v_\ell, v_i) \in E\} \ , \end{aligned}$$

depending on whether so-called **tri-meshes** or **quad-meshes** are considered, respectively, is called **mesh**.

Halfedges can be thought of as oriented edges, i.e. $(v_i, v_j) \neq (v_j, v_i)$, and are used because they deliver more information and allow systematic movement on the mesh. Faces are seen as being traversed counter-clockwise, which means that it is possible to orient them, see fig.(2.2).

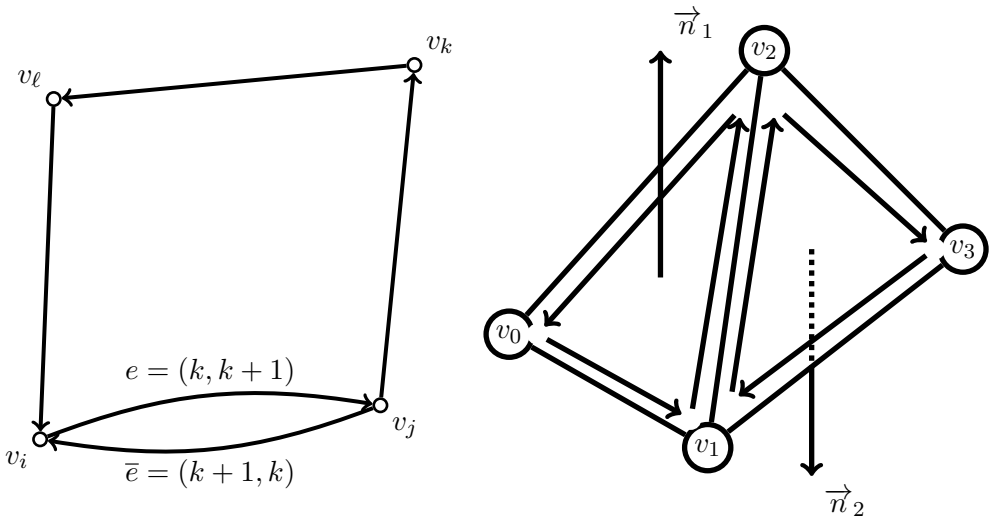


Figure 2.2: Exemplary quad-face, with halfedge e belonging to the face, while the opposite edge $\overline{e}$ is part of of the neighboring face.

On a mesh $\mathbf{x}(u, v)$, the operators τ_i and δ_i operate as follows,

$$\begin{aligned} \tau_1 \mathbf{x}(u,v) = \mathbf{x}(u+1,v) \quad &, \quad \tau_2 \mathbf{x}(u,v) = \mathbf{x}(u,v+1) \ , \\ \delta_1 \mathbf{x}(u,v) = \mathbf{x}(u+1,v) - \mathbf{x}(u,v) \quad &, \quad \delta_2 \mathbf{x}(u,v) = \mathbf{x}(u,v+1) - \mathbf{x}(u,v) \ . \end{aligned}$$

DEFINITION 2.18 *Let $M = (V, E, F)$ be a mesh, then* $\mathrm{val}(v) = \#\{f \in F | v \in f\}$ *is the* ***valence*** *of $v \in V$. If every face $f \in F$ is a quad and for all interior vertices v holds* $\mathrm{val}(v) = 4$*, it is called* ***quad-mesh****. Further, if every face is planar, the mesh is called* ***planar quad-mesh*** *or short* ***PQ-mesh****.*

It is easy to define normals on the faces, as the normals of the planes in which the faces lie can be taken, but in the following normals at vertices are needed. Thus, it is necessary to transfer the normals from the faces to the vertices of the mesh. For this purpose an adequate method must be found because in order to adopt smooth concepts to discrete objects, these must be viewed as approximations of smooth ones and therefore the normals at the vertices of the mesh should correspond to those of a smooth surface approximating the mesh locally.

The method chosen for this task is the vertex normal computation scheme described in [ZX06], which needs more computation time than more direct approaches but also ensures better general results. The idea is to compute a vertex normal considering the local geometry of the surface.

For the computation of a vertex normal, the surrounding faces have influence, that is for a normalized normal $\overrightarrow{n}$ of a vertex v with surrounding faces $\{f_i\} \subset F$

$$\overrightarrow{n} = \frac{\sum_{\overrightarrow{n}_i : v \in f_i} \omega_i \overrightarrow{n}_i}{||\sum_{\overrightarrow{n}_i : v \in f_i} \omega_i \overrightarrow{n}_i||}$$

with appropriate weights ω_i. What "appropriate" means differs from case to case. Some weights exhibit very good results with one class of surfaces and different ones with another class of surfaces. In this thesis the weights will be chosen to yield overall useful results.

Based on the *mean weighted by sine and edge length reciprocals* (MWSELR) method with weights $\omega_i = \frac{\sin \alpha_i}{||vv_i|| \cdot ||vv_{i+1}||}$, where $\alpha_i = \angle(vv_i, vv_{i+1})$, the weights are chosen to second order approximate a surface for which the mesh can be seen as a discretization. Thus take

$$\omega_i = \frac{\sin \alpha_i}{||vv_i|| k_i || vv_{i+1}|| k_{i+1}} \tag{2.15}$$

with k_j is the normal curvature of the shortest curve $g_i(s)$ parametrized by arc-length connecting v and v_i at $g_i(s_i) = v_i$. The curvatures k_j are replaced with the second order approximates k_j^* and substituting $\overrightarrow{a}_j := vv_j$ and $x_j := ||\overrightarrow{a}_j|| k_j^*$ yields for the weights in (2.15) the approximation $\omega_i = \frac{\sin \alpha_i}{x_i x_{i+1}}$ where x_j is part of the solution of the linear system $\mathrm{A}\overrightarrow{x} = \overrightarrow{b}$ with

$$\overrightarrow{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad , \quad \overrightarrow{b} = \begin{pmatrix} < \frac{\overrightarrow{a}_1}{||\overrightarrow{a}_1||}, \frac{\overrightarrow{a}_2 \wedge \overrightarrow{a}_3}{||\overrightarrow{a}_2 \wedge \overrightarrow{a}_3||} > \\ < \frac{\overrightarrow{a}_2}{||\overrightarrow{a}_2||}, \frac{\overrightarrow{a}_3 \wedge \overrightarrow{a}_4}{||\overrightarrow{a}_3 \wedge \overrightarrow{a}_4||} > \\ \vdots \\ < \frac{\overrightarrow{a}_n}{||\overrightarrow{a}_n||}, \frac{\overrightarrow{a}_1 \wedge \overrightarrow{a}_2}{||\overrightarrow{a}_1 \wedge \overrightarrow{a}_2||} > \end{pmatrix} ,$$

$$\mathrm{A} = \frac{1}{2}\begin{pmatrix} 1 & -\frac{\sin(\alpha_{1,2})}{\sin\alpha_2} & \frac{\sin\alpha_1}{\sin\alpha_2} & 0 & \dots & 0 & 0 \\ 0 & 1 & -\frac{\sin(\alpha_{2,3})}{\sin\alpha_3} & \frac{\sin\alpha_2}{\sin\alpha_3} & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -\frac{\sin(\alpha_{n-2,n-1})}{\sin\alpha_{n-1}} & \frac{\sin\alpha_{n-2}}{\sin\alpha_{n-1}} \\ \frac{\sin\alpha_{n-1}}{\sin\alpha_n} & 0 & 0 & 0 & \dots & 1 & -\frac{\sin(\alpha_{n-2,n})}{\sin\alpha_n} \\ -\frac{\sin(\alpha_{n,1})}{\sin\alpha_1} & \frac{\sin\alpha_n}{\sin\alpha_1} & 0 & 0 & \dots & 0 & 1 \end{pmatrix}$$

where $\alpha_{i,j} := \alpha_i + \alpha_j$.

For a detailed description see [ZX06].

Consider a cylindrical strip, a discrete approximation can be chosen to be a is a PQ-mesh. The directrix $c(u)$ in definition 2.17 is replaced by a polyline $c = (c_0, c_1, \dots, c_n)$ which can be defined formally as $c : \mathbb{Z} \to \mathbb{R}^3$ with $c(i) =: c_i$ and the faces $f_0, \dots, f_{n-1}$ are given by

$$f_j = (v_{0,j}, v_{1,j}, v_{1,j+1}, v_{1,j})$$

with

$$\begin{aligned} v_{0,i} &= c_i + \ell_{0,i}\,\overrightarrow{d} \\ v_{1,i} &= c_i + \ell_{1,i}\,\overrightarrow{d}\ . \end{aligned}$$

where $\ell_{1,i} - \ell_{0,i}$ defines the width of the edge e_i. Therefore a discrete cylinder is a quad mesh where the edges $e_i = \{v_{0,i}, v_{1,i}\}$ are pairwise parallel, thus the faces are planar and it is a special case of a PQ-mesh, see fig.(2.3). Conversely, a smooth cylinder strip can be seen as the limit of a discrete cylinder strip undergoing a subdivision process with quads tending to rulings, motivating the classification of cylinder strip models as semi-discrete objects and the representation as B-Spline surfaces for computational purposes.

2.2.3 B-Splines

B-Splines exhibit a rich theory of their own, but since they are just used as a tool for other calculations, the given statements only provide a rough introduction and overview.

For a B-Spline curve of order k with control points $\mathbf{b}_i$, $i = 0, 1, \dots, n$ one needs an ordered knot vector $T = (t_0, t_1, \dots, t_{n+k})$, $t_0 < t_1 < \dots < t_{n+k}$, in order to define B-Spline basis functions. The vector T is also called support.

The normalized B-Spline basis functions $N_i^k(t)$, $i = 0, \dots, n$, of order k are defined via

$$\begin{aligned} N_i^1(t) &:= \begin{cases} 1 & \text{, for } t_i \le t < t_{i+1} \\ 0 & \text{, else} \end{cases} \text{, and for } k > 1 \\ N_i^k(t) &:= \frac{t - t_i}{t_{i+k-1} - t_i} N_i^{k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1}^{k-1}(t)\ . \end{aligned} \tag{2.16}$$

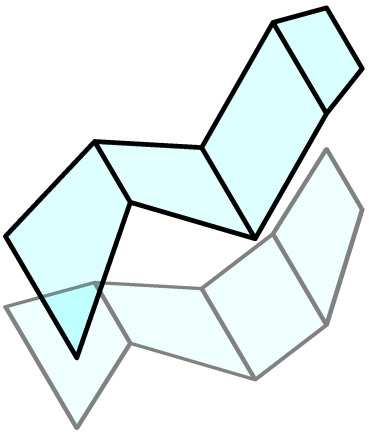


Figure 2.3: A cylindrical strip as a special case of a PQ-mesh.

Direct consequences of this definition are $N_i^k(t) > 0$ for $t \in (t_i, t_{i+k})$ and $N_i^k(t) = 0$ for $t \in [t_0, t_i] \cup [t_{i+k}, t_{n+k}]$, because the fractions in (2.16) are non-negative. Further, for $t \in [t_{k-1}, t_{n+1}]$ holds

$$\sum_{i=0}^{n} N_i^k(t) = 1 \tag{2.17}$$

which, together with the two above properties, is called partition of one.

DEFINITION 2.19 *For points* $(\mathbf{b}_i)_{i=0,\ldots,n}$, $\mathbf{b}_j \in \mathbb{R}^3$, *and an ordered knot vector* $T = (t_0, t_1, \ldots, t_{n+k})$ *the function*

$$X : [t_{k-1}, t_{n+1}] \to \mathbb{R}^3 : t \mapsto \sum_{i=0}^{n} N_i^k(t)\mathbf{b}_i \tag{2.18}$$

defines the ***B-Spline curve*** *of order* k *with support* T, *where* $n \geq k-1$ *and the points* $\mathbf{b}_j$ *are called* ***de Boor points****.*

B-Spline curves do generally not interpolate their endpoints. In order to achieve endpoint interpolation, which is desired, it is necessary to adapt the support of the curve, which is stated explicitly after some properties are noted.

Such properties include affine invariance, which allows a location independent description as well as local polynomial form. A B-Spline curve exhibits local control, which means that moving a de Boor point only changes the curve in a neighborhood. More formally, the number of intervals that influence a point on the curve is given by $(n - k + 2)$; see (2.17) which motivates the restricted domain of (2.18). A de Boor

point $\mathbf{b}_i$ influences the curve in the parameter interval $[t_i, t_{i+k}]$ and $X(t)$ depends on $\mathbf{b}_{j-k+1}, \ldots, \mathbf{b}_j$, for $t \in (t_j, t_{j+1})$.

For B-Spline curves the **convex hull property** holds, which means that every point $X(t)$ lies inside the convex hull of its k depended de Boor points $\mathbf{b}_i$. Additionally, for pairwise different knots t_j the B-Spline curves of order k have differential order $(k-2)$. If one also allows m subsequent knots to be equal, the differential order is reduced to $k-2-(m-1)$, thus $X(t) \in C^{k-m-1}$, which is a consequence of the differential order of the basis functions.

Finally, endpoint interpolation can be achieved by setting the first, respectively the last $(k-1)$ knots to be equal, e.g.

$$T = (\underbrace{0,\ 0,\ \ldots,\ 0}_{(k-1)\ times},\ \ 1,\ 2,\ \ldots,\ n-k+3,\ \ \underbrace{n-k+4,\ n-k+4,\ \ldots,\ n-k+4}_{(k-1)\ times})\ .$$

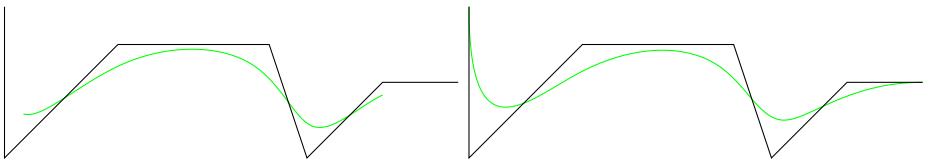


Figure 2.4: Example of B-Spline curve with and without endpoint interpolation.

Important for the following tasks is a surface description based on B-Splines, namely the tensor product B-Spline surface which allows easy description and fast evaluation.

DEFINITION 2.20 *The **tensor product B-Spline surface** of orders k and ℓ is defined by*

$$X(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_i^k(u) N_j^\ell(v) \mathbf{b}_{ij}\ , \tag{2.19}$$

*where $(\mathbf{b}_{ij})_{i,j}$ is called **de Boor net**.*

Setting $v = v_0 = const$ in (2.19), one can substitute $\mathbf{P}_i := \sum_{j=0}^{n} N_j^\ell(v_0)\mathbf{b}_{ij}$ to obtain the B-Spline curve

$$X(u) = \sum_{i=0}^{n} N_i^k(u)\mathbf{P}_i \tag{2.20}$$

and for $u = u_0 = const$ in an analogous manner. Hence fixing one parameter yields a B-Spline curve and thus all of their properties. For evaluation of $X(u_0, v_0)$ evaluate the curve $X(u, v_0)$ to get the curve (2.20) which then can be evaluated again to get the desired point. A fast and stable method for the evaluation is given by the *de Boor algorithm*, see [HL92].

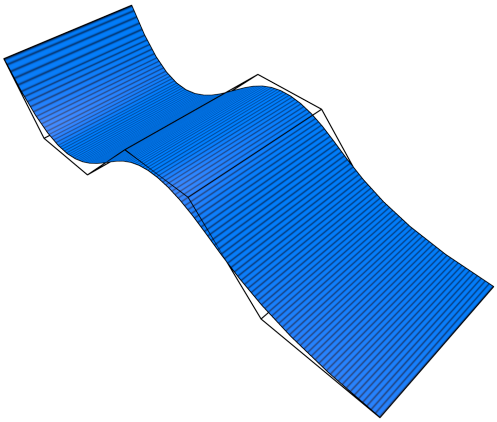


Figure 2.5: Example of tensor product B-Spline surface with marked parameter curve and cylindrical strip represented as tensor product B-Spline surface.

Take a polyline $c_i = (p_0, p_1, \ldots, p_n)$ with direction $\overrightarrow{d}$ and set up the de Boor net $C = (\mathbf{a}_j, \mathbf{b}_i)_j$ with control points at positions

$$\begin{aligned} \mathbf{a}_j &= p_j - \delta_j \overrightarrow{d} \ , \\ \mathbf{b}_j &= p_j + \delta_j \overrightarrow{d} \end{aligned}$$

for suitable $\delta_j \in \mathbb{R}$ and $j = 0, \ldots, n$. This clearly is a discrete cylinder strip.
The surface strip $\mathbf{x}(u, v)$, which uses C as a control surface, given by

$$\mathbf{x}(u, v) = (1 - v)a(u) + vb(u) \ , \tag{2.21}$$

where

$$\begin{aligned} a(u) &= \sum_j N_j^4(u)\mathbf{a}_j \\ b(u) &= \sum_j N_j^4(u)\mathbf{b}_j \ , \end{aligned}$$

is a B-Spline surface with degrees 1 and 3 (it is of order 2 and 4, respectively) and represents a smooth cylinder strip, created from a discrete cylinder strip using a subdivision process. For an example see fig.(2.5).

2.3 Optimization Problems

This section deals with the foundations of the optimization methods used in this thesis. To gain deeper insight one is referred to [GK02] and [Kel04].

2.3.1 Classification

DEFINITION 2.21 *Given $f : \mathbb{R}^n \to \mathbb{R}$, the problem*

$$\min f(x) \qquad \text{subject to} \qquad x \in \mathbb{R}^n \tag{2.22}$$

*is called an **unconstrained optimization problem** with objective function f. Let $X \subset \mathbb{R}^n$ be non-empty and $f : X \to \mathbb{R}$, then the problem*

$$\min f(x) \qquad \text{s.t.} \qquad x \in X \tag{2.23}$$

*is called a **constrained optimization problem** with objective function f and feasible region X.*

Similarly, one can define (2.22) and (2.23) using $\max f(x)$. As in (2.23), the term "subject to" will be abbreviated "s.t." for the rest of the thesis.

DEFINITION 2.22 *Assume $n, m \in \mathbb{N}\backslash\{0\}$, then the optimization problem*

$$\min f(x) \qquad \text{s.t.} \qquad g(x) = b, \tag{2.24}$$

*is called **quadratic programming**, if $f(x)$ is quadratic and $g(x)$ is linear, i.e.*

$$\min \frac{1}{2} \overrightarrow{x}^T \mathrm{Q} \overrightarrow{x} + \overrightarrow{c}^T \overrightarrow{x} + \gamma \qquad \text{s.t.} \qquad \mathrm{A} \overrightarrow{x} = b, \tag{2.25}$$

with $\mathrm{Q} \in \mathbb{R}^{n \times n}$ symmetric, $\overrightarrow{c} \in \mathbb{R}^n$, $\gamma \in \mathbb{R}$, $\mathrm{A} \in \mathbb{R}^{m \times n}$ and $\overrightarrow{b} \in \mathbb{R}^m$.

Convex optimization problems (those where $g(x)$ is a convex function) have the advantage that local minima are already global minima , thus they are favored, but as convexity cannot always be guaranteed, local optimization must be satisfactory in the following.

DEFINITION 2.23 *A feasible $x^* \in X$ is called **local constrained minimizer** if*

$$f(x^*) \leq f(x) \ , \quad \forall x \in U_\varepsilon(x^*) := \{y : ||x^* - y|| < \varepsilon\} \ . \tag{2.26}$$

To identify a necessary (but not sufficient) condition the subsequent definitions provide a tool.

2.3.2 Methods

DEFINITION 2.24 *The function*

$$L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} : (x, \lambda) \mapsto f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

*is called **Lagrange-function** of the optimization problem (2.24).*

DEFINITION 2.25 *Let $f(x)$ and $g(x)$ in (2.24) be continuously differentiable and let $\nabla_x L(x,\lambda) := \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x)$ denote the gradient of the Lagrange-function with respect to x. Then the* ***Karush-Kuhn-Tucker conditions*** *(KKT conditions) are defined by*

$$\begin{aligned} \nabla_x L(x,\lambda) &= 0 \ , \\ \lambda_i g_i(x) &= 0 \ , \quad \forall i \in \{1,\ldots,m\}. \end{aligned}$$

Every point (x^*,λ^*) that satisfies the KKT conditions is called ***KKT-point*** *and λ_i^* are called* ***Lagrange-multipliers****.*

With definition 2.25 it is possible to express the necessary first order conditions.

THEOREM 2.26 *Let x^* be a local constrained minimizer of a constrained optimization problem with affine linear constraints $g_i(x)$. Then there exist Lagrange-multipliers λ_i^* so that (x^*,λ^*) is a KKT-point of the optimization problem.*

COROLLARY 2.27 *Let x_s be feasible, then a pair (x_s,λ_s) is a KKT-point if and only if x_s is a stationary point.*

For second order sufficient and necessary conditions, the second partial derivatives are needed, therefore the presented functions in any occurring optimization problem should be at least two times differentiable. Because of the formulation, it will be always convenient that this is fulfilled.

The case of quadratic optimization problems is treated more in detail first. They are of the form

$$\min f(x) = \frac{1}{2} x^T \mathrm{H} x + c^T x + \gamma \qquad \text{s.t.} \qquad g(x) = \mathrm{J}x = 0 \tag{2.27}$$

where $\mathrm{H} = f''(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{i,j}$ denotes the Hessian of the objective function and $\mathrm{J} = g'(x) = \left(\frac{\partial g_i}{\partial x_j}\right)_{i,j}$ denotes the Jacobian of the constraint function.

Given $(x^*,\lambda^*) \in \mathbb{R}^n \times \mathbb{R}^m$ then (x^*,λ^*) is a KKT-point of (2.27) if and only if (x^*,λ^*) is solution of the linear system

$$\begin{pmatrix} \mathrm{H} & -\mathrm{J}^T \\ -\mathrm{J} & 0 \end{pmatrix} \begin{pmatrix} \overrightarrow{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} -\overrightarrow{c} \\ 0 \end{pmatrix} . \tag{2.28}$$

The KKT conditions 2.25 applied to (2.27) are of the form

$$\begin{aligned} \mathrm{H}\overrightarrow{x} + \overrightarrow{c}^T - \mathrm{J}^T\lambda &= 0 \ , \\ -\mathrm{J}\overrightarrow{x} &= 0 \end{aligned}$$

therefore a pair (x^*,λ^*) that fulfills the KKT conditions is also a solution of (2.28) and conversely.

An important class of methods for solving quadratic optimization problems is **Sequential Quadratic Programming** (SQP). These iterative methods are described more in detail in [MNT04],[GK02].

So given a current iterate (x_c, λ_c), one wants to find the next iterate $(x_c + \Delta x, \lambda_c)$ that fulfills (2.28), thus with

$$\begin{aligned}
\begin{pmatrix} \mathrm{H} & -\mathrm{J}^T \\ -\mathrm{J} & 0 \end{pmatrix} \begin{pmatrix} x_c + \Delta x \\ \lambda_c \end{pmatrix} &= \begin{pmatrix} -\overrightarrow{c} \\ 0 \end{pmatrix} \Leftrightarrow \\
\begin{pmatrix} \mathrm{H} & -\mathrm{J}^T \\ -\mathrm{J} & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \lambda_c \end{pmatrix} &= \begin{pmatrix} -\overrightarrow{c} - \mathrm{H}x_c \\ \mathrm{J}x_c \end{pmatrix} \Leftrightarrow \\
\begin{pmatrix} \mathrm{H} & -\mathrm{J}^T \\ -\mathrm{J} & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \lambda_c \end{pmatrix} &= \begin{pmatrix} -\nabla f(x_c) \\ g(x_c) \end{pmatrix}
\end{aligned} \tag{2.29}$$

the step direction Δx can be determined by (2.29). In [MNT04] this system is modified by exchanging $H = f''(x)$ with $W := \frac{\partial^2}{\partial x^2} L(x, \lambda)$ and therefore adapting the system for a faster convergence.

If x^* is a local minimum, the H is positive semi-definite and the Cholesky decomposition $\mathrm{H} = \mathrm{L}\mathrm{L}^T$ exists, with L being a non-singular lower triangular matrix with positive diagonal. The Cholesky decomposition provides a stable and useful method for solving smaller linear systems.

Another approach is the so called penalty method which puts constrained optimization down to unconstrained optimization. The constraint, which gives the feasible region, is used to "punish" the objective function through a penalty term if it leaves the feasible region, e.g. the penalty term is zero for a feasible element and positive else. There are different ways of defining a penalty function, one is given below.

DEFINITION 2.28 *Given an optimization problem of the form*

$$\min f(x) \qquad \text{s.t.} \qquad g(x) = 0 \ , \tag{2.30}$$

then the corresponding ***penalty function*** *is*

$$P : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R} : (x, \alpha) \mapsto f(x) + \frac{\alpha}{2} g(x)^T g(x) \ .$$

The parameter α is called penalty parameter.

For **Nonlinear Least Squares** problems, these are problems with objective function of the form

$$f(x) = \frac{1}{2} \sum_{i=1}^{N} ||r_i(x)||^2 \ , \tag{2.31}$$

the iterative **Gauss-Newton method** can be applied. The functions $r_j(x)$ are called residuals. In order to solve the problem, the second order derivatives are needed,

which, given that $f(x)$ can be written as $f(x) = \frac{1}{2}R(x)^T R(x)$ using the vector $R(x)^T = (r_1(x), r_2(x), \ldots, r_N(x))$, are

$$\begin{aligned} \nabla f(x) &= R'(x)^T R(x) \ , \\ \nabla^2 f(x) &= R'(x)^T R'(x) + R''(x)^T R(x) \ . \end{aligned}$$

In the above equations $R'(x)$ stands for the Jacobian matrix of $f(x)$, $R''(x)$ is a tensor, so the term $R''(x)^T R(x)$ may be written as $\sum_i r_i(x)^T \nabla^2 r_i(x)$ with Hessians $\nabla^2 r_i(x)$. As the computations of the latter term are very costly, they should be avoided. The Gauss-Newton method bypasses it by computing the update step x_+ for an iterate x_k using the approximation

$$f(x) = f(x_k) + (x - x_k)\nabla f(x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

by

$$x_+ = -(R'(x_k)^T R'(x_k))^{-1} R'(x_k)^T R(x_k) \ . \tag{2.32}$$

This is a consequence of the necessary first order condition for optimization problems. With $f(x_k + x_+) = f(x_k) + \nabla f(x_k)x_+ + \mathcal{O}(x_+^2)$, it reads

$$\nabla f(x_k + x_+) = \nabla f(x_k) + \nabla^2 f(x_k)x_+ + \mathcal{O}(x_+^2) = 0 \ .$$

The next iterate is $x_{k+1} = x_k + x_+$ and the omitted term $R''(x_k)^T R(x_k)$ does not cause much trouble assumed that the initial point x_k is near an optimum, because it would vanish for a zero residual problem.

If the matrix $R'(x_k)^T R'(x_k)$ in (2.32) does not have full rank or is not uniformly bounded and well-conditioned, one can add a regularization parameter $\nu > 0$, called **Levenberg-Marquardt parameter**, so

$$x_+ = -(\nu_k \mathrm{I} + R'(x_k)^T R'(x_k))^{-1} R'(x_k)^T R(x_k) \ .$$

3 Rationalization

3.1 Initialization

In order to cover a surface with cylindrical strips, one needs to find curves on the surface along which these strips can be laid out. The approach undertaken in this thesis to find such curves is by using parallel illumination. Via parallel illumination one can cover a surface with contours. Evaluating these contours and extracting "good" ones leads to a set of curves which can be used as directrices for the cylindrical strips.

It is a well studied area in computer graphics, as the area of non-photorealistic rendering heavily relies on good algorithms for this task. A good overview can be found in [Ise+03], where a rough classification of the methods is done and related algorithms are mentioned. The authors note, that – contrary to this thesis – some distinguish between contours and silhouettes, for example the contours being a subset of the silhouettes, and define other lines, often called feature lines, like creases (edges with sufficient big angles between adjacent triangles), borders of open meshes and self-intersections. In the classification of the contour detection algorithms into image space algorithms, hybrid algorithms and object space algorithms, the last ones are of interest for the tasks in this thesis. This is because the contours are used as a first input for further computations and thus shall be available in an analytic description which cannot be provided by image space or hybrid algorithms in general, as the former ones operate on pixel matrices and the latter ones too, after some geometry manipulations in object space.

In object space, pre-computations can be done to reduce the computation time of the contours, but these methods are omitted as there is no need to accept the time for pre-computations when the actual algorithms rerun only a limited number of times and do not need to terminate in interactive spans. An example for such an algorithm is given in [Wan+08], where hierarchial clustering accelerates the computations at the expense of additional data structures. The idea in that paper is to view the normalized face normals as elements of the Gaussian sphere and split it into cells delimited by four normals in each case. Therefore, the visibility of the faces within each cell can be determined by the corner elements of those cells in case they share the same visibility or otherwise via subdividing the cells.

This chapter starts with the basics of parallel illumination and defines and describes contour generators, stating many properties for a better understanding. The definitions and theorems follow closely [CG00]. Afterwards, the theory is adapted to be applied on meshes, mentioning the necessary steps to achieve suitable results for the use as an initial input in the approximation phase.

3.1.1 Parallel Illumination and Contour Generators

Given a plane $\Sigma \subset \mathbb{R}^3$ through the origin with normal $\overrightarrow{d} \in \mathbb{R}^3$ and a surface $S \subset \mathbb{R}^3$, the set

$$P(\overrightarrow{d}) := \{q \in \Sigma \mid \exists p \in S : \exists \lambda \in \mathbb{R} : \; q + \lambda \overrightarrow{d} = p\}$$

is the **orthographic projection** of S onto Σ.

The use of a plane through the origin is not necessary, but since all orthographic projections onto a plane that do not contain the origin are parallel translates along the normal $\overrightarrow{d}$, this easy to handle definition is sufficient.

The useful part of the orthographic projection is the boundary, but as stated below, a bit more than this is needed. That is the apparent contour, a superset of the boundary of the orthographic projection. For a surface $S \subset \mathbb{R}^3$ and a point $p \in S$, let $\mathrm{N} : S \to \mathbb{R}^3$ be the normal vector field and $\mathrm{N}_p := \mathrm{N}(p)$ the normal vector in the point p.

DEFINITION 3.1 *The set*

$$\Gamma(\overrightarrow{d}) := \{p \in S \mid \overrightarrow{d} \cdot \mathrm{N}_p = 0\} \tag{3.1}$$

is called ***contour generator*** *and the set*

$$\gamma(\overrightarrow{d}) := \{q \in \mathbb{R}^3 \mid \overrightarrow{d} \cdot q = 0 \text{ and } \exists \lambda \in \mathbb{R} : q + \lambda \overrightarrow{d} \in \Gamma(\overrightarrow{d})\} \tag{3.2}$$

apparent contour*.*

This definition is accompanied by fig.(3.1).

There is no consistent nomenclature, the *contour generator* is also called *rim* in some papers and the *apparent contour* is referred to as *contour*. There is also a difference between opaque and semi-opaque surfaces. While definition 3.1 uses semi-opaque surfaces, where only the surface normal is crucial, one must consider visibility in case of opaque surfaces. That means that $\Gamma(\overrightarrow{d}) \neq \Gamma(-\overrightarrow{d})$ may occur and the position of the light source is important too. This thesis only uses semi-opaque surfaces, which means that contours actually visually hidden behind parts of the surface are treated as part of the contour generator, i.e. the visual ray pierces the surface.

Assume S is a surface, then in order to identify problematic points, the apparent contour can be used to detect them as it exhibits many properties of the contour generator. Let $\Gamma(\overrightarrow{d})$ be a contour generator on S and $p \in \Gamma(\overrightarrow{d})$ an arbitrary contour point. With $\pi(p)$ denoting the projection of p onto the image plane of the apparent contour and $\overrightarrow{v}_{\pi(p)}$ the tangent to $\gamma(\overrightarrow{d})$ at $\pi(p)$, the normal at p is parallel to $\overrightarrow{v}_{\pi(p)} \wedge \overrightarrow{d}$ if $\gamma(\overrightarrow{d})$ is regular at $\pi(p)$. If $\gamma(\overrightarrow{d})$ has a **cusp** at $\pi(p)$ – that means for $\gamma(\overrightarrow{d})(s_0) = \pi(p)$ that $\gamma(\overrightarrow{d})'(s_0) = 0$ – one has to distinguish two cases.

The first case is an ordinary cusp, defined by $\gamma(\overrightarrow{d})'(s_0) = 0$ and $\gamma(\overrightarrow{d})''(s_0) \neq 0$. Then the limit normal at $\pi(p)$ is parallel to the surface normal at p. An ordinary cusp indicates an asymptotic direction at the contour generator, hence the tangent

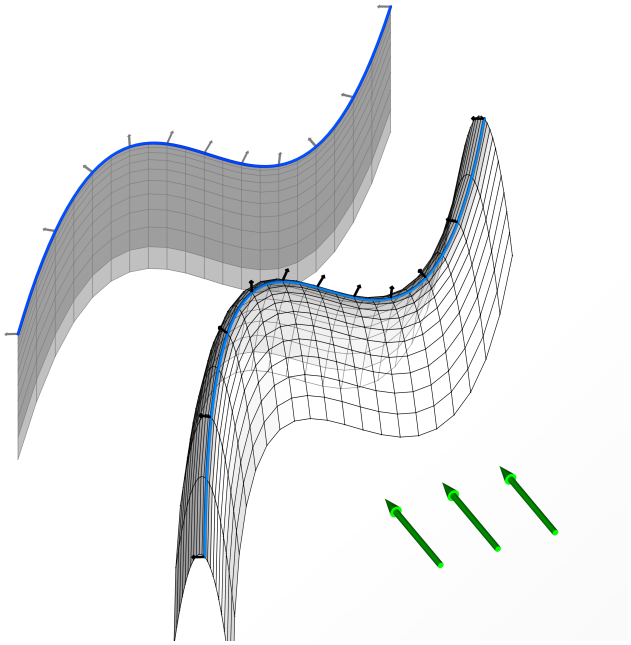


Figure 3.1: Example of a contour generator and apparent contour.

to $\Gamma(\overrightarrow{d})$ at p denoted as $\overrightarrow{v}_p$ and $\overrightarrow{d}$ are parallel, so the tangential line has exactly 3-point contact with the surface at p (this means that for a local parametrization of the surface $\mathbf{x}$ and a parametrization of the tangential line ℓ, that $(\mathbf{x} - \ell)$ has a root of order 3 in p). The second case is a cusp where the tangential line to $\Gamma(\overrightarrow{d})$ passing through p has at least 4-point contact with the surface, also called flecnodal point.

Thus, an apparent contour $\gamma(\overrightarrow{d})$ is smooth at a point $\pi(p)$ unless $\overrightarrow{d}$ is an asymptotic direction at p, therefore the apparent contour gives information about problematic points of the contour generator.

Another special type of contour generators, one with two crossing contours in one point, called **beaks point**, can be detected using the apparent contour if depth information (the distance between p and $\pi(p)$) is stored. A beaks point is a singular point of the contour generator and given the apparent contour components $\{\gamma_i(\overrightarrow{d})\}_i$ in the same plane of different components belonging to one contour generator, their intersection points are beaks points if the depth information agree. Additionally, a contour generator can contain an isolated point, a so-called **lips point**.

An example surface given by $\mathbf{x}(u, v) = \left(u, v, -(v^2 - u)^2\right)$ with illumination direction along the x-axis is shown in fig.(3.2). In (a), the contour with asymptotic point in $p = (0, 0, 0)$ is highlighted and in (b), the entire contour generator is outlined with the beaks point even coincident with the asymptotic point.

The contour generator is defined as a point set, but it will also be treated as a set of continuous curves. The projection of a contour is likewise seen as a connected

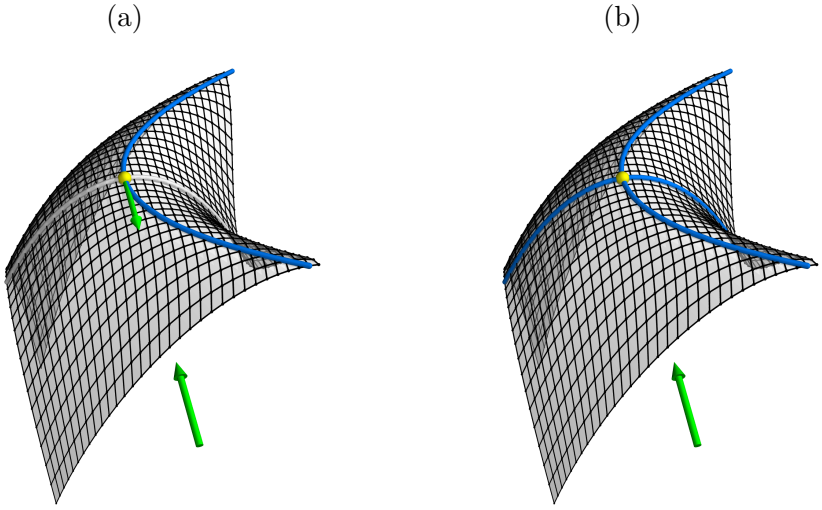


Figure 3.2: Contour generators.

component of the apparent contour.

To conclude this section it will be checked that the property of conjugate directions, defined in 2.10, also holds for contour generators in a certain sense, before the next section tackles this property more in detail.

LEMMA 3.2 *Given a surface S, an illumination direction $\overrightarrow{d}$ and a contour generator $\Gamma(\overrightarrow{d})$ on S. Then, if $\Gamma(\overrightarrow{d})$ is smooth, for every $p \in \Gamma(\overrightarrow{d})$ the tangent $\overrightarrow{v}$ at p is conjugate to $\overrightarrow{d}$.*

Proof As stated in definition 2.10, two elements in T_pS are conjugate at $p \in S$ if the second fundamental form is zero or equally in coordinates $(d_1, d_2)\mathrm{II}(v_1, v_2)^T = 0$. Without loss of generality, assume $p = (0,0,0)^T$ and the surface S parametrized by

$$\mathbf{x}(x,y) = \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix} \quad , \quad \mathbf{x}(0,0) = p = 0$$

in a neighborhood of p with $\overrightarrow{d} = (1,0,0)^T$ and the tangential plane to agree with the xy-plane. Hence

$$\mathbf{x}_x = \begin{pmatrix} 1 \\ 0 \\ f_x \end{pmatrix} \quad , \; \mathbf{x}_y = \begin{pmatrix} 0 \\ 1 \\ f_y \end{pmatrix} \quad ,$$

$$\mathbf{x}_{xx} = \begin{pmatrix} 0 \\ 0 \\ f_{xx} \end{pmatrix} \quad , \; \mathbf{x}_{yy} = \begin{pmatrix} 0 \\ 0 \\ f_{yy} \end{pmatrix} \quad , \; \mathbf{x}_{xy} = \begin{pmatrix} 0 \\ 0 \\ f_{xy} \end{pmatrix}$$

and

$$\mathrm{N} = \frac{\mathbf{x}_x \wedge \mathbf{x}_y}{||\mathbf{x}_x \wedge \mathbf{x}_y||} = \frac{1}{\sqrt{f_x^2 + f_y^2 + 1}} \begin{pmatrix} -f_x \\ -f_y \\ 1 \end{pmatrix} .$$

With $z = f(x,y) = \frac{1}{2}(ax^2 + 2bxy + cy^2) + \mathcal{O}\left(\{x,y\}^3\right)$ – the linear term vanishes because the tangential plane has normal parallel to $\overrightarrow{n} = (0,0,1)^T$, thus $f_x(0,0) = 0$ and $f_y(0,0) = 0$ – follows

$$f_{xx}(0,0) = a \ , \quad f_{yy}(0,0) = b \ , \quad f_{xy}(0,0) = c \tag{3.3}$$

as well as

$$\mathrm{N} \cdot \overrightarrow{d} = 0 \tag{3.4}$$

$$\frac{1}{\sqrt{f_x^2 + f_y^2 + 1}} \begin{pmatrix} -f_x \\ -f_y \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = 0$$

$$f_x = 0 \tag{3.5}$$

and at $(x,y) = (0,0)$

$$\begin{aligned} L &=< \mathrm{N}, \mathbf{x}_{xx} > &&= \frac{1}{\sqrt{f_x^2 + f_y^2 + 1}} \begin{pmatrix} -f_x \\ -f_y \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ a \end{pmatrix} \\ & && = a \ , \\ M &=< \mathrm{N}, \mathbf{x}_{xy} > &&= b \ , \\ N &=< \mathrm{N}, \mathbf{x}_{yy} > &&= c \ , \end{aligned}$$

so

$$\mathrm{II} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} .$$

Given that (3.4) is the condition for a point to be an element of the contour generator $\Gamma(\overrightarrow{d})$, one can write down its equation explicitly using (3.3) and (3.5);

$$\Gamma(\overrightarrow{d}) : \qquad ax + by + \mathcal{O}\left(\{x,y\}^2\right) = 0 \ . \tag{3.6}$$

Suppose $a \neq 0$ or $b \neq 0$ (else compare asymptotic directions), w.l.o.g. $a \neq 0$, then (3.6) can be transformed into

$$x = -\frac{b}{a} y + \mathcal{O}(\{x,y\}^2)$$

and thus

$$\begin{aligned} z = f(x,y) &= f\left(-\frac{b}{a} y + \mathcal{O}\left(\{x,y\}^2\right), y\right) \\ &= \frac{ac - b^2}{2a} y^2 + \mathcal{O}\left(\{x,y\}^3\right) \end{aligned}$$

along $\Gamma(\overrightarrow{d})$. These computations serve as a tool to express the tangent of $\Gamma(\overrightarrow{d})$ at p. All that is left to do now is differentiating the curve Γ,

$$\Gamma(\overrightarrow{d})(y) = \begin{pmatrix} -\frac{b}{a}y + \mathcal{O}(\{x,y\}^2) \\ y \\ \frac{ac-b^2}{2a}y^2 + \mathcal{O}(\{x,y\}^3) \end{pmatrix} \Rightarrow \Gamma'(\overrightarrow{d})(0) = \begin{pmatrix} -\frac{b}{a} \\ 1 \\ 0 \end{pmatrix} .$$

Since $\overrightarrow{d} = (1,0,0)^T$ and $\overrightarrow{v} = \Gamma'(\overrightarrow{d})(0) = \left(-\frac{b}{a}, 1, 0\right)^T$ lie in the tangent plane T_pS with $p = (0,0,0)^T$ and plane equation $z = 0$, the elements $e_1 = (1,0,0)^T, e_2 = (0,1,0)^T$ clearly form a basis of T_pS and in this basis, $\overrightarrow{d}$ has coordinates $(d_1, d_2) = (1,0)$ and $(v_1, v_2) = \left(-\frac{b}{a}, 1\right)$. To conclude the proof

$$\begin{aligned} (d_1,d_2)\mathrm{II}(v_1,v_2)^T &= (1,0)\begin{pmatrix} a & b \\ b & c \end{pmatrix}\begin{pmatrix} -\frac{b}{a} \\ 1 \end{pmatrix} \\ &= (a,b)\begin{pmatrix} -\frac{b}{a} \\ 1 \end{pmatrix} \\ &= -b + b = 0 \ . \end{aligned}$$

■

3.1.2 Conjugate curve networks

This section deals with conjugate curve networks, as their property of covering (regular) surfaces with a two-parametric family of curves, is used for initialization in the following sections. A comprehensive overview over conjugate nets and their discrete counterparts - these are PQ-meshes - can be found in [Sau70].

The term conjugate was introduced in definition 2.10. An example of a conjugate net is the principle curvature net. Given a surface S then for every point $p \in S$ one can compute the principle directions $\overrightarrow{e}_1, \overrightarrow{e}_2$ in p as defined in 2.13. From proposition 2.12 it follows immediately that $\overrightarrow{e}_1$ and $\overrightarrow{e}_2$ are conjugate. This enables a parametrization of a surface along the principle curvature lines which in turn is a conjugate net parametrization. Although this is a useful method in order to initialize for conical meshes, as shown for example in [Liu+06], the approach taken here to initialize for cylindrical meshes is via conjugate nets from parallel illumination.

To gain more insight, the conjugacy property of contour generator tangent and illumination direction is generalized to tangent surfaces which touch a given surface along a curve.

First it is useful to rewrite the second fundamental form using (2.10). Let $\mathbf{x}_1(t) := \mathbf{x}(u_1(t), v_1(t))$ and $\mathbf{x}_2(t) := \mathbf{x}(u_2(t), v_2(t))$ be two curves on a surface S which intersect in a common point p. Then these two curves are conjugated in p if per definition for $\mathbf{x}_1' = \mathbf{x}_u u_1' + \mathbf{x}_v v_1'$ and $\mathbf{x}_2' = \mathbf{x}_u u_2' + \mathbf{x}_v v_2'$ holds

$$II(\mathbf{x}_1', \mathbf{x}_2') = 0$$

and thus

$$\begin{aligned}
(u_1', v_1') \begin{pmatrix} L & M \\ M & N \end{pmatrix} \begin{pmatrix} u_2' \\ v_2' \end{pmatrix} &= 0 \\
Lu_1'u_2' + Mu_1'v_2' + Mu_2'v_1' + Nv_1'v_2' &= 0 \qquad (3.7) \\
-\mathbf{x}_u \mathrm{N}_u u_1'u_2' - \mathbf{x}_u \mathrm{N}_v u_1'v_2' - \mathbf{x}_v \mathrm{N}_u u_2'v_1' - \mathbf{x}_v \mathrm{N}_v v_1'v_2' &= 0 \\
(\mathbf{x}_u u_1')(\mathrm{N}_u u_2') + (\mathbf{x}_v v_1')(\mathrm{N}_u u_2') + (\mathbf{x}_u u_1')(\mathrm{N}_v v_2') + (\mathbf{x}_v v_1')(\mathrm{N}_v v_2') &= 0 \\
(\mathbf{x}_u u_1' + \mathbf{x}_v v_1')(\mathrm{N}_u u_2' + \mathrm{N}_v v_2') &= 0
\end{aligned}$$

which can be written as

$$\mathbf{x}_1' \mathrm{N}_2' = 0 \text{ or because of symmetry as } \mathbf{x}_2' \mathrm{N}_1' = 0 \ . \qquad (3.8)$$

Equation (3.7) can also be interpreted as a differential equation for a family of curves $\mathcal{X}_2$ if a family of curves $\mathcal{X}_1$ is prescribed (or otherwise). If a family of curves $\mathcal{X}_1$ contains a curve $\mathbf{x}_1$ for which holds $II(\mathbf{x}_1', \mathbf{x}_1') = 0$ in a point p, then the tangent is self-conjugate there (it is an asymptotic direction), otherwise the families of curves $\mathcal{X}_1$ and $\mathcal{X}_2$ form a **conjugate curve network**. So, if there exists a conjugate curve network on a parametrized surface, it is possible to parametrize the surface via $\mathbf{x}(u, v)$ with $\mathbf{x}(u, v_0) \in \mathcal{X}_1$ for $v = v_0 = const$ and $\mathbf{x}(u_0, v) \in \mathcal{X}_2$ for $u = u_0 = const$.

For a curve on a surface it is of interest to examine the properties of a surface strip that touches the surface along this curve. A surface strip along a curve is a ruled surface so that for every point of the curve the corresponding ruling lies in the tangent space of the surface in this point. The length of each ruling shall be small enough to not intersect with the surface apart from this directrix.

Consider a one-parametric family of planes, defined via

$$\mathrm{N}(t)\mathbf{x} = d(t) \ . \qquad (3.9)$$

For now, omit the case where all planes share a common point (finite or at infinity). To compute the envelope differentiate with respect to t, thus

$$\mathrm{N}'(t)\mathbf{x} = d'(t) \qquad (3.10)$$

and hence the points $\mathbf{x}$ lie on a family of straight lines with directions parallel to $e(t) = \mathrm{N}(t) \wedge \mathrm{N}'(t)$. Demanding also that $\mathbf{x}$ fulfils

$$\mathrm{N}''(t)\mathbf{x} = d''(t) \qquad (3.11)$$

yields a curve $\mathbf{x}(t)$ for which (3.9), (3.10) and (3.11) holds. Applying (3.10) to the differential of (3.9), that is $\mathrm{N}'(t)\mathbf{x}(t) + \mathrm{N}(t)\mathbf{x}'(t) = d'(t)$, and (3.11) to the differential of (3.10), thus $\mathrm{N}''(t)\mathbf{x}(t) + \mathrm{N}'(t)\mathbf{x}'(t) = d''(t)$, these equations become

$$\begin{aligned}
\mathrm{N}(t)\mathbf{x}'(t) &= 0 \ , \\
\mathrm{N}'(t)\mathbf{x}'(t) &= 0 \ . \qquad (3.12)
\end{aligned}$$

This means that $\mathbf{x}'(t)$ is orthogonal to $\mathrm{N}(t)$ as well as to $\mathrm{N}'(t)$, hence

$$\mathbf{x}'(t) = \lambda e(t) \ . \tag{3.13}$$

Thus the tangents of $\mathbf{x}(t)$ are the rulings of the envelope of (3.9).

In the conical case, e.g. all planes share a common finite point p, equation (3.9) can be written as

$$\mathrm{N}(t)\mathbf{x} = \mathrm{N}(t)p$$

and thus $\mathrm{N}(t)(\mathbf{x} - p) = 0$ which leads to

$$\begin{aligned} \mathrm{N}'(t)(\mathbf{x} - p) = 0 \quad &\Rightarrow \quad (\mathbf{x} - p) = se(t) \\ &\Rightarrow \quad \mathbf{x} = p + se(t) \ . \end{aligned}$$

With the above one gets a curve $\mathbf{x}(u)$ for which holds

$$\begin{aligned} \mathrm{N}(t)(\mathbf{x}(u) - p) &= 0 \\ \mathrm{N}'(t)(\mathbf{x}(u) - p) &= 0 \\ \mathrm{N}'(t)\frac{\partial}{\partial s}\mathbf{x}(t) &= < \mathrm{N}'(t), \mathrm{N}(t) \wedge \mathrm{N}'(t) >= 0 \end{aligned} \tag{3.14}$$

which means that the lines through $\mathbf{x}(t)$ and p are the rulings of the strip.

For the cylindrical case see the previous section and examples of the surfaces are given in fig.(3.3).

Comparing (3.12) respectively (3.14) with (3.8), these calculations lead to the following theorem.

THEOREM 3.3 *The ruling directions of a surface strip that touches a surface along a curve are conjugated to the tangent directions of that curve.*

COROLLARY 3.4 *For a one-parametric family of curves $\mathcal{X}_1$, the conjugate curves $\mathcal{X}_2$ are given as the integral curves of the conjugated tangential directions.*

Additionally, for a surface that is parametrized as a conjugate curve network like described above, so $II(\mathbf{x}_u, \mathbf{x}_v) = 0$, it is obvious that $\mathbf{x}_u \mathrm{N}_v = \mathbf{x}_v \mathrm{N}_u = 0$ which leads to $M = 0$ in the second fundamental form and one can formulate corollary 3.5 because the converse holds too.

COROLLARY 3.5 *A surface is parametrized as a conjugate curve network if and only if $M = 0$ in the second fundamental form.*

With $M = 0$, which means $\mathbf{x}_{uv}$ is normal to $\mathbf{x}_u \wedge \mathbf{x}_v$, a parametrization $\mathbf{x}(u, v)$ is a conjugate net if for all parameters (u, v) holds $\mathbf{x}_{uv} = \mathbf{x}_{vu} \in \mathrm{span}\{\mathbf{x}_u, \mathbf{x}_v\}$ and motivates the discrete version below.

DEFINITION 3.6 *A mesh $\mathbf{x} : \mathbb{Z}^2 \to \mathbb{R}^3 : (i, j) \mapsto \mathbf{x}_{ij}$ is called discrete conjugate net if for all pairs (i, j) holds $\delta_i \delta_j \mathbf{x}_{ij} = \delta_j \delta_i \mathbf{x}_{ij} \in \mathrm{span}\{\delta_i \mathbf{x}_{ij}, \delta_j \mathbf{x}_{ij}\}$.*

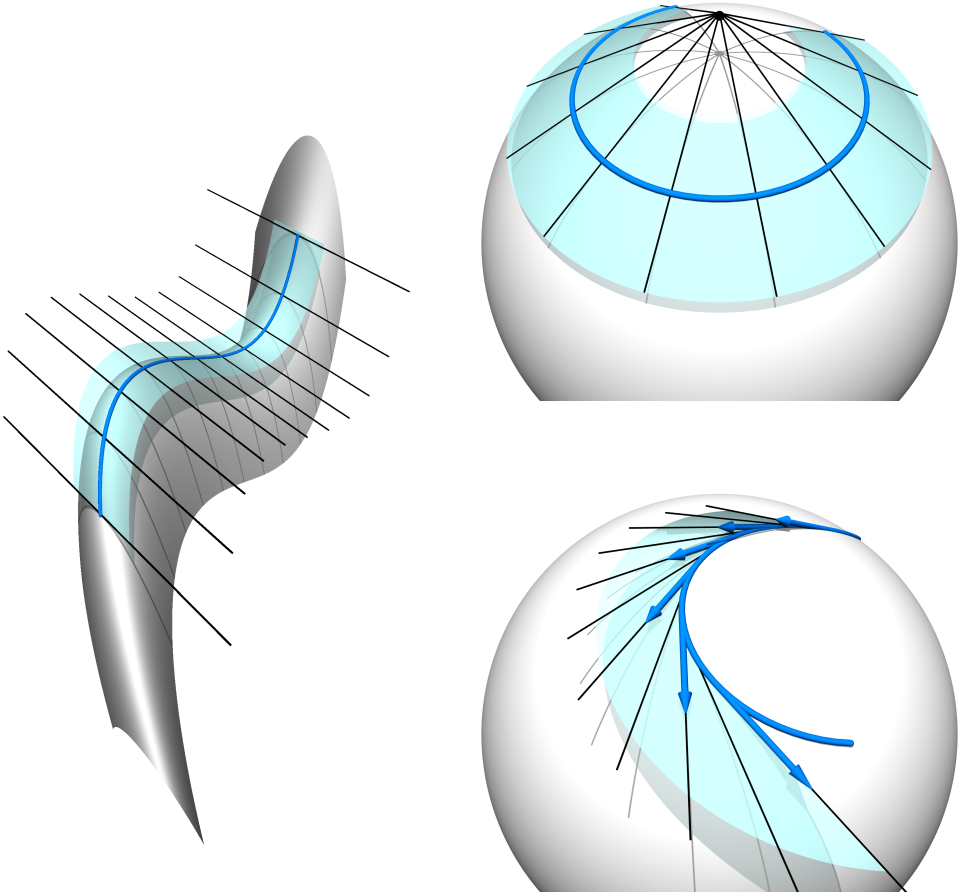


Figure 3.3: Examples of cylinder surface, cone surface and tangent surface.

So clearly, PQ-meshes form a conjugate net.

Recall the conjugate direction property from above, $\mathbf{x}_1' \mathrm{N}_2' = 0$. Instead of writing the curve as $\mathbf{x}_1(u_1(t), v_1(t))$ denote it as $c(t)$ then the above calculations can be applied to the surface curve. From theorem 3.3 immediately follows the subsequent corollary.

COROLLARY 3.7 *Given a curve $c(t)$ in a conjugate curve network $\mathbf{x}(u, v)$ then the tangents of the conjugated curves along $c(t)$ form a developable surface.*

Conjugate curve networks are defined for smooth surfaces like above. As for the applications in this thesis mostly discrete objects are treated, a discrete analogon is needed. The following motivates that planar quad-meshes can be seen as such. For a complete overview see [Sau70] which the subsequent follows very closely.

For a surface $\mathbf{x}(u,v)$, $\varepsilon > 0$ and parameter values $(u,v) \in \{u_i | u_i := u_0 + i\varepsilon\} \times \{v_j | v_j := v_0 + j\varepsilon\}$ the triangles $T = (\mathbf{x}(u,v), \mathbf{x}(u+\varepsilon, v), \mathbf{x}(u, v+\varepsilon))$ form the so-called **chord triangle network** and the quads that consist of T and the triangles $\hat{T} = (\mathbf{x}(u, v+\varepsilon), \mathbf{x}(u+\varepsilon, v), \mathbf{x}(u+\varepsilon, v+\varepsilon))$ the **chord quadrilateral network**, compare fig.(3.4).

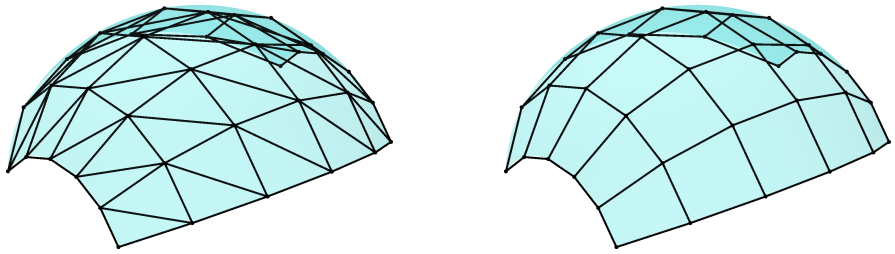


Figure 3.4: Example of chord triangle network respectively chord quadrilateral network.

One is interested in the angle between the triangles T and $\hat{T}$ which will be denoted α_ε, see fig.(3.5). Using the notation

$$\begin{aligned}
\mathbf{a}_\varepsilon(u,v) &= \mathbf{x}(u+\varepsilon, v) - \mathbf{x}(u,v) \ , & a_\varepsilon(u,v) &= ||\mathbf{a}_\varepsilon(u,v)||^2 \ , \\
\mathbf{b}_\varepsilon(u,v) &= \mathbf{x}(u, v+\varepsilon) - \mathbf{x}(u,v) \ , & b_\varepsilon(u,v) &= ||\mathbf{b}_\varepsilon(u,v)||^2 \ , \\
\mathbf{c}_\varepsilon(u,v) &= \mathbf{x}(u+\varepsilon, v) - \mathbf{x}(u, v+\varepsilon) \ , & c_\varepsilon(u,v) &= ||\mathbf{c}_\varepsilon(u,v)||^2 \ , \\
\mathbf{d}_\varepsilon(u,v) &= \mathbf{x}(u+\varepsilon, v+\varepsilon) - \mathbf{x}(u,v) \ , & d_\varepsilon(u,v) &= ||\mathbf{d}_\varepsilon(u,v)||^2
\end{aligned}$$

for the edges of the triangle and

$$\begin{aligned}
a(u,v) &= < \mathbf{x}_u(u,v), \mathbf{x}_u(u,v) >^{\frac{1}{2}} &&= \sqrt{E} \ , \\
b(u,v) &= < \mathbf{x}_v(u,v), \mathbf{x}_v(u,v) >^{\frac{1}{2}} &&= \sqrt{G} \ , \\
c(u,v) &= ||\mathbf{x}_u(u,v) - \mathbf{x}_v(u,v)|| &&= \sqrt{E+G-2F} \ , \\
d(u,v) &= < \mathbf{x}_{uv}(u,v), \mathbf{x}_{uv}(u,v) >^{\frac{1}{2}}
\end{aligned}$$

for the smooth counterparts, where E, F, G are the coefficients of the first fundamental form, one can approximate the discrete quantities for $\varepsilon \to 0$ as ($z = \{a, b, c, d\}$)

$$\begin{aligned}
z_\varepsilon(u,v) &= \varepsilon z(u,v) + \varepsilon^2 z'(u,v) + \mathcal{O}(\varepsilon^3) \ , \\
\alpha_\varepsilon(u,v) &= \varepsilon \alpha(u,v) + \varepsilon^2 \alpha'(u,v) + \mathcal{O}(\varepsilon^3) \ .
\end{aligned}$$

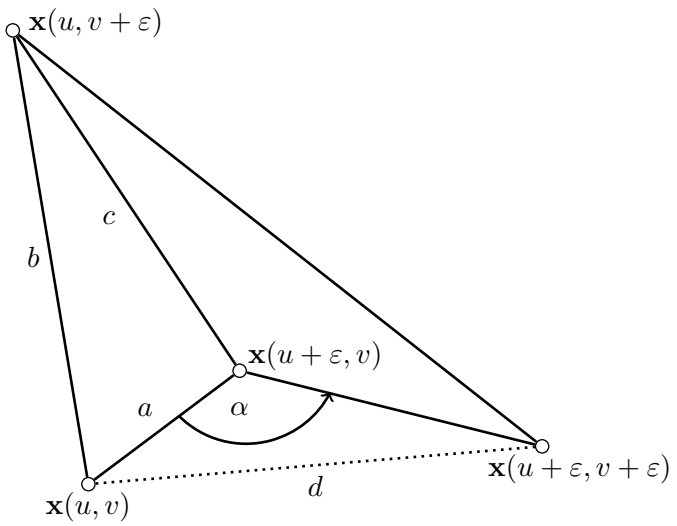


Figure 3.5: A chord quadrilateral, divided into two triangles with the angle they enclose.

Continuing in this manner, one can express the sides of $\hat{T}$ and all other angles inside the triangles. As with $\mathrm{Vol}_\varepsilon(u,v) =< \mathbf{d}_\varepsilon(u,v), \mathbf{b}_\varepsilon(u+\varepsilon, v) \wedge \mathbf{a}_\varepsilon(u, v+\varepsilon) >$, this is six times the volume of the tetrahedron spanned by the corresponding vertices,

$$\sin\alpha_\varepsilon(u,v) = \frac{c_\varepsilon(u,v)\mathrm{Vol}_\varepsilon(u,v)}{\mathrm{A}_\varepsilon(u,v)\hat{\mathrm{A}}_\varepsilon(u,v)} \tag{3.15}$$

where $\mathrm{A}_\varepsilon(u,v)$ is the area of the parallelogram spanned by $\mathbf{a}_\varepsilon(u,v)$ and $\mathbf{b}_\varepsilon(u,v)$,thus $\mathrm{A}_\varepsilon(u,v) = ||\mathbf{a}_\varepsilon(u,v) \wedge \mathbf{b}_\varepsilon(u,v)||$, and $\hat{\mathrm{A}}_\varepsilon(u,v) = ||\mathbf{a}_\varepsilon(u,v+\varepsilon) \wedge \mathbf{b}_\varepsilon(u+\varepsilon,v)||$ is the area of the corresponding parallelogram in $\hat{T}$, it can then be shown that

$$\alpha(u,v) = \frac{\sqrt{E-2F+G}}{\sqrt{EG-F^2}}M$$

with M being the coefficient of the second fundamental form, because the smooth version of (3.15) reads

$$\begin{aligned}
\sin\alpha(u,v) &= \frac{\sqrt{E+G-2F} < \mathbf{x}_u(u,v) \wedge \mathbf{x}_v(u,v), \mathbf{x}_{uv}(u,v) >}{\sqrt{EG-F^2}\sqrt{EG-F^2}} \\
&= \frac{\sqrt{E+G-2F} < \mathrm{N}(u,v), \mathbf{x}_{uv}(u,v) > \sqrt{EG-F^2}}{\sqrt{EG-F^2}\sqrt{EG-F^2}} \\
&= \frac{\sqrt{E-2F+G}}{\sqrt{EG-F^2}}M \ .
\end{aligned}$$

Using the Taylor expansion of the sine function, i.e. $\sin(x) = x - \frac{x^3}{3!} + \dots$ and therefore

$$\begin{aligned}\sin(x_\varepsilon) &= \sin(\varepsilon x + \varepsilon^2 x' + \mathcal{O}(\varepsilon^3)) \\ &= \varepsilon x + \varepsilon^2 x' + \mathcal{O}(\varepsilon^3) - \frac{(\varepsilon x + \varepsilon^2 x' + \mathcal{O}(\varepsilon^3))^3}{6} + \dots \\ &= \varepsilon x + \varepsilon^2 x' + \mathcal{O}(\varepsilon^3) \ ,\end{aligned}$$

one sees that the use of the sine function can be suppressed.

These computations yield for a surface $\mathbf{x}(u, v)$ which is parametrized as as conjugate curve network that the angles between the triangles, which form the chord quadrilaterals, are for $\varepsilon > 0$ given by

$$\begin{aligned}\alpha_\varepsilon(u,v) &= \varepsilon \frac{\sqrt{E - 2F + G}}{\sqrt{EG - F^2}} M + \varepsilon^2 \alpha'(u,v) + \mathcal{O}(\varepsilon^3) \\ &\overset{(3.5 \Rightarrow M=0)}{=} \varepsilon^2 \alpha'(u,v) + \mathcal{O}(\varepsilon^3) \\ &= \mathcal{O}(\varepsilon^2) \ .\end{aligned}$$

So for conjugate curve networks, the chord quadrilaterals are planar with an error of $\mathcal{O}(\varepsilon^2)$, thus the (almost) planar chord quadrilaterals of a conjugate curve network parametrization converge to that conjugate curve network. Conversely PQ-meshes are in general not chord quadrilaterals of a conjugate curve network parametrization, thus subdivision processes do not lead to conjugated curve networks in general.

This motivates the use of strips initialized with contours as boundary curves in the section 3.2.

3.1.3 Sampling Methods and Discretization

Until now, mostly smooth surfaces were of interest. Several statements on surfaces with continuously changing properties were given so far. For the rest of the thesis, discrete objects play a more important role.

Since discrete surfaces are of interest, a discrete method shall be given to compute the contour generators of an arbitrary surface. First, one must consider the possible light directions.

For semi-opaque surfaces, it was stated that $\Gamma(\overrightarrow{d}) = \Gamma(-\overrightarrow{d})$, this means that it is sufficient to consider the direction $\overrightarrow{d}$ or the direction $-\overrightarrow{d}$. Further, as the position of the light source is not relevant, the illumination direction can be chosen as an element of the unit sphere $\mathcal{S}^2$ without loss of generality.

Summed up, one gets all possible contour generators for semi-opaque surfaces by positioning light sources at O with illumination directions $\ell \in H^2 = \{(x, y, z) \in \mathcal{S}^2 | z \geq 0\}$. As it is not possible to compute the contour generators for every point in H^2, a good subset is needed. The approach is to approximate H^2 with a discrete set $I = I_H$ that has the form

$$I_H = \{(\cos(\phi_i)\sin(\psi_j), \sin(\phi_i)\sin(\psi_j), \sin(\psi_j))\} \ ,$$

where $\phi_i \in \{0, \frac{360}{n}, 2\frac{360}{n}, \ldots, (n-1)\frac{360}{n}\}$ and $\psi_j \in \{0, \frac{90}{m}, 2\frac{90}{m}, \ldots, (m-1)\frac{90}{m}\}$. Choosing $n, m \in \mathbb{N}$ big enough yields a good approximation of H^2. Another method would be to draw the parameters ϕ and ψ from a uniform random distribution. Also, as a hemisphere is not the best choice for every surface that shall be covered, other illumination direction sets can be used, for example the discrete subset

$$I_T = \{(\cos(\phi_i)\sin(\psi_j), \sin(\phi_i)\sin(\psi_j), \sin(\psi_j)) \,|0 \leq \phi_i < 360, -45 \leq \psi_j \leq 45\} \subset \mathcal{S}^2 \ . \tag{3.16}$$

This may seem unnecessary at this point, but makes sense as soon as tangential fields on the meshes, constructed from I, are integrated to levelsets in section 3.1.5.

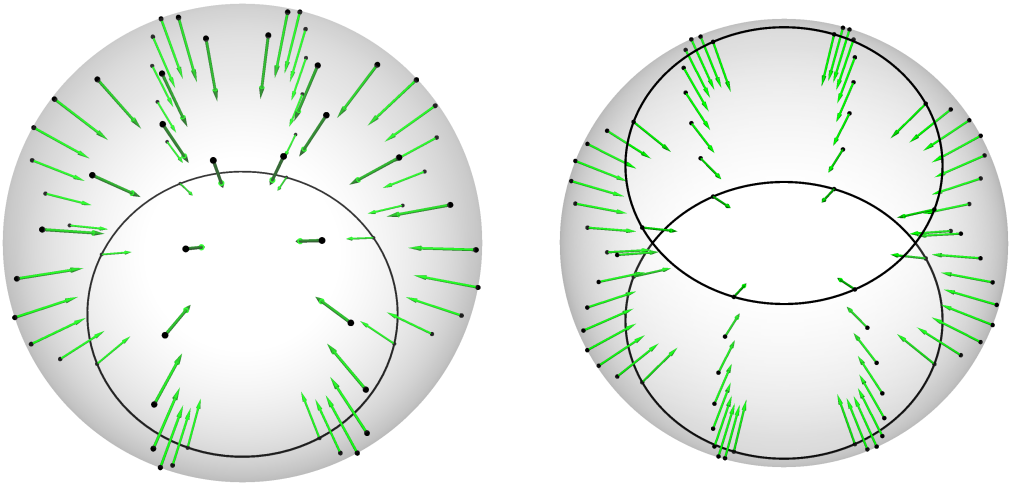


Figure 3.6: Example for I_H, I_T with illumination directions.

Given a mesh M and an illumination direction $\overrightarrow{d}$, a direct identification of points which lie in $\Gamma(\overrightarrow{d})$ using (3.1) is not effective because the normal field on M is discrete too, thus does not change continuously.

One approach could be to identify those edges of M for which one adjacent face is pointing towards the light source (viewed as located at infinity) and the other adjacent face is pointing away. Mathematically this means for neighboring faces f_1, f_2 with face normals $\overrightarrow{n}_1, \overrightarrow{n}_2$

$$(\ \overrightarrow{n}_1 \cdot \overrightarrow{d} \geq 0 \text{ and } \overrightarrow{n}_2 \cdot \overrightarrow{d} < 0\ ) \ \text{ or } \ (\ \overrightarrow{n}_1 \cdot \overrightarrow{d} \leq 0 \text{ and } \overrightarrow{n}_2 \cdot \overrightarrow{d} > 0\ ) \ .$$

The problem with this method is, that it heavily relies on the resolution of the mesh M, so another – less resolution-dependent – method is needed.
Instead of identifying edges of the mesh, one can identify contour lines on the mesh going through the triangles.

With proper normals at the vertices of the mesh, the argument for finding contour generators is similar to the method stated above. For a given mesh M and an illumination direction $\overrightarrow{d}$, consider all vertices $v_0, v_1, v_2 \in f$ for every face $f \in F$ and the corresponding normals $\overrightarrow{n}_0, \overrightarrow{n}_1, \overrightarrow{n}_2$. If for two of these three normals $\overrightarrow{n}_i, \overrightarrow{n}_j$, $i \neq j$, holds

$$\overrightarrow{n}_i \cdot \overrightarrow{d} < 0 \quad \text{and} \quad \overrightarrow{n}_j \cdot \overrightarrow{d} > 0$$

then there is a point v^* on the edge connecting v_i and v_j for which

$$\overrightarrow{n}^* \cdot \overrightarrow{d} = 0 \ , \tag{3.17}$$

where $\overrightarrow{n}^*$ is the interpolated normal at v^*. To get v^* respectively $\overrightarrow{n}^*$ a simple linear interpolation is applied, which means

$$\begin{aligned} v^* &= v_i(1-t) + v_j t \\ \overrightarrow{n}^* &= \overrightarrow{n}_i(1-t) + \overrightarrow{n}_j t \end{aligned}$$

with t given as the solution of

$$\begin{aligned} 0 &= (\overrightarrow{n}_i \cdot \overrightarrow{d})(1-t) + (\overrightarrow{n}_j \cdot \overrightarrow{d})t \ , \text{ leading to} \\ t &= \frac{\overrightarrow{n}_i \cdot \overrightarrow{d}}{(\overrightarrow{n}_i \cdot \overrightarrow{d}) - (\overrightarrow{n}_j \cdot \overrightarrow{d})} \ . \end{aligned}$$

In case of a second pair of vertices of the same face fulfilling that condition, one has a root of (3.17) for two different edges and can connect them with a straight line through the face. This line is a local linear approximation of the contour generator and proceeding with this method for the other faces and connecting the lines in adjacent faces if possible yields a discrete contour generator for direction $\overrightarrow{d}$ (that may consist of several components).

3.1.4 Evaluation and Extraction of Samples

Not every contour line is suitable as a directrix of a cylindrical strip. To guarantee a good covering, the angle between directrix and rulings should be near 90°. That is, the illumination direction should be as perpendicular to the contour generator as possible. This restriction provides a good visual effect but above all it is important to avoid singularities (an angle of zero yields a locally degenerate strip with width zero) that impede the method from working correctly.

Now assume M is a mesh. The contour generators do not consist of curves but of polygonal lines and as the normals in the vertices are linearly interpolated, using the apparent contour to identify properties of the contour generator would not work directly. Instead of explicitly naming contour points with special features and treating them appropriately, the following method tries to avoid such contours.

The idea is, given the contour generators for every sample $\overrightarrow{d} \in I$, discard those with components which are “too parallel” to the illumination direction. As every

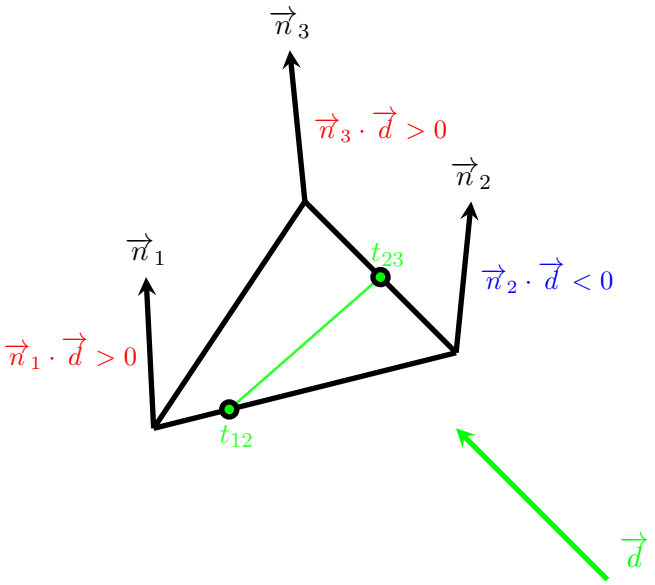


Figure 3.7: Procedure of root finding.

component $c \in \Gamma(\vec{d})$ is a polyline, c can be written as an edge-list $c = (e_1, e_2, \ldots, e_n)$ and an edge e_i is admissible if the normalized direction vector $\vec{e_i}$ of e_i encloses an angle near enough to 90° or equally

$$\vec{e_i} \cdot \vec{d} \in (-\varepsilon_\alpha, \varepsilon_\alpha) \tag{3.18}$$

where ε_α denotes an angle-threshold.
If a contour generator admits not solely but mostly admissible edges it can be regarded sufficient good. Therefore the following definition makes sense as a requisite for not discarding contour generators.

DEFINITION 3.8 *Let $\Gamma(\vec{d})$ be a contour generator on a surface S, then a component $c \in \Gamma(\vec{d})$, $c = (e_1, e_2, \ldots, e_n)$ is admissible if for a given angle-threshold ε_α and a length-threshold ε_δ holds*

$$\frac{1}{n}\#\{e_i | e_i \text{ satisfies (3.18)}\} > 1 - \varepsilon_\delta \ . \tag{3.19}$$

The contour generator $\Gamma(\vec{d})$ is admissible if at least one component is admissible.

For the above definition note, that if only a few edges are allowed to be not admissible, the case of self-conjugate directions does not occur for meshes with sufficient resolution because then the angles between illumination direction and subsequent edges does not change that fast. Therefore, with increasing resolution, more edges are not

admissible around self-conjugate edges and hence the component is disqualified by (3.19).

Removing those contour generators and their corresponding samples from I which are not admissible, one gets a more coarse sampling $\tilde{I} \subset I$. Also components that are not admissible are removed from admissible contour generators, getting for every $\overrightarrow{d} \in \tilde{I}$ a (maybe partial) contour generator $\tilde{\Gamma} \subset \Gamma$ – consisting of admissible components only.

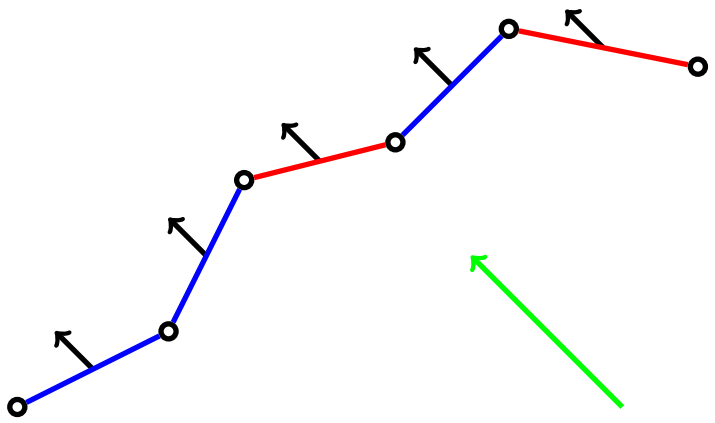


Figure 3.8: Scheme of admissible (blue) and not admissible (red) edges with $\varepsilon_\alpha = 0.5$. With $\varepsilon_\delta = 0.5$ this polyline would be admissible.

3.1.5 Computation of Initializing Curves

Consider $\tilde{I}$, a one-parametric family Φ of contours shall be extracted, which is done by finding a polyline connecting those elements of $\tilde{I}$ for which the corresponding contour generators $\tilde{\Gamma}$ fulfill the needed criteria of quality.

To do this, one needs some structure on $\tilde{I}$. Because of the fact that every $\overrightarrow{d} \in \tilde{I}$ is also an element of $\mathcal{S}^2$, the boundary of the convex hull of $\tilde{I}$ contains all elements of $\tilde{I}$, so instead of regarding $\tilde{I}$ as a set, consider it as a triangle mesh that can be found by computing the convex hull of $\tilde{I}$, see [BDH96].

In order to get a good covering of the mesh, the contour generators must be evenly distributed or rather a contour should be nearly a translate of its neighboring contours. Thus, given $\tilde{\Gamma}$ one needs to find a step direction which tells the illumination direction where to move next in order to generate such a contour, producing a curve consisting of points that represent illumination directions, which in turn generate a covering in the end. These step directions, which are to be found, correspond to the tangents of the curve of illumination directions.

If one denotes the contour as an element of $\mathcal{X}_1$, a good translate is found by moving along the curves in $\mathcal{X}_2$. The method is motivated by the idea, that the illumination direction is conjugate to the contour. A set of illumination directions is already sam-

pled, so for every of those the question is where to move next from them, in order to get a one-parametric family of illumination direction curves.

As $\overrightarrow{d} \in \mathcal{S}^2$, the wanted step direction $\overrightarrow{x} \in T_d\mathcal{S}^2$ is perpendicular to $\overrightarrow{d}$ and for a good offset property it clearly also has to be perpendicular to the contour. Let $c = (e_1, \ldots, e_k)$ be the best component of $\tilde{\Gamma}(\overrightarrow{d})$ with respect to (3.19). At this point, the component already (mostly) consists of edges enclosing an angle of nearly $90°$ with $\overrightarrow{d}$. Denoting the summed up mesh normals along the component c with

$$\overrightarrow{n}_c = \frac{1}{k} \sum_{i=0}^{k} \overrightarrow{n}_i$$

and its projection onto the tangential plane of $\tilde{I}$ in $\overrightarrow{d}$ with $\overrightarrow{x}$, one has a tangential vector field on $\tilde{I}$.

This vector field is seen as the gradient field of a scalar piecewise-linear function $f : \tilde{I} \to \mathbb{R}$. Also, it may exhibit the problem of disoriented directions in the sense of their signs, because for a solution $\overrightarrow{x}$ one gets $(-\overrightarrow{x})$ as a solution as well, thus the vector field is combed, i.e. vectors may exchange their sign depending on their neighbors.

The levelsets of this function then serve as the wanted one-parametric family of curves. The subsequent paragraphs deal with this problem following [Pin] by investigating the properties of such a function.

Assume a triangle $\Delta \subset \mathbb{R}^3$ with vertices v_i, v_j, v_k, halfedges e_{ij}, e_{jk}, e_{ki} oriented counter-clockwise and function values $f_\ell := f(v_\ell)$, $\ell \in \{i, j, k\}$. As f is linear on Δ the gradient ∇f is constant on Δ and for the restriction of f to a halfedge, for example e_{ij}, holds

$$f_{\restriction e_{ij}}(t) = (1-t)f_i + tf_j \quad , \; t \in [0,1]$$

and therefore

$$\begin{aligned} < \nabla f, \overrightarrow{e_{ij}} > &= \frac{\partial f}{\partial \overrightarrow{e_{ij}}} \\ &= {f_{\restriction e_{ij}}}'(t) \\ &= f_j - f_i \; . \end{aligned}$$

The face normal is given by $\mathrm{N} = \frac{\overrightarrow{e_{ij}} \wedge \overrightarrow{e_{jk}}}{2\mathrm{Area}}$, with $\mathrm{Area} = \frac{1}{2}||\overrightarrow{e_{ij}} \wedge \overrightarrow{e_{jk}}||$, all statements analogous for the other halfedges. Consider

$$\overrightarrow{g} := \frac{1}{2\mathrm{Area}} \mathrm{N} \wedge (f_k \overrightarrow{e_{ij}} + f_i \overrightarrow{e_{jk}} + f_j \overrightarrow{e_{ki}})$$

which behaves like the gradient of f, to wit

$$\begin{aligned}
< \overrightarrow{g}, \overrightarrow{e_{ij}} > &= \frac{1}{2\text{Area}} < \text{N} \wedge (f_k \overrightarrow{e_{ij}} + f_i \overrightarrow{e_{jk}} + f_j \overrightarrow{e_{ki}}), \overrightarrow{e_{ij}} > \\
&= \frac{1}{2\text{Area}} < \text{N} \wedge f_i \overrightarrow{e_{jk}} + \text{N} \wedge f_j \overrightarrow{e_{ki}}, \overrightarrow{e_{ij}} > \\
&= \frac{1}{2\text{Area}} (f_i < \text{N} \wedge \overrightarrow{e_{jk}}, \overrightarrow{e_{ij}} > + f_j < \text{N} \wedge \overrightarrow{e_{ki}}, \overrightarrow{e_{ij}} >) \\
&= \frac{1}{2\text{Area}} (f_i < \text{N}, \overrightarrow{e_{jk}} \wedge \overrightarrow{e_{ij}} > + f_j < \text{N}, \overrightarrow{e_{ki}} \wedge \overrightarrow{e_{ij}} >) \\
&= \frac{1}{2\text{Area}} (-f_i < \text{N}, \overrightarrow{e_{ij}} \wedge \overrightarrow{e_{jk}} > + f_j < \text{N}, \overrightarrow{e_{ki}} \wedge \overrightarrow{e_{ij}} >) \\
&= f_j - f_i \ .
\end{aligned}$$

Thus $\nabla f = \overrightarrow{g}$ and using the gradient rotated by 90° into the plane of the triangle the formula is

$$\text{N} \wedge \nabla f = -\frac{1}{2\text{Area}} (f_k \overrightarrow{e_{ij}} + f_i \overrightarrow{e_{jk}} + f_j \overrightarrow{e_{ki}}) \ . \tag{3.20}$$

As mentioned earlier, $\tilde{I}$ is a mesh approximating a subset of $\mathcal{S}^2$ where every vertex is identified with an illumination direction and also holds the corresponding contour generator. Further a vector $\overrightarrow{x}$ pointing into the direction of another illumination direction that guarantees a good covering is attached to every vertex. But because of the above, one needs such directions on the faces of the mesh. So for every triangle $\Delta \in \tilde{I}$, the directions $\overrightarrow{x}$ of its vertices are averaged to - here viewed as such - a gradient on the face and attached to it.

This means for vertices $v_i, v_j, v_k \in \Delta$ with directions $\overrightarrow{x}_i, \overrightarrow{x}_j, \overrightarrow{x}_k$, respectively, and halfedges e_{ij}, e_{jk}, e_{ki}, one has

$$\nabla f_{\restriction \Delta} = \frac{1}{3} (\overrightarrow{x}_i + \overrightarrow{x}_j + \overrightarrow{x}_k) \ .$$

Knwowing $\nabla f_{\restriction \Delta}$ implies the knowledge of the left-hand-side of equation (3.20) and thus yields a linear equation system for the unknown values f_ℓ, that is the sparse system

$$\underbrace{\begin{pmatrix} E_{\Delta_1} \\ \hline E_{\Delta_2} \\ \hline \vdots \\ \hline E_{\Delta_m} \end{pmatrix}}_{:= \text{E}} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} \text{N} \wedge \nabla f_{\restriction \Delta_1} \\ \hline \text{N} \wedge \nabla f_{\restriction \Delta_2} \\ \hline \vdots \\ \hline \text{N} \wedge \nabla f_{\restriction \Delta_m} \end{pmatrix} \ ,$$

where $n = |V|, m = |F|$ and $\text{E} \in \mathbb{R}^{n \times 3m}$ with entries $E_{\Delta_\ell} \in \mathbb{R}^{3 \times n}$,

$$E_{\Delta_\ell} = -\tfrac{1}{2\text{Area}(\Delta_\ell)} (0 \ \dots \ 0 \ \underset{\substack{\uparrow \\ i}}{\overrightarrow{e_{jk}}} \ 0 \ \dots \ 0 \ \underset{\substack{\uparrow \\ j}}{\overrightarrow{e_{ki}}} \ 0 \ \dots \ 0 \ \underset{\substack{\uparrow \\ k}}{\overrightarrow{e_{ij}}} \ 0 \ \dots \ 0)$$

for $\Delta_\ell = (v_i, v_j, v_k)$ with corresponding halfedges.

Solving the sparse system yields function values f_i at the vertices v_i. As mentioned earlier, the gradient of f in a vertex always points into the direction of the contour generator that guarantees a good covering with the current one. However, for the computation above the gradient was rotated about 90° which means that it does not point into the wanted direction but is orthogonal to it. As well known the gradient in a point is orthogonal to the levelset through this point, thus the levelsets that correspond to the funtion values f_i are exactly the desired curves.

For practical reasons, a prescribed number of levelsets K are computed. For this task the minimum and maximum function values f_{min} and f_{max}, respectively, are taken and the levelsets with function values $f = f_{min} + j\frac{f_{max}-f_{min}}{K-1}$, $j = 0, \ldots, K-1$, are computed.

Only one levelset is needed for covering, hence a choice must be made. The question which would be the best cannot be answered definitely, because it depends on the input and illumination meshes as well as on personal preferences.

An example of the initialization method is given in fig.(3.9). The initial set of illumination directions is displayed as a mesh top left and the final set of illumination directions after removing not admissible ones below, also showing the computed levelsets with the one used for the covering of the mesh on the right side emphasized in red. With a sampling size of 64 directions and 2176 mesh vertices, the computation of all contour generators takes 750 milliseconds, the initialization method as a whole takes 1160 milliseconds, thus only a bit more than a second.

3.2 Approximation

The chosen levelset from the last section yields a one-parametric set of illumination directions and corresponding contour generators which are used for the surface approximation explained in the following.

Given the contours from the last section, one can now start to use them for covering the given surface with cylindrical strips. As a representation of these strips, tensor product B-Splines are used. It is already known from previous statements, that the contours together with those curves that have the corresponding illumination directions as tangential directions form a conjugate network on the surface, thus it is convenient to set up the cylindrical surfaces with a contour as directrix and the illumination direction as the ruling direction.

For practical reasons, the question of ordering the contour generators arises. As contour generators usually consist of more than one component, all components must be arranged in a manner that guarantees a useful surface initialization. In the case of a smooth closed surface and parallel illumination, this question would be easier to answer as every possible normal direction must occur and thus for two illumination directions $\overrightarrow{d}_i$, $\overrightarrow{d}_j$ there is always a normal $\overrightarrow{n}$ so that $\overrightarrow{n}$ is parallel to $\overrightarrow{d}_i \wedge \overrightarrow{d}_j$ which yields an intersection point $p = \Gamma(\overrightarrow{d}_i) \cap \Gamma(\overrightarrow{d}_j)$ with normal $\overrightarrow{n}$.

In some cases, when problematic components (cusps, lips, ...) are already removed,

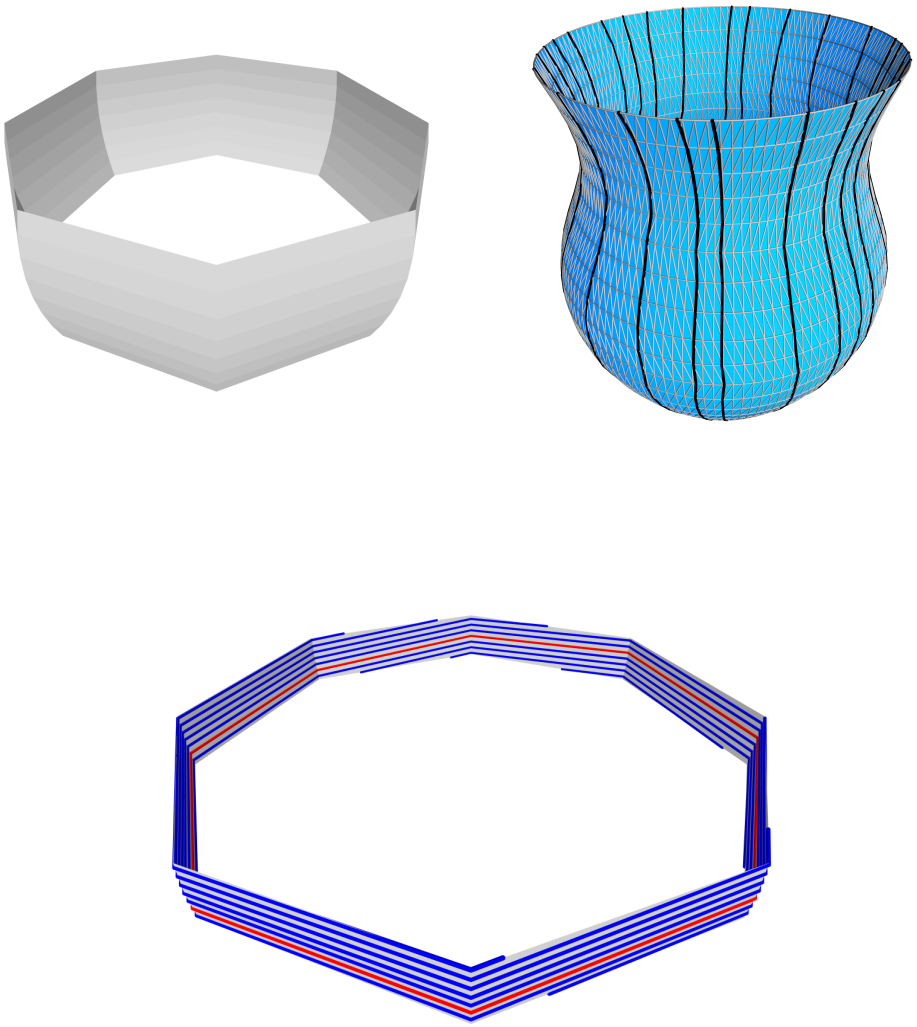


Figure 3.9: Example of the initialization method.

thus having a covering that resembles a conjugate curve network, and a sequential arrangement which makes sense exists, one can use the illumination direction to move from one contour to the next one.

Take a point p_0 on a component c_0 of a contour generator and append the corresponding illumination direction $\overrightarrow{d}_0$. Construct the ray with direction given as the projection of $\overrightarrow{d}_0$ on the mesh through p_0 which lies in the same triangle as p_0 and intersect it with the edges of this triangle giving a new point p_0'. Again, construct a ray with direction given as the projection of $\overrightarrow{d}_0$ onto the next triangle through p_0' and intersect it with the next edges yielding p_0''. This procedure is applied until the

ray intersects with another contour c_1 inside a triangle yielding the point p_1 for which the procedure is repeated using $\overrightarrow{d}_1$ as new direction. Iterating until every contour occurred (maybe this method leads to the boundary of the mesh, before every component occurred, then start with the next unused component in line), one has a list of points $(p_0, p_1, \ldots, p_N)$ which can be used to order the corresponding components.

Constructing cylinder strips given the contours as directions in the manner of (2.21), the main step is to change this coarse surface approximation in a way that improves the approximation as well as it preserves the cylindrical property which is fulfilled when using the mentioned construction. An alternative is to take of two neighboring contours as the boundary curves of a strip which then is not cylindrical in general and use an optimization which enforces the cylindrical property. This idea is motivated by the already treated chord quadrilaterals and their property to be nearly planar. The following optimization problem tackles both these tasks similar to [Pot+08].

The target functional shall guarantee that the strips are as close to the original mesh M as possible, thus for sample points $\{x_k\}_k$ of the B-Spline surface $\mathbf{x}(u, v)$

$$f_{close} = \sum_k \|x_k - \pi_M(x_k)\|^2 \tag{3.21}$$

provides a closeness term where $\pi_M(x)$ denotes the projection on M and

$$f_{fair} = \int \|\mathbf{x}_{uu}\|^2 \mathrm{d}u \tag{3.22}$$

takes care of the fairness. The first one pushes the initial strip towards the mesh by minimizing the distance of sample points on the strip to the projections of these points onto the mesh. The fairness functional smooths the strip by minimizing the bending energy of the strip which helps suppressing noise along the directrix which results in a higher visual smoothness.

Using these two functionals as a basis for the approximation, one can add several others which help optimize into the desired form. So clearly, one first needs a functional to ensure the cylindrical property.

There are two possibilities of doing this, using a cylindrical constraint or a cylindrical penalty. In both cases, the rulings are required to be parallel, so an exemplary direction can bet stated towards which they are forced. Let $\overrightarrow{\ell_i}$ be this direction for a strip $\mathbf{x}_i$. The linear constraint $\mathrm{A}\overrightarrow{x} = \overrightarrow{b}$ has the control points of the strip as variable vector and the matrix A is chosen so that the rows $3j, 3j+1, 3j+2$ of the left-hand side are equal to $\mathbf{b}_{ij} - \mathbf{a}_{ij}$ and the vector $\overrightarrow{b}$ consists of entries of the direction $\overrightarrow{\ell_i}$, yielding

$$g_{cyl} = \mathrm{A}\overrightarrow{x} - \overrightarrow{b} \ .$$

Similarly,

$$f_{cyl} \quad = \quad \sum_j \left\| \frac{b_i(u_j) - a_i(u_j)}{\|b_i(u_j) - a_i(u_j)\|} - \overrightarrow{\ell}_i \right\|^2 \tag{3.23}$$

provides the cylinder property penalty, e.g. rulings parallel to $\overrightarrow{\ell_i}$, see fig.(3.10).

The idea behind the functional f_{cyl} is to minimize the difference between the normalized ruling direction $\frac{b_i(u_j)-a_i(u_j)}{\|b_i(u_j)-a_i(u_j)\|}$ given at a parameter value u_j and the prescribed target direction $\overrightarrow{\ell_i}$. The target direction can be chosen as the mean value of all ruling directions and should be updated in every optimization iteration.

Further functionals which may be helpful are $f_{boundary}$ and f_{gap} which extend f_{close} to the boundary curve of the surface and minimize the distance between two succeeding strips, respectively. So $f_{boundary}$ is set as

$$f_{boundary} = \sum_k \|x_k - \pi_{\partial M}(x_k)\|^2$$

for sampled points x_k at the surface boundary and

$$f_{gap} = \sum_k \operatorname{dist}(b_i(x_k), a_{i+1}(\tilde{x}_k))^2$$

describes the gap penalty between two strips. The use of parameters x_k and $\tilde{x}_k$ is a shortcut to denote the distance between sample points of a boundary curve and their footpoints on the opposite boundary curve.

To enforce the cylindrical property even more, it may be useful to also apply a functional which helps to transform the given strip into a developable surface, i.e. a strip with planar faces but without the parallelism of the rulings. Therefore the tangential planes along the rulings need to agree, which is enforced by minimizing the length of the projection of the ruling direction $\overrightarrow{r}_j = \frac{b_i(u_j)-a_i(u_j)}{\|b_i(u_j)-a_i(u_j)\|}$ onto the normal direction $\overrightarrow{n}_j$ given as the cross product of the normalized directions of the diagonals

$$\begin{aligned} d_j^1 &= a_i(u_j) \vee \left(b_i(u_j) + \frac{\|\|b_i(u_j) - a_i(u_j)\|}{\|b_i'(u_j)\|} b_i'(u_j) \right) , \\ d_j^2 &= b_i(u_j) \vee \left(a_i(u_j) + \frac{\|\|b_i(u_j) - a_i(u_j)\|}{\|a_i'(u_j)\|} a_i'(u_j) \right) \end{aligned}$$

resulting in

$$f_{dev} = \sum_j < \overrightarrow{n}_j, \overrightarrow{r}_j >^2 \ .$$

Finally, for the visual quality of the approximation

$$f_{width} = \sum_j \left(||b_i(u_j) - a_i(u_j)|| - ||b_{i+1}(u_j) - a_{i+1}(u_j)|| \right)^2$$

provides a functional which ensures strips which are similar in width and

$$f_{dist} = \sum_j \|\pi(m_i(u_j)) - M_i(u_j)\|^2$$

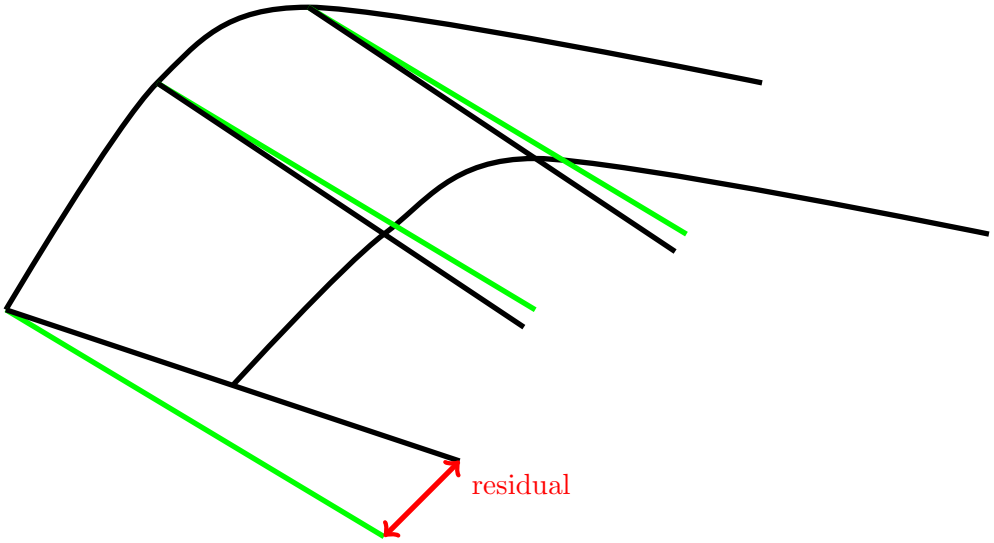


Figure 3.10: Idea behind f_{cyl}.

comes in handy when the initial strips are not distributed evenly enough. The used functions are $m_i(u) = \frac{a_i(u)+b_i(u)}{2}$ and $M_i(u) = \frac{m_{i-1}(u)+m_{i+1}(u)}{2}$, with $\pi(m_i(u))$ being the projection of $m_i(u)$ onto the line connecting $m_{i-1}(u)$ with $m_{i+1}(u)$.

Since there are different functionals which play a role for the optimization, they may interfere with each other, hence priorities should be set. This is achieved by weighting the individual functionals before putting them together into a single objective function, so for example

$$f_{basic} = w_{close} f_{close} + w_{fair} f_{fair}$$

with positive weights w_{close}, w_{fair} is used as the basic objective function,

$$f_{extended} = w_{boundary} f_{boundary} + w_{gap} f_{gap} + w_{width} f_{width} + w_{dist} f_{dist} + w_{dev} f_{dev}$$

then extends it. So the constrained optimization problem is given by

$$\min \; f_{basic} + f_{extended} \qquad \text{s.t.} \qquad g_{cyl} = \overrightarrow{0}$$

and the penalty optimization has the form

$$\min \; f_{basic} + f_{extended} + w_{cyl} f_{cyl} \; .$$

Depending on the initialization of the strips and the current state of the approximation, the type of optimization problem as well as the weights should be set appropriately. Below in section 3.3, some examples are given.

3.3 Examples and Results

In order to outline the quality of the rationalization, the examples use color-coded strips. Measuring the angles between consecutive rulings sampled along the strip, a maximum angle α of more than 10° leads to a black strip, similarly

$$\begin{aligned} \alpha \in (5.0^\circ, 10.0^\circ] &\;\hat{=}\; \text{red} , \\ \alpha \in (2.0^\circ,\ 5.0^\circ] &\;\hat{=}\; \text{blue} , \\ \alpha \in (1.0^\circ,\ 2.0^\circ] &\;\hat{=}\; \text{green} , \\ \alpha \in (0.5^\circ,\ 1.0^\circ] &\;\hat{=}\; \text{light green} \end{aligned}$$

and $\alpha \leq 0.5^\circ$ yields a white strip.

While the mesh for the first example was solely created to test the method, starting with the second example, the used meshes come from external sources to provide a richer variety of test cases.

Example 1

Similarly to the example of the initialization method using a vase mesh, a simple object is considered first. In fig.(3.11), the strips are initialized using two neighboring contours. Width and distribution of the strips are not ideal, but good enough to correct them using higher weights for the respective functionals in the beginning. Afterwards, high values for developability and cylindrical weights, together with positive weights for boundary approximation and strip gaps, yield the final covering.

In the result, the deviation between subsequent drawn rulings is less than 0.5° and the maximum deviation along the strip, measured as the sum of the consecutive unsigned angles between subsequent rulings is less than 1.3°.

Example 2

In order to cover the given mesh, shown in fig.(3.12), the strips were initialized and subdivided to control the initial strip width. Again, width and distribution are enforced before the focus lies on the other functionals. The angles between consecutive rulings are less than 0.3°, summed up along the strips, none has a deviation of more than 3°, using a cylinder penalty. For the final covering, a cylinder constraint was used to get perfect cylinders.

The upper part of the surface does not hold any contour generators. This is a result of the threshold settings, because the faces at the sides needed different values. Nevertheless, initial strips emerge from connecting contours on the edges between the sides and the upper part.

Examples 3

This example (top of Lilium Tower, Warsaw, by Zaha Hadid Architects) shows the capabilities and limitations of the presented method by means of approximating a

clearly non-suited surface in fig.(3.13). Here, the initialization algorithm only yields a very coarse set of contours, thus the initial covering is far off an optimum. Higher values for closeness and gap weights allow a first usable layout which then is optimized further using the typical steps.

Two different final coverings are given. As the surface does not allow a good overall covering with cylinder strips, a priority has to be chosen. Close-ups are given in fig.(3.14). Focus on closeness gives a covering that resembles the underlying mesh shape better, but gaps between the strips have to be treated outside the method in addition. Focus on continuity closes the gaps between the strips, but much of the shape is lost.

For the final coverings, the cylinder property was enforced using the cylinder constraint, thus leading to a deviation of nearly zero (all values $< 10^{-5}$ °).

Examples 4

Even if it is not always possible to get a set of contours that allows a covering of the whole surface, at least quite a bit can be covered. In fig.(3.15), large parts of the mesh are covered initially. Obviously, the proposed method for rearranging the contours does not work in this case, because no meaningful order can be defined, so for this example all functionals which involve more than one strip at a time must be set up manually. Using a cylinder penalty with gap and width functionals, the strips are aligned before a cylinder constraint guarantees a result consisting of perfect cylinders.

3.4 Discussion and Conclusions

This chapter first introduced an initialization method in 3.1 which can be used to generate a set of contour lines that are useful for surface approximation in the subsequent section 3.2.

The quality of the method highly depends on two properties. First, on the type of mesh which shall be covered, i.e. the curvature and the structure. This raises the question, for which input meshes the method should even be considered and is treated in the paragraphs below.

Second, it depends on the resolution of the mesh. If there are too few faces, the local geometry of the surface cannot be reflected correctly by the normal computation scheme and the linear interpolation of the vertex normals in order to compute the contour generators. High resolutions allow lower angle- and length-thresholds, which leads to a better quality of the contours and, because there can only be one contour line through one face, it also results in a higher covering density, but the backsides of high resolution meshes are increased computational time and storage.

In the examples different types of meshes were considered to show the potential of the method. The initialization is always applied using exactly one curve of illumination directions as described in section 3.1.5. In example 2, this leads to a blank area on top of the mesh in favor of a better covering on the sides, which is an outcome of the

choices of thresholds. Therefore, it may be of use for future implementations to use more than one illumination curve, possibly generated using different thresholds.

Another possibility for future applications of this method is the exchange of the mesh which shall be covered by another mesh which approximates the actual one but exhibits nicer features with respect to the initialization algorithm, thus resulting in a better initial covering. Using this initial covering but approximating the original mesh with it may solve the rationalization problem better than working only with the original mesh. Needless to say, this would require additional work by creating a second mesh with adapted geometry, which is not always possible. Also, segmentation of the input mesh and local coverage could lead to a more practical method in case of complex geometry and topology.

Examining example 4, one sees that it is even possible to (nearly) cover meshes with a more complicated geometry. This is, because the mesh can be deconstructed into parts which are clearly easy to approximate using cylinder strips.

On the other side, example 3 exhibits a geometry that poses more problems. Even segmentation would yield parts which are difficult to approximate with cylinders, although better sets of contours could be extracted.

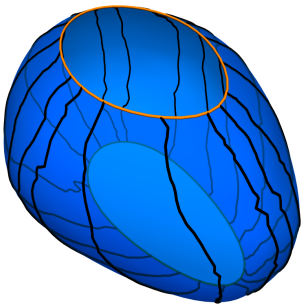
So, to conclude this discussion, a major limitation shall be stated. Particularly when considering the fourth example, one may expect contour generators of the form shown in fig.(3.2), thus intersecting contour generators resulting from one illumination direction. These cases cannot occur because of the piece-wise linear interpolation of the contour generators in each triangle, allowing only one line per face. Hence, contrary to the smooth case, there are no critical points where it is necessary to split the contour lines and complicate the initial strip alignment, but this also means loss of accuracy.

Nevertheless, the optimization step itself seems to produce good results using the approach of first concentrating on width and gaps (and maybe distribution) with smaller values for cylinder and developability weights and then optimizing towards developable and cylindrical strips which also minimize the distance to the boundary of the mesh. Depending on the mesh, for the final covering the cylinder penalty may be exchanged for a cylinder constraint resulting in nearly perfect cylinders.

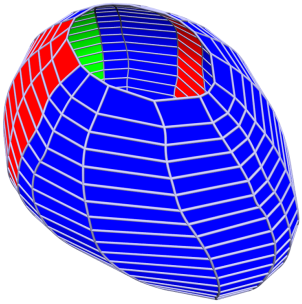
Some key figures read

Ex.	$\#I_M$	*Thresh.*		*Approx. Err.*		*Gap Err.*		*Cyl. Err.*		
		ε_α	ε_δ	*avg*	*max*	*avg*	*max*	*cons*	*avg*	*len*
1	64	.45	.45	0.28%	0.96%	.028%	.029%	.56°	.06°	1.27°
2	160	.55	.45	0.16%	2.75%	.011%	.012%	.27°	.12°	2.91°
3.1	64	.60	.35	0.92%	5.06%	.125%	.140%	.00°	.00°	0.00°
3.2	64	.60	.35	2.23%	9.17%	.043%	.048%	.00°	.00°	0.00°
4	256	.60	.45	0.77%	5.68%	.010%	.035%	.00°	.00°	0.00°

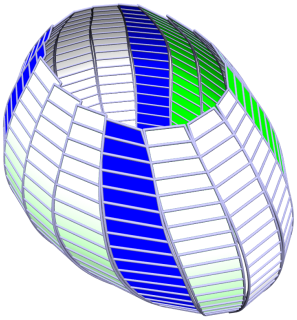
with the approximation error measured with respect to the diagonal of the bounding box of the underlying mesh using samples. The cylindrical error is measured as explained in the examples.



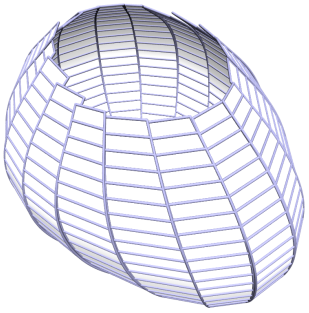
Initial mesh.



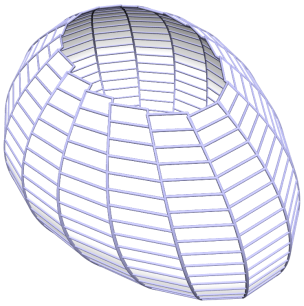
Initial covering of the mesh.



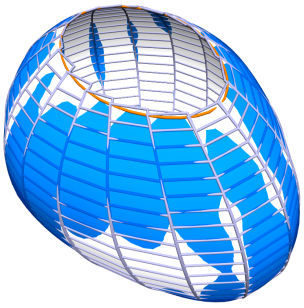
Focus on distribution and width.



Focus on developability and cylinder property.

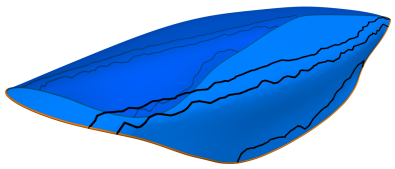


Final covering.

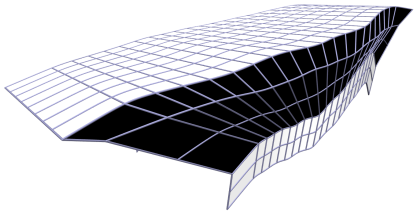


Final covering with mesh.

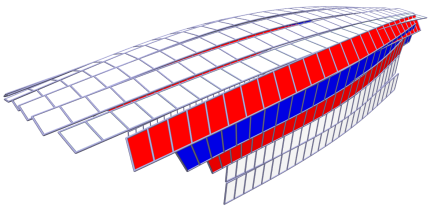
Figure 3.11: Rationalization Example 1.



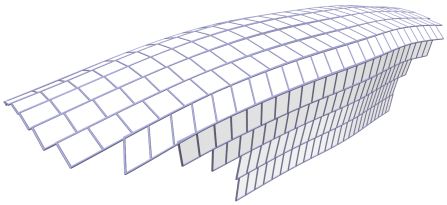
Initial mesh.



Initial covering of the mesh.



After some iterations.



Final result.

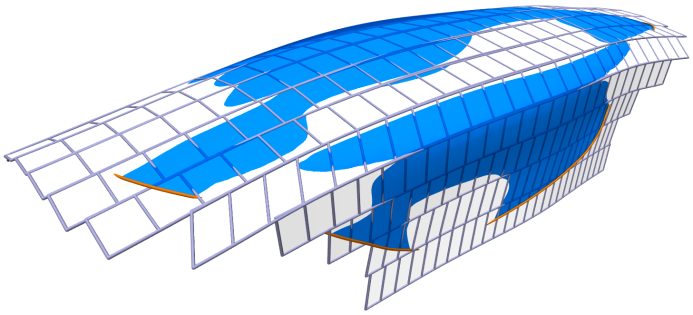
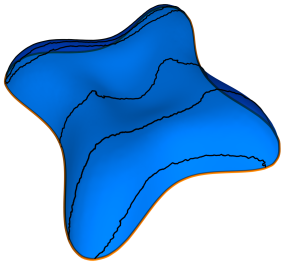
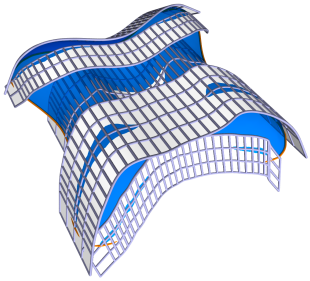


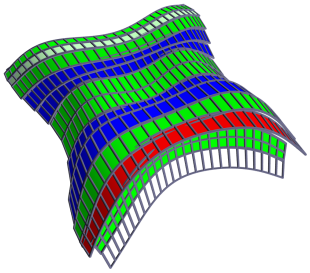
Figure 3.12: Rationalization Example 2.



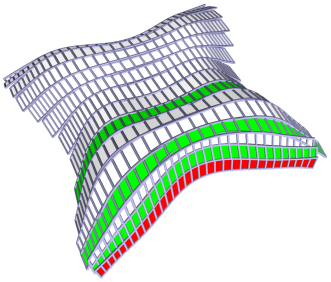
Initial mesh.



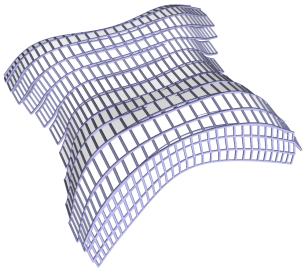
Initial covering.



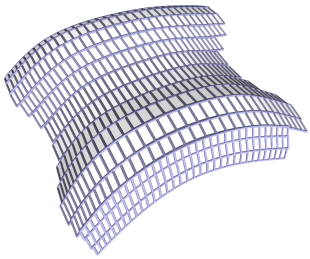
After optimizing the layout.



Enforced cylindrical property.



(1) Result with focus on closeness.



(2) Result with focus on continuity.

Figure 3.13: Rationalization Example 3.

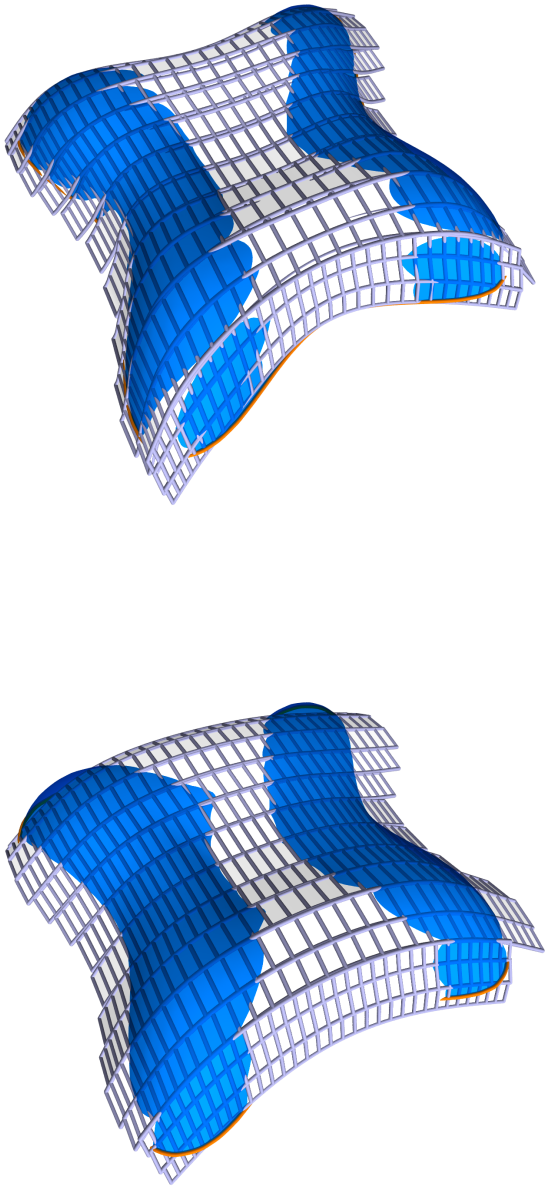
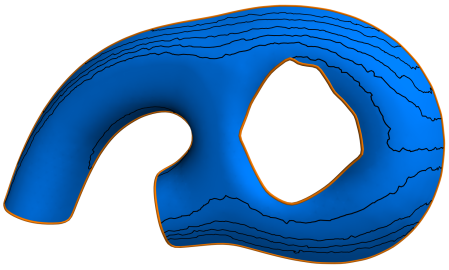
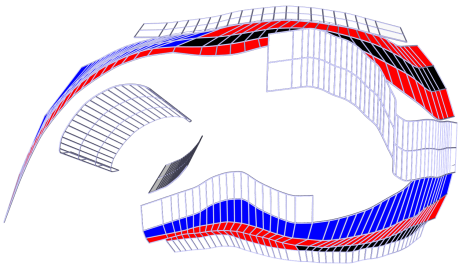


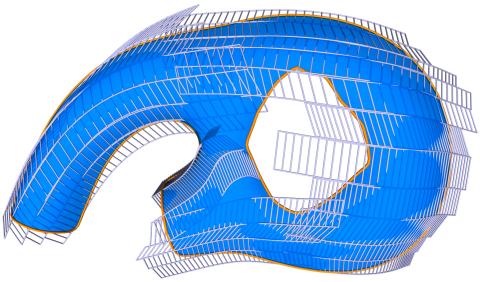
Figure 3.14: Rationalization Example 3 close-ups.



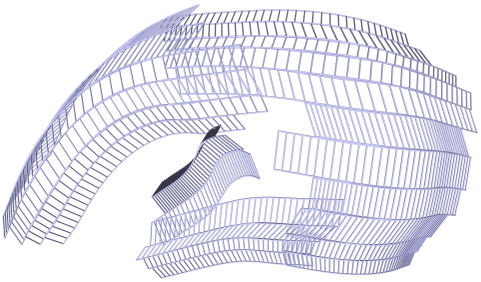
Initial mesh.



Initial covering.



Optimized covering.



Wireframes of the final covering.

Figure 3.15: Rationalization Example 4.

4 Design

4.1 Motivation

The previous sections dealt with a method for approximating given surfaces with cylindrical strips. That method guaranteed, that the resulting surface consisted of parts which have the cylinder property, i.e. parallel rulings. The negative side of the method is, that much computation is needed to only set up the initial covering.

Another interesting method is the deformation of a parametrizable quad-mesh towards a quad-mesh that exhibits cylindrical properties along one parameter line. As a discrete cylinder strip is a special case of a PQ-strip, the PQ-perturbation method presented in [Liu+06] can be adapted for this intent.

The combination of optimization and subdivision can be used, although the surface will not be subdivided in a common manner but only along the directrices to keep the cylindrical structure as good as possible while getting a smoother cylindrical strip.

This method makes it possible to design a coarse input mesh which represents the desired form of the surface and then transform it towards a piecewise finer and more cylindrical mesh.

4.2 Subdivision

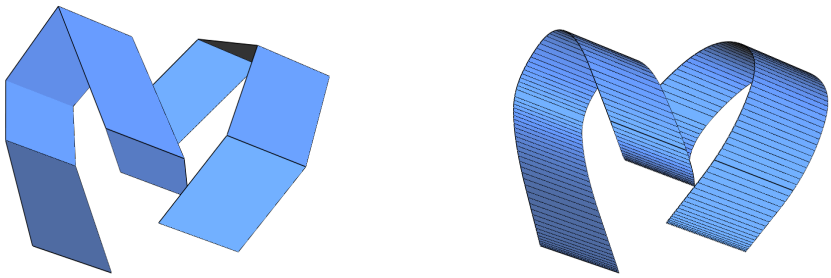
Let $C = ((\mathbf{a}_j), (\mathbf{b}_j))_{j=0,\dots,n}$ be a cylindrical PQ-strip, that is $(\mathbf{a}_j \vee \mathbf{b}_j) || (\mathbf{a}_k \vee \mathbf{b}_k)$ $\forall j, k \in \{0, \dots, n\}$. A mesh refinement along the rulings $(\mathbf{a}_j \vee \mathbf{b}_j)$ is omitted because it would yield a mesh with two families of rulings and could be interpreted as two cylindrical PQ-strips which are glued together. For subdivision along the directrices one needs a method that preserves the cylinder properties, like the Chaikin-rule applied to the boundary polylines.

The Chaikin-rule is a simple subdivision scheme which works by cutting off the corners of the polyline. For a polyline $p = (\mathbf{p}_0, \dots, \mathbf{p}_n)$ the new points are computed by

$$\begin{aligned} \hat{\mathbf{p}}_{2j} &= \frac{3}{4}\mathbf{p}_j + \frac{1}{4}\mathbf{p}_{j+1} \\ \hat{\mathbf{p}}_{2j+1} &= \frac{1}{4}\mathbf{p}_j + \frac{3}{4}\mathbf{p}_{j+1} \ , \end{aligned}$$

thus for two subsequent points, the new points are calculated as convex combinations and therfor lie on the connecting line segment of the old ones. Also, the total number of points increases from $n+1$ to $2n = (n+1)+(n-1)$ as all points except the first and last are doubled (although they are kept in the implementation to fix the boundary).

Applying this method on the boundary curves of a cylindrical PQ-strip yields a refined strip $\hat{C} = ((\hat{\mathbf{a}}_j), (\hat{\mathbf{b}}_j))_{j=0,\dots,2n-1}$. The faces spanned by $(\hat{\mathbf{a}}_{2i}, \hat{\mathbf{b}}_{2i}, \hat{\mathbf{b}}_{2i+1}, \hat{\mathbf{a}}_{2i+1})$ are obviously planar and cylindrical again. For the faces $(\hat{\mathbf{a}}_{2i+1}, \hat{\mathbf{b}}_{2i+1}, \hat{\mathbf{b}}_{2(i+1)}, \hat{\mathbf{a}}_{2(i+1)})$ holds the same, because both rulings $(\hat{\mathbf{a}}_{2i+1} \vee \hat{\mathbf{b}}_{2i+1})$ and $(\hat{\mathbf{a}}_{2(i+1)} \vee \hat{\mathbf{b}}_{2(i+1)})$ are parallel to the original ruling $(\mathbf{a}_{i+1}, \mathbf{b}_{i+1})$. Hence, $\hat{C}$ describes a cylindrical PQ-strip again and thus this technique can be applied multiple times without loss of the wanted features, for an example see fig.(4.1).



Initial surface strip. After some subdivision steps.

Figure 4.1: The subdivision process applied on a cylinder strip.

Given a regular quad-mesh $M = (v_{ij})_{i,j}$, the edges $(v_{i_0,j}, v_{i_0+1,j})_{j=0,\dots,n}$ are seen as the rulings of a strip for $i_0 = const$, so the mesh is considered to consist of strips that are glued together. If the strips are cylindrical, applying the Chaikin-rule produces a refined mesh based on cylinder strips again, so one has a method to create visually semi-smooth meshes for the specified input.

In the following, the considered mesh is required to have a suitable topology, i.e. a mesh as described in the previous paragraph. The strips will not be cylindrical or PQ-strips in general, thus the subdivision does not produce a mesh with the desired properties, but as described in the next section, it still is useful, because at least one does not lose those features.

The design method works on meshes of the form (2.14), i.e. meshes defined over a rectangular parameter domain, because these can be seen as a set of cylinder strips which are glued together along one boundary, which has to be chosen beforehand.

So given a mesh that is not closed in any direction, the boundary can be split in four parts which are seen as the parameter lines where $u = 0$, $u = 1$, $v = 0$ or $v = 1$, respectively. Choosing one of them determines which faces belong to the same strip. It does not matter if $u = 1$ is chosen over $u = 0$ or $v = 1$ is chosen over $v = 0$ as these boundary parts yield the same strips.

In case the surface is closed in one direction, the boundary is already split into two polylines and it does not matter which one would be chosen.

This method is displayed in fig.(4.2).

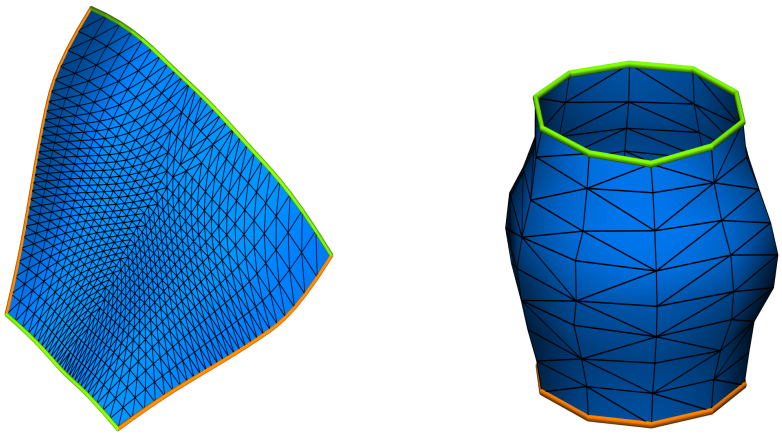


Figure 4.2: Splitting the boundary to parametrize the mesh.

4.3 Optimization

Arbitrary strips or meshes result in refined arbitrary strips or meshes, thus in these cases - which are likely to occur more often than the case of cylindrical input - a method for deforming the arbitrary surfaces into cylindrical surfaces is needed. This leads to an optimization problem similar to that in the previous chapter.

The objective function

$$f_{close}(x) = \sum_{i,j} ||y_{i,j} - x_{i,j}||^2$$

where $y_{i,j}$ are the footpoints of $x_{i,j}$ on the initial surface guarantees that the approximation surface stays close to the initial one and

$$f_{fair}(x) = \sum_{i,j} ||x_{i,j+1} - 2x_{i,j} + x_{i,j-1}||^2 + ||x_{i+1,j} - 2x_{i,j} + x_{i-1,j}||^2$$

ensures a low bending energy, preventing sharp features resulting from the optimization.

Again, a developability functional supports the cylinder objective. If a quad-face is planar, one can define a uniform normal on the face which is orthogonal to all halfedges. For an arbitrary quad-face, one can define a normal and measure how much the halfedges deviate from being orthogonal to it. Given a face f with normalized diagonal vectors $\overrightarrow{d}_1$, $\overrightarrow{d}_2$, the normal is set to $\overrightarrow{n} = \overrightarrow{d}_1 \wedge \overrightarrow{d}_2$ and for every normalized

halfedge vector $\overrightarrow{e}$ the deviation is given by $< \overrightarrow{n}, \overrightarrow{e} >$. Thus,

$$f_{dev} = \sum_{f \in F} \sum_{e \in f} < \overrightarrow{n}_f, \overrightarrow{e} >^2$$

yields a developability penalty.

The cylinder property is enforced in the same manner as in the rationalization,

$$f_{cyl} = \sum_{i,j} \left\| \frac{x_{i+1,j} - x_{i,j}}{||x_{i+1,j} - x_{i,j}||} - \overrightarrow{\ell}_i \right\|^2 .$$

These functionals lead to an optimization of the form

$$\min \; w_{close} f_{close} + w_{fair} f_{fair} + w_{dev} f_{dev} + w_{cyl} f_{cyl}$$

which is being solved using a Gauss-Newton method as in the case of rationalization. Between two subdivision steps, the optimization is done using several iterations with predefined stopping criteria like maximum number of iterations and minimal size of update step. The method is demonstrated on a strip in fig.(4.3) and further examples can be found in 4.4.

4.4 Examples and Results

It is necessary to optimize while still having only a few faces, in case the mesh is too far away from consisting of cylinder strips. So as an input, it is not useful to take a high-resolution mesh.

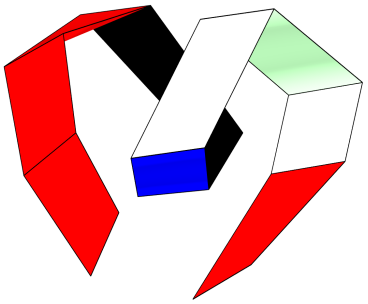
It is necessary to have a positive fairness term in the beginning because there is more perturbation first in order to approximate a cylindrical strip and the fairness term prevents visually bad perturbation. Later on, when the cylindrical property is good enough one can sometimes get rid of it completely. For surfaces which are closed in one direction, fairness must also be linked with closeness because if not, one gets a right-angular cylinder as a result.

Just as used for the rationalization examples, the quality of the approximation is color-coded. For the concrete grading see section 3.3.

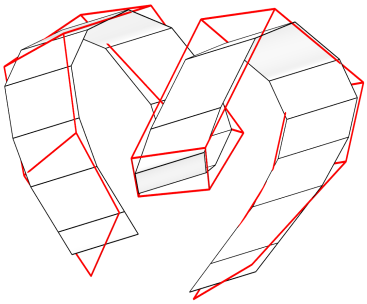
Example 1

In fig.(4.4), a deformed cylinder is processed. In the first optimization step, a small value for closeness ensures that the developability and cylindrical functional do not change the shape to much and a small fairness weight provides good visual quality. After a subdivision step, optimization is done without closeness and fairness, as the shape is already near an optimum.

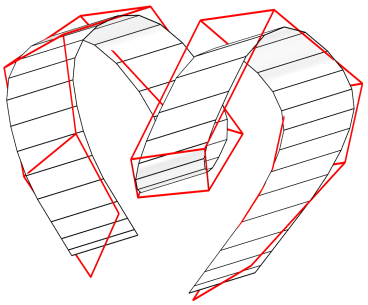
The maximum angle between successive rulings in a single face is less than 10^{-4} °, the maximum absolute deviation (the maximum absolute value of the sum of successive angles) is less than 10^{-3} °.



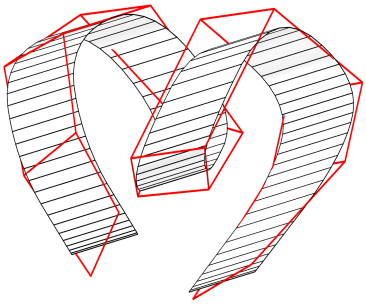
Initial strip.



After one step.



After two steps.



Refined strip with cylinder property.

Figure 4.3: Method demonstrated on a strip.

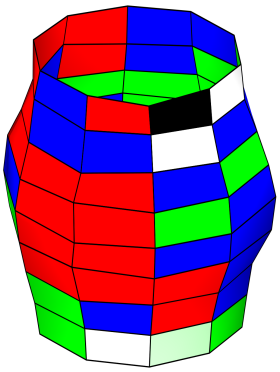
Examples 2

The same idea as above applied to another mesh, as can be seen in fig.(4.5).

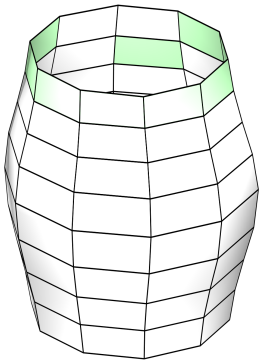
The maximum angle between successive rulings in a single face is 0.203°, the maximum absolute deviation (the maximum absolute value of the sum of successive signed angles) is 1.981°.

Example 3

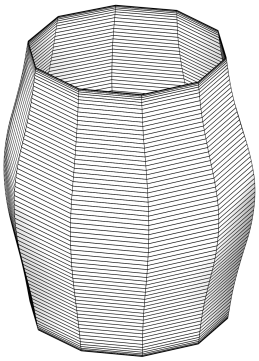
This example shall show how different choices of parameters affect the result. While both presented version work well with respect to the cylindrical property, the visual features diverge. The initial surface is refined using closeness and fairness additionally to cylindrical and developability objectives in the left column of fig.(4.6) and in the right column only fairness is added to the basic optimization.



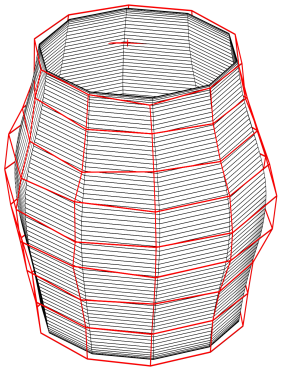
Initial surface.



After optimization.



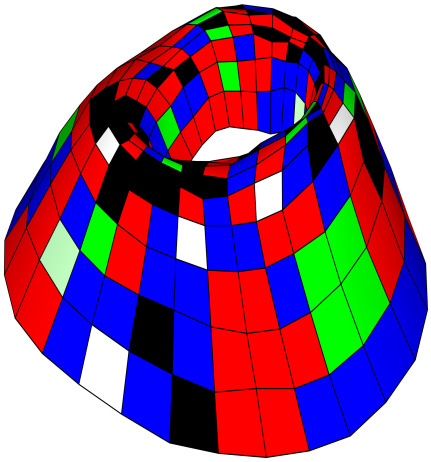
Final result.



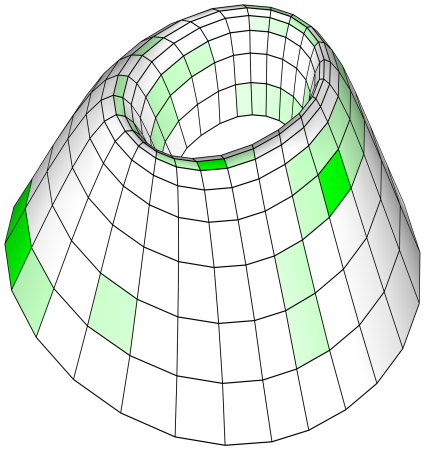
Final result with original net.

Figure 4.4: Design Example 1.

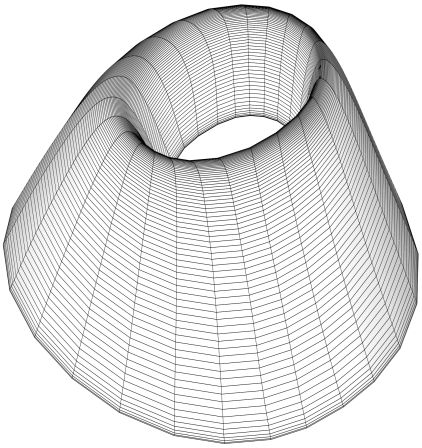
Using closeness gives a result which stays close to the original mesh, but as can be seen in fig.(4.7), it comes at the cost of visual smoothness, while on the other side, using fairness only results in a visually more appealing surface, but differs a lot from the original.



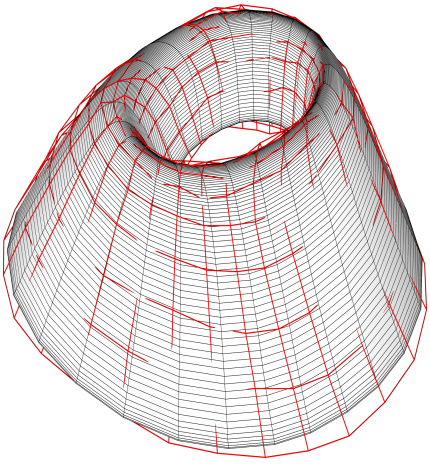
Initial surface.



After optimization.



Final result.

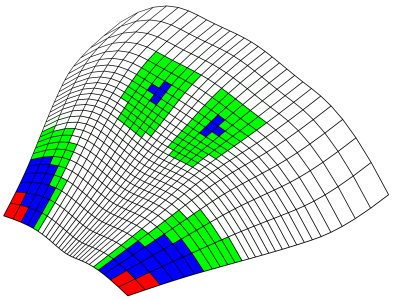


Final result with original net.

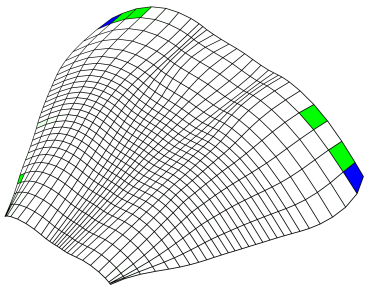
Figure 4.5: Design Example 2.

4.5 Discussion and Conclusions

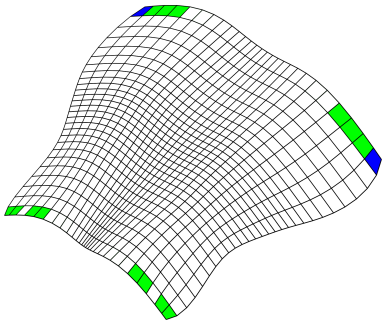
If the input mesh is not very cylindrical, it cannot be approximated reasonably, because, contrary to simply planar faces for example, the cylindrical property has to be impeded along the whole strip, thus for very skew strips the final cylindrical mesh may not have much in common with the original shape. This means that, like shown in the third example, a trade-off between different objectives has to be made depending on ones priorities.



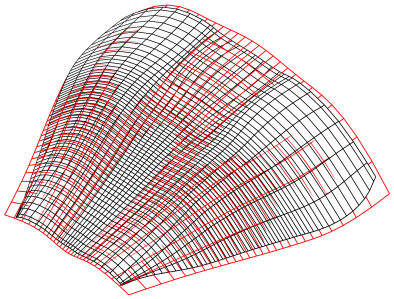
Initial surface.



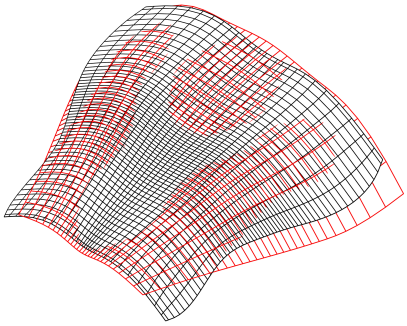
Using closeness and fairness.



Using fairness.



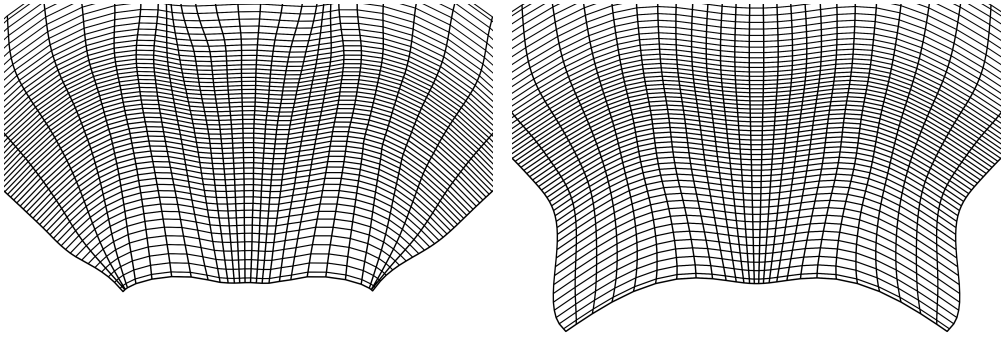
Final result with closeness and fairness.



Final result with fairness.

Figure 4.6: Design Example 3: Comparison of the method using different parameters.

Looking at some results based on the presented examples, where the number of faces is given per strip, this becomes obvious in the table on the next page.



Close-up of optimized mesh with closeness and fairness.

Close-up of optimized mesh with fairness.

Figure 4.7: Design Example 3: Detail comparison.

Object	*#Faces*		*Approx. Err.*			*Cyl. Err.*			*ms*
	init	*final*	*min*	*avg*	*max*	*cons*	*avg*	*len*	
Ex. 1	8	72	.42%	1.74%	2.85%	.001°	.000°	.016°	9050
Ex. 2	12	104	.04%	1.30%	4.39%	.018°	.003°	.343°	24490
Ex. 3 (1)	28	56	.04%	0.99%	5.59%	.021°	.002°	.136°	67870
Ex. 3 (2)	25	500	.11%	1.60%	9.90%	.003°	.000°	.012°	66260

Further subdivision was neglected for the purpose of displaying the objects with emphasized rulings. To measure the approximation quality, the distances between vertices of the final mesh and corresponding vertices in a mesh that resembles the original mesh, which undergoes only subdivision process, are taken into account and displayed with respect to the length of the diagonal of the bounding box of the original mesh.

The proposed method currently only works on regular quad-meshes. Further work should include the extraction of local strips which can be modified towards cylindrical strips while on the rest of the mesh only developability is enforced, because with this extension it would be more useful for practical applications to use such a strong objective as the cylinder property. This would also demand another subdivision method, because the Chaikin-rule as applied here is not suitable for a more complex topology. Other interesting directions are provided by works on paper-craft models like [MGE08] and [MS04], where developable strips are extracted based on features like curvature and hard edges to allow a construction based on paper strips.

5 Implementation Details

This section deals with the concrete implementation of some methods described in the previous chapters. Most of the methods were presented constructively and can be implemented straight forward, so this chapter only deals with the computation of the contour generators, the parametrization in the design method and explains the whole initialization in an end-to-end and consistent way with focus on implementation instead of mathematics.

All methods were tested and examples created using *Evolute SDK*[1] on a computer with 2.4 GHz Intel Core 2 Duo, 2 GB RAM, running Linux 32-bit.

5.1 Initialization Algorithms

Algorithm 5.1 shows the initialization algorithm that is used. As an input (`[1]`, 2.2.2), a triangulated mesh M is required as well as sampled light directions in the form of a mesh $I_M \subset \mathcal{S}^2$, then for each vertex $\overrightarrow{d} \in I_M$ (`[2]`, 3.1.3), interpreted as an illumination direction, the corresponding contour generator is computed (`[3]`, 3.1.3). As this algorithm is not as straight-forward as the rest of the initialization, it is given explicitly in the algorithms 5.1 and 5.2. Each contour generator is then evaluated (`[4]`, 3.1.4) and if found admissible its illumination direction is used as a vertex in a new, initially empty, mesh $\tilde{I}_M$. As all vertices of I_M and thus $\tilde{I}_M$ lie on the unit sphere, the missing faces of $\tilde{I}_M$ can be computed using an convex hull algorithm (compare [BDH96]), which is adapted considering the type of I_M (for samples on the upper hemisphere I_H use the upper convex hull only, for samples forming a ring I_T as in (3.16) use faces of the convex hull, where the normals form an angle between $-45°$ and $45°$ with the xy-plane and so on).

For every admissible illumination direction in $\tilde{I}_M$ (`[5]`), the corresponding optimal direction, which describes the best change of illumination in order to get a good covering, is computed (`[6]`, 3.1.5). These optimal directions are linked to the sampling mesh as elements of the tangential plane in the assigned vertex, where transfered to the faces of the mesh and rotated about $90°$, they are used as a vector field which is integrated (`[7]`, 3.1.5). From the given levelsets, one polyline can be chosen as a one-parametric subset of samples which yields those contour generators that are actually used (`[8]`, 3.1.5). The methods used in step `[9]` are a test for intersection of different contour generators (and if so, the intersecting contour with worse quality is deleted) and a sorting algorithm that guarantees that neighboring contours on the input mesh are recognized as such (if a reasonable order is possible). The output consists of the

[1] www.evolute.at

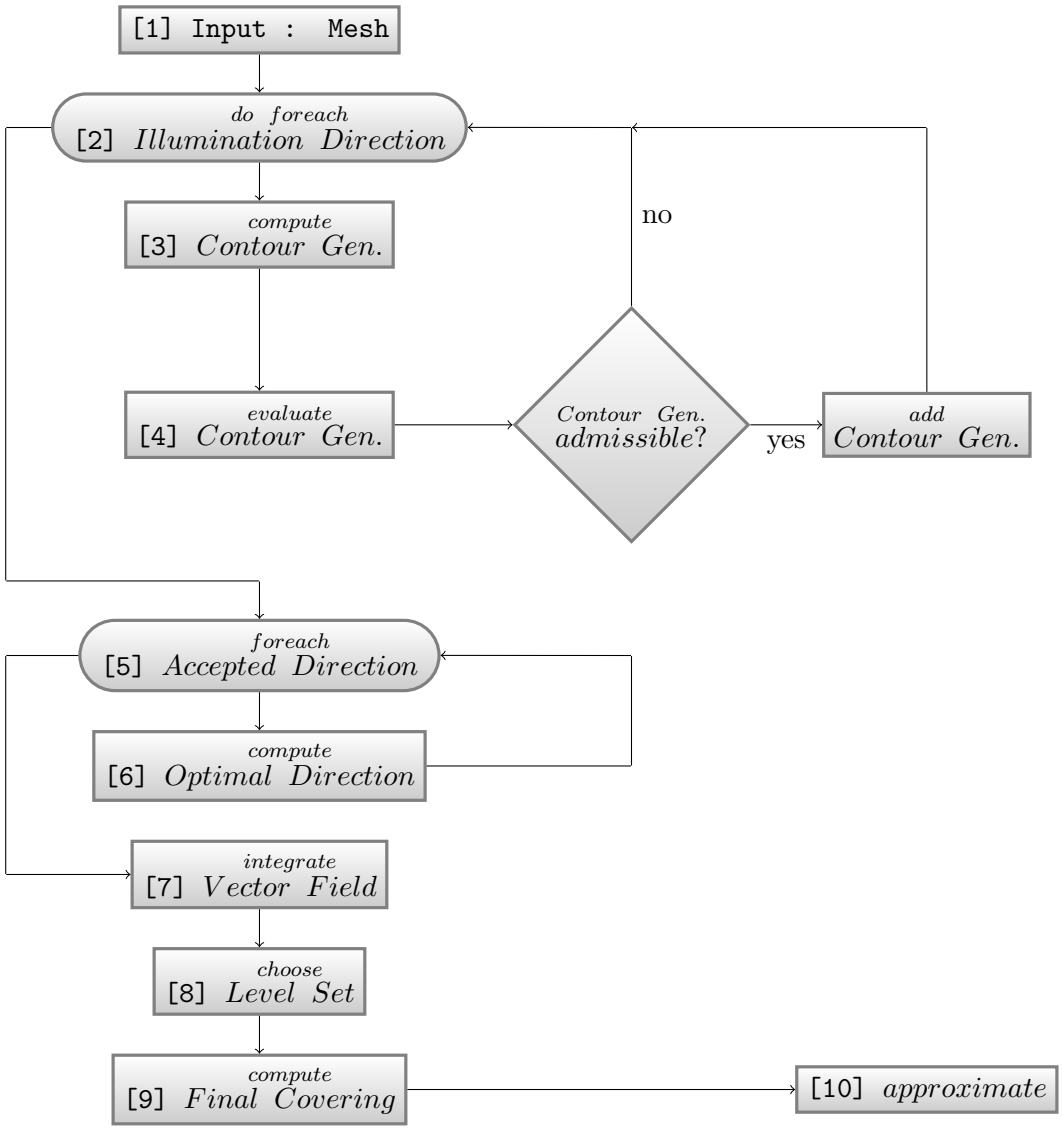


Figure 5.1: Initialization Algorithm

finally accepted contours, also referred to as final covering which then is used as input for the approximation algorithms.

The initialization is not very time consuming for small meshes, which is a consequence of the complexity of the algorithms. Let N be the number of faces of the input mesh and let K be the number of vertices of the sample mesh, then the complexities

of the sub-steps are given by

compute Contour Generator	$\mathcal{O}(N)$	
evaluate Contour Generator	$\mathcal{O}(N)$	
$\Rightarrow$ *foreach Illumination Direction*	K	$\mathcal{O}(K \cdot N)$
compute Optimal Direction	$\mathcal{O}(N)$	
$\Rightarrow$ *foreach Accepted Direction*	$\mathcal{O}(K)$	$\mathcal{O}(K \cdot N)$
test for intersections	$\mathcal{O}(K \cdot N)$	
sort contours	$\mathcal{O}(K)$ *or* $\mathcal{O}(K \cdot N)$	
$\Rightarrow$ *compute Final Covering*	∞	$\mathcal{O}(K \cdot N)$
		$\mathcal{O}(K \cdot N)$

with K usually being bounded (in the examples by 256). The running times from the examples are given below.

	K	N	*time*
Ex. 1	64	3400	1.85 *sec*
Ex. 2	160	1787	1.77 *sec*
Ex. 3	64	19957	10.04 *sec*
Ex. 4	256	7162	11.36 *sec*

5.2 Design Algorithms

To conclude the implementation, the parametrization algorithm for the design method is stated in 5.3. For the algorithm to work, the underlying mesh has to consist of quad-faces. The idea is to use the halfedge-structure to move systematically from one quad to the next. Given a halfedge in a quad considered as a ruling, one gets the corresponding halfedge of the next quad by moving along the boundary of the quad for two halfedges and then switching to the opposite halfedge.

Algorithm 5.1 Computation of the contour generators.

```
fn computeCG( mesh, dir )
  list contours;

  forall mesh.faces |f| do
    f.visited := false;
  end
  forall mesh.halfedges |e| do
    e.has_root := false;
  end

  forall mesh.faces |f| do
    if f.visited = true then continue;

    list points;
    forall f.halfedges |e| do
      if test_for_root( e, dir ) = true then
        e.has_root := true;
        points.add( root );
        next_face = mesh.get_face( e.opposite_halfedge );

        computeCG_recursive( mesh, dir, next_face, points );
      end
    end

    if points.empty() = false then contours.add( points );

    f.visited := true;
  end

  return contours;
end
```

Algorithm 5.2 Recursive method for the computation of the contour generators.

```
fn computeCG_recursive( mesh, dir, next_face, points )
  if next_face.visited = true OR
     next_face.is_valid() = false then return;

  forall next_face.halfedges |e| do
    if e.has_root = true then continue;

    if test_for_root( e, dir ) = true then
      e.has_root := true;
      points.add( root );
      next_face = mesh.get_face( e.opposite_halfedge );

      computeCG_recursive( mesh, dir, next_face, points );
      break;
    end
  end
end
```

Algorithm 5.3 Compute the parametrization of the mesh.

```
fn computeParametrization( boundary : halfedge[] )
        param : vertex[,];

        length := boundary.size();
        foreach i from 1 to length do
                e := boundary[i];
                idx := 0;

                while ( e.is_valid() = true ) do
                        param[i-1,idx] := e.start_vertex;
                        if (i = length) then
                            param[i,idx] := e.end_vertex;
                        end
                        ++idx;

                        e := e.next_halfedge;
                        e := e.next_halfedge;
                        e := e.opposite_halfedge;
                end
                e := e.opposite_halfedge;
                param[i-1,idx] := e.end_vertex;
                if (i = length) then param[i,idx] := e.start_vertex;
        end

        return param;
end
```

List of Figures

Bibliography

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. "The Quickhull algorithm for convex hulls". In: *ACM Transactions on Mathematical Software* 22 (4 1996), pp. 469–483. DOI: 10.1145/235815.235821.

[BE99] F. Benichou and Gershon Elber. "Output Sensitive Extraction of Silhouettes from Polygonal Geometry". In: *Pacific Conference on Computer Graphics and Applications*. 1999, pp. 60–69. DOI: 10.1109/PCCGA.1999.803349.

[BS08] Alexander Bobenko and Yuri Suris. *Discrete differential geometry: Integrable Structure*. Graduate Studies in Math. 98. American Math. Soc., 2008. ISBN: 978-0-8218-4700-8.

[Car76] Manfredo Perdigao do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Inc., 1976.

[CB90] Roberto Cipolla and Andrew Blake. "The dynamic analysis of apparent contours". In: *International Conference on Computer Vision*. 1990. DOI: 10.1109/ICCV.1990.139606.

[CG00] Roberto Cipolla and Peter J. Giblin. *Visual motion of curves and surfaces*. 2000.

[FHK02] G. Farin, J. Hoschek, and M. Kim. *Handbook of Computer Aided Geometric Design*. 2002.

[FP10] Simon Flöry and Helmut Pottmann. "Ruled Surfaces for Rationalization and Design in Architecture". In: *Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*. 2010.

[GK02] Carl Geiger and Christian Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer-Verlag Berlin Heidelberg, 2002.

[Goo+98] Amy Gooch et al. "A non-photorealistic lighting model for automatic technical illustration". In: *Annual Conference on Computer Graphics*. 1998, pp. 447–452. DOI: 10.1145/280814.280950.

[GW03] E. Michael Gertz and Stephen J. Wright. "Object-oriented software for quadratic programming". In: *ACM Transactions on Mathematical Software* 29 (1 2003), pp. 58–81. DOI: 10.1145/641876.641880.

[HL92] Josef Hoschek and Dieter Lasser. *Grundlagen der geometrischen Datenverarbeitung (2. Aufl.)* 1992.

[HSL] HSL(2013). *A collection of Fortran codes for large scale scientific computation.* URL: http://www.hsl.rl.ac.uk.

[Ise+03] Tobias Isenberg et al. "A Developer's Guide to Silhouette Algorithms for Polygonal Models". In: *IEEE Computer Graphics and Applications* 23 (4 2003), pp. 28–37. DOI: 10.1109/MCG.2003.1210862.

[Kel04] C. T. Kelley. "Iterative Methods for Optimization". In: (2004).

[Koe84] Jan J Koenderink. "What does the occluding contour tell us about solid shape?" In: *Perception* 13 (3 1984), pp. 321–330. DOI: 10.1068/p130321.

[Lau94] Aldo Laurentini. "The Visual Hull Concept for Silhouette-Based Image Understanding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16 (2 1994), pp. 150–162. DOI: 10.1109/34.273735.

[LBP01] Svetlana Lazebnik, Edmond Boyer, and Jean Ponce. "On Computing Exact Visual Hulls of Solids Bounded by Smooth Surfaces". In: *Computer Vision and Pattern Recognition.* Vol. 1. 2001, pp. 156–161. DOI: 10.1109/CVPR.2001.990469.

[Liu+06] Yang Liu et al. "Geometric modeling with conical meshes and developable surfaces". In: *ACM Transactions on Graphics* 25 (3 2006), pp. 681–689. DOI: 10.1145/1141911.1141941.

[Mar+97] Lee Markosian et al. "Real-time nonphotorealistic rendering". In: *Annual Conference on Computer Graphics.* 1997, pp. 415–420. DOI: 10.1145/258734.258894.

[MGE08] Fady Massarwi, Craig Gotsman, and Gershon Elber. "Paper-craft from 3D polygonal models using generalized cylinders". In: *Computer Aided Geometric Design* 25 (2008).

[MNT04] K. Madsen, H. Nielsen, and O. Tingleff. *Optimization with constraints.* 2004.

[MS04] Jun Mitani and Hiromasa Suzuki. "Making papercraft toys from meshes using strip-based approximate unfolding". In: *ACM Transactions on Graphics* 23 (3 2004), pp. 259–263. DOI: 10.1145/1015706.1015711.

[Pin] Ulrich Pinkall. *Gradient of scalar functions.* URL: http://dgd.service.tu-berlin.de/wordpress/vismathws10/2012/10/17/gradient-of-scalar-functions/.

[Pot] Helmut Pottmann. *Skriptum zur Vorlesung Geometrie in der Technik.*

[Pot+07] Helmut Pottmann et al. *Architectural Geometry.* Bentley Institute Press, 2007.

[Pot+08] Helmut Pottmann et al. "Freeform surfaces from single curved panels". In: *ACM Transactions on Graphics* 27 (3 2008). DOI: 10.1145/1360612.1360675.

[Ran98] Thomas Randrup. "Approximation of surfaces by cylinders". In: *Computer-Aided Design* 30.10 (1998), pp. 807–812. DOI: 10.1016/S0010-4485(98)00038-4.

[Sau70] R. Sauer. *Differenzengeometrie*. Springer, 1970.

[Wal] Johannes Wallner. *Skriptum zur Vorlesung Computergestuetzte Differentialgeometrie*.

[Wan+08] Lu Wang et al. "Silhouette Smoothing for Real-Time Rendering of Mesh Surfaces". In: *IEEE Transactions on Visualization and Computer Graphics* 14 (3 2008), pp. 640–652. DOI: 10.1109/TVCG.2008.8.

[Wei+02] Volker Weiss et al. "Advanced surface fitting techniques". In: *Computer Aided Geometric Design* 19 (1 2002), pp. 19–42. DOI: 10.1016/S0167-8396(01)00086-3.

[ZX06] Huanxi Zhao and Ping Xiao. *An Accurate Vertex Normal Computation Scheme*. 2006, pp. 442–451. DOI: 10.1007/11784203_38.