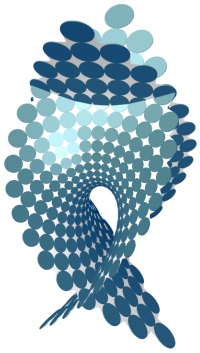
# Discrete Minimal Surfaces, Koebe Polyhedra, and Alexandrov's Theorem. Variational Principles, Algorithms, and Implementation.

Diploma Thesis

by
Stefan Sechelmann



Technische Universität Berlin
Fakultät II - Institut für Mathematik

Prof. Alexander I. Bobenko

Berlin, March 15, 2007

Die selbständige und eigenhändige Anfertigung
versichere ich an Eides statt.

Berlin, den 16. März 2007

# Contents

## Introduction

We describe the mathematics and implementation of three different computer programs. The first chapter is about Koebe polyhedra, these are polyhedra with edges tangent to the unit sphere. They are called Koebe polyhedra in honor of Paul Koebe (1882 - 1905).

THEOREM. *(Koebe) For every triangulation of the sphere there is a packing of circles on the sphere such that circles correspond to vertices and two circles touch if and only if the corresponding vertices are adjacent. This circle pattern is unique up to Möbius transformations of the sphere.*

The theorem, published in [**Koe36**], can be generalized to polytopal cellular decompositions of the sphere (Definition 1.1.2) and to circle patterns. In this setting there is a circle for every vertex and every face and these circles intersect orthogonally. A circle pattern on the sphere is defined by the radii of the circles. For such a circle pattern we can derive a polyhedron with edges tangent to the unit sphere. The resulting polyhedra are subject of the first part of this work. Figure 0.1 shows examples of Koebe polyhedra. For

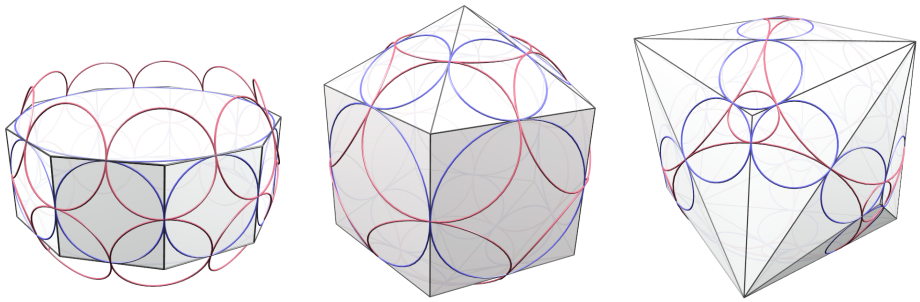


FIGURE 0.1. Koebe polyhedra

the calculation of orthogonal circle patterns on sphere we use the methods developed in [**BS04, Spr05, Spr03**]. We developed Java software for the construction and modification of Koebe polyhedra.

The second chapter of this document describes the creation of discrete minimal surfaces. Smooth minimal surfaces have been studied quite comprehensively in the past century. The latest efforts on this subject now seek for discrete analogs for the well developed theory of smooth minimal surfaces. In [**BHS06**] discrete minimal surfaces are defined and these definitions directly give rise to algorithms for calculating the surfaces. In the smooth case minimal surfaces are a subclass of isothermic surfaces and defined by their property to be dual to the image of the Gauss map. The duality transform is the Christoffel transform and this duality carries over to the discrete case. Here Koebe polyhedra play the role of the discrete Gauss map and the transformation is the discrete Christoffel transform. We describe the creation of algorithms for calculating discrete minimal surfaces and develop methods to cope with boundary conditions. We present many nice examples to illustrate

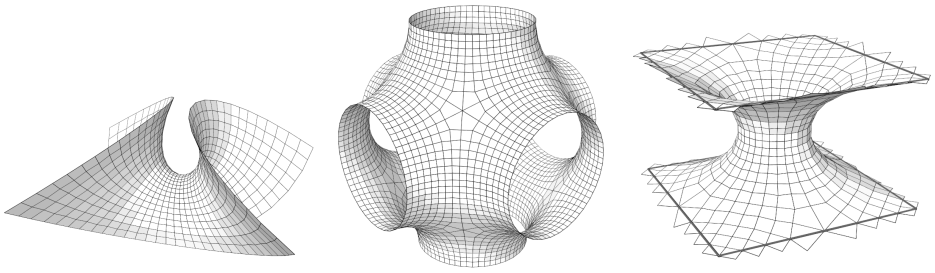


FIGURE 0.2. Discrete minimal surfaces: Enneper, Schwarz P, Schoen I-6

the usage of our algorithms. Figure 0.2 shows the Enneper, Schwarz P and the Schoen I-6 discrete minimal surfaces.

In the third chapter we focus on the creation of convex polyhedra in general. Unlike in the first part the data we have now is non-combinatorial. Alexandrov's theorem states that is it sufficient to prescribe a convex polyhedral metric on the sphere to define a convex polyhedron uniquely. The metric then defines the combinatorics of the polyhedron uniquely. [**BI06**] contains a constructive proof of Alexandrov's theorem that gives the idea for the algorithm described in this part of the work. We present software that is able to calculate convex polyhedra from a given metric. Figure 0.3 shows poly-

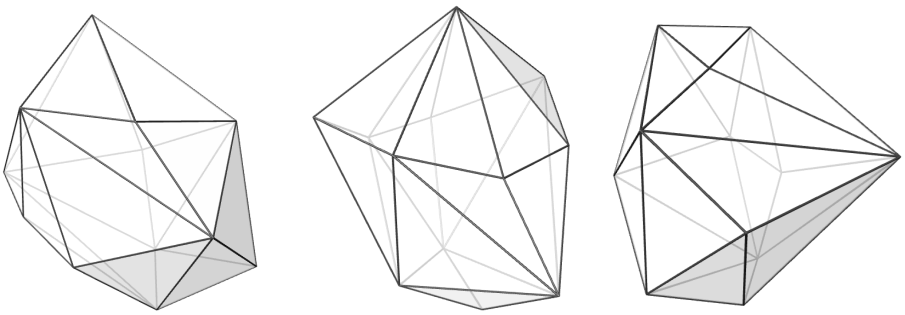


FIGURE 0.3. Random edge lengths on the icosahedron combinatorics

hedra created by using the combinatorics of the icosahedron with random edge length as input for our algorithm. To test the algorithm we calculate some D-Forms [**Bou**]. These are piecewise developable surfaces which are glued from two pieces. As a generalization we construct a Reuleaux-Triangle-Tetrahedron, a tetrahedron glued from four Reuleaux triangles.

In Chapter 4 of this work we describe some aspects of the software developed for this project. We give details about the combinatorial data structure used throughout the entire work. All the software is implemented using the programming language Java.

CHAPTER 1

# Koebe Polyhedra

## 1.1. Theory

In this chapter we describe how to construct Koebe polyhedra, these are convex polyhedra with edges tangent to the sphere. We follow the approach of [**BS04**] and add implementational details. The goal is to create an algorithm that computes Koebe polyhedra with the combinatorics of a given planar graph. To understand for which planar graphs we can compute a corresponding polyhedron, we need to have a look at the main theorems of [**BS04**]. Firstly we define the objects we are going to use.

DEFINITION 1.1.1. A family $E = \{e_\alpha\}_{\alpha \in A}$ of disjoint subsets of $S^2$ a is called a *cellular decomposition of the sphere* if

(1) $S^2 = \bigcup_{\alpha \in A} e_\alpha$,
(2) for all $e \in E$ there exists a $k(e) \in \mathbb{N}_0$ and an homeomorphism $D^{k(e)} \to e$,
where $D^{k(e)}$ is the $k(e)$ dimensional open disk,
(3) for all $e \in E$ there exists a continuous map $\varphi_e : \overline{D^{k(e)}} \to S^2$ such that the restriction

$$\varphi_e \,|_{D^{k(e)}} : D^{k(e)} \to e$$

is a homeomorphism and $\varphi_e \left(\partial D^{k(e)}\right) \subset X^{[k(e)-1]}$,
where $X^{[n]} := \bigcup\{e \in E,\, k(e) \leq n\}$ is the $n$-skeleton of $E$.

DEFINITION 1.1.2. A *polytopal cellular decomposition* is a cellular decomposition of the sphere with the combinatorics of a convex three-dimensional polytope. In particular this means that there are no edge loops, no edges share the same vertices, and every vertex has at least three adjacent edges.

The program described in the implementation section works on these polytopal cellular decompositions. The graph of such a decomposition, which is the edges and vertices, can easily be entered via a user interface. Essential for the construction of Koebe polyhedra are circle patterns and packings.

DEFINITION 1.1.3. A *circle packing* on the sphere/Euclidean plane is a configuration of circular disks in the sphere/Euclidean plane such that the disks touch but do not overlap. A *circle pattern* on the sphere/Euclidean plane is a configuration of circular disks in the sphere/Euclidean plane such that the disks might overlap.

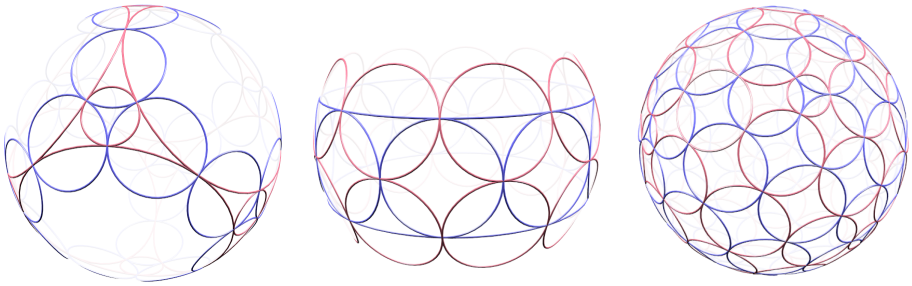


FIGURE 1.1. Circle patterns on the sphere

Figure 1.1 shows circle patterns on the sphere with orthogonally intersecting circles. The red and blue circles form circle packings on the sphere. The following theorem deals with circle patterns on the sphere:

THEOREM 1.1.4. *For every polytopal cellular decomposition of the sphere, there exists a pattern of circles on the sphere with the following properties. There is a circle corresponding to each face and to each vertex. The vertex circles form a packing with two circles touching if and only if the corresponding vertices are adjacent. Likewise, the face circles form a packing with circles touching if and only if the corresponding faces are adjacent. For each edge, there is a pair of touching vertex circles and a pair of touching face circles. These pairs touch at the same point, intersecting each other orthogonally. This circle pattern is unique up to Möbius transformations.*

This theorem gives the existence (and uniqueness) of orthogonal circle patterns on the sphere. To arrive at a polyhedron we need the next theorem.

THEOREM 1.1.5. *For every polytopal cellular decomposition of the sphere, there is a combinatorially equivalent polyhedron with edges tangent to a sphere. This polyhedron is unique up to projective transformations that fix the sphere. There is a simultaneous realization of the dual polyhedron, such that corresponding edges of the dual and the original polyhedron touch the sphere at the same points and intersect orthogonally.*

This theorem inspires the definition of Koebe polyhedra. A Koebe polyhedron is a convex polyhedron which exhibits the properties of Theorem 1.1.5.

Now we can give the answer to the question above. For every planar graph, which is combinatorially equivalent to a polytopal cellular decomposition of the sphere, a corresponding Koebe polyhedron can be computed. We call such a graph a valid graph. As Theorem 1.1.4 and 1.1.5 suggest we need to compute circle patterns on the sphere. For such a circle pattern we can construct the corresponding Koebe polyhedron. Starting with a valid graph, what is the combinatorics of the circle pattern in question? It is the medial combinatorics which has a face corresponding to every face and every vertex of the initial graph. Given a valid planar graph we construct the medial graph which has the combinatorics of the circle pattern of Theorem 1.1.4.

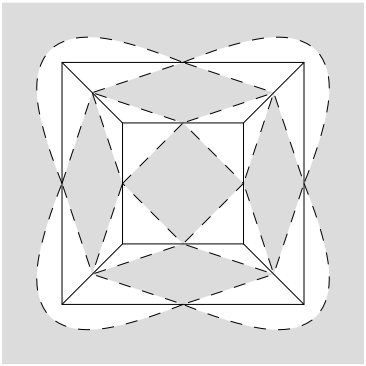


FIGURE 1.2. The medial combinatorics of the cube

The medial graph is a quad-graph, that is every face has four edges and each vertex has degree of four. Every face corresponds to a circle of the circle pattern. At each vertex two vertex circles and two face circles touch. Figure 1.2 shows the combinatorics equivalent to the cube and the corresponding medial combinatorics. Shaded faces belong to the faces of the cube, white faces correspond to vertices. The outside region is considered to be a face as well.

A circle pattern on the sphere is defined uniquely up to rotation of the sphere by its radii and intersection angles. If we prescribe intersection angles and calculate the radii, then we can recover the circle pattern by successive construction.

Faces of the initial graph belong to a circle packing on the sphere. Vertices correspond to the other circle packing. Figure 0.1 shows Koebe polyhedra with blue circles for the faces and red vertex circles. A vertex of the Koebe polyhedron is the apex of the cone that touches the unit sphere at the corresponding circle. This apex can be calculated in terms of pole and polar of the plane through the circle. Another way to get the cones apexes is to reflect the centers of the circles on the unit sphere.

In [**BS04**] a variational principle is used to approximate the radii of a circle pattern on the sphere. Here the faces of the medial combinatorics are denoted by $f_i$. For the radius $r_f$ of a face $f$ define the logarithmic radius

$$\rho_f := \log r_f. \tag{1}$$

Edges are defined as directed edges $e_j$ and $-e_j$. En edge is a pair of directed edges $e_j$ and $-e_j$. There are no single edges. Vertices are denoted by $v$.

Every undirected edge carries an angle $\theta_e$. For those angles define the intersection angle

$$\theta_e^* := \pi - \theta_e. \tag{2}$$

A circle pattern must then admit the following non-linear equations. For every face $f$ it is

$$2\pi - \sum_b (p_b + \theta_b^*) = 0 \tag{3}$$

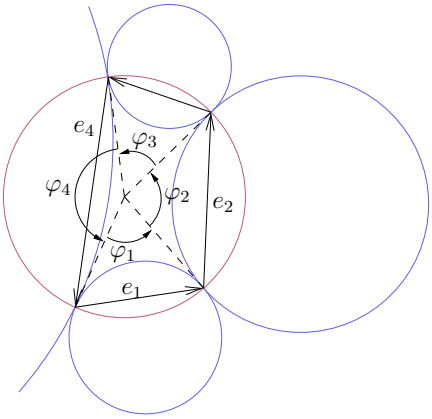



FIGURE 1.3. Angles at a circle

where the sum is taken over all boundary edges of $f$. $p_b$ is a function in terms of $\theta_e$, $\rho_{f_l}$, and $\rho_{f_r}$defined by

$$p_b \quad = \quad 2 \arctan \left( \tan \frac{\theta_b^*}{2} \tanh \frac{\rho_{f_l} - \rho_{f_r}}{2} \right).$$

Here $\rho_{f_l}$ and $\rho_{f_r}$ are the radii on the left and on the right of the edge $b$. Figure 1.3 shows the angles between the intersections of the circles and the center. For an edge $e$ the angle $\varphi_e$ is

$$\varphi_e = p_e + \theta_e^*.$$

Thus, for a circle around face $f$, all neighboring circles fit around and form a correct circle pattern if the radii admit Equation 3. For an orthogonal circle pattern we set $\theta_e = \pi$ for all edges.

The solution of Equation 3 can be understood as the critical point of a functional $S$. Such a circle pattern functional is a function

$$S : \mathbb{R}^F \to \mathbb{R}$$

where $F$ is the number of circles in the pattern or the number of faces in the underlying combinatorics. [**BS04, Spr03**] define three different functionals: An Euclidean, a spherical, and a hyperbolic functional. It is shown that critical points of $S$ belong to circle patterns in the respective space. Thus minimizing the spherical functional leads to a circle pattern on the sphere.

As pointed out by the authors, the spherical functional is not convex. Thus standard numerical methods are not guaranteed to succeed. The Euclidean circle pattern functional on the other hand is convex and can be optimized using Newton's method. Fortunately the Euclidean circle pattern functional can often be used to calculate circle patterns on the sphere as well. A circle pattern in the Euclidean plane can be projected stereographically to the sphere. Since stereographic projection preserves angles, the resulting intersection angles are correct. In our special case of orthogonal circle patterns the projected pattern will be an orthogonal circle pattern as well.

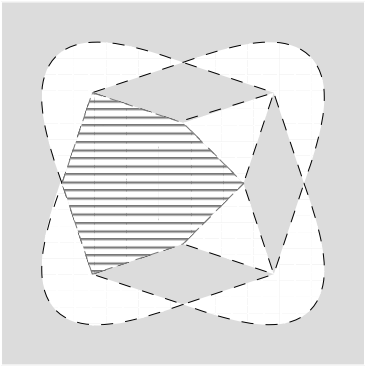


FIGURE 1.4. The cube medial combinatorics with $v_\infty$ removed

The Euclidean circle pattern functional is

$$\begin{aligned} S_{Euc}(\rho) &= \sum_{f_j \circ \!\!\!\!\bullet\!\!\!\!\mid \circ f_k} [p_e(\rho_{f_k} - \rho_{f_j}) - \theta_e^*(\rho_{f_j} + \rho_{f_k}) \\ &\quad + \mathrm{Cl}_2(\theta_e^* + p_e) + \mathrm{Cl}_2(\theta_e^* - p_e) - \mathrm{Cl}_2(2\theta_e^*)] \\ &\quad + \sum_f \Phi_f \rho_f. \end{aligned} \tag{4}$$

A definition of Clausen's integral $\mathrm{Cl}_2$ can be found in the appendix of [**BS04, Spr03**]. To understand why a circle pattern is a critical point of this functional we have to inspect its derivatives. It is

$$\frac{\partial S_{Euc}}{\partial \rho_f} = \Phi_f - \sum_b (p_b + \theta_b^*) \tag{5}$$

$$S''_{Euc} = \sum_{f_j \circ \!\!\!\!\bullet\!\!\!\!\mid \circ f_k} \frac{\sin\theta_e}{\cosh(\rho_{f_k} - \rho_{f_j}) - \cos\theta_e} (\mathrm{d}\rho_{f_k} - \mathrm{d}\rho_{f_j})^2 \tag{6}$$

where the first sum is taken over all boundary edges of $f$. Setting $\Phi_f = 2\pi$ for a face therefore implies $\sum_b \varphi_b = 2\pi$ for a critical point of the functional.

To calculate circle patterns for the entire sphere we choose a vertex $v_\infty$ from the medial combinatorics and use it as the center of projection. Stereographic projection maps this vertex to infinity. Thus the circles through this vertex are straight lines after projection. This observation leads to the idea to remove the vertex $v_\infty$ from the medial graph and calculate the circle pattern for the remaining disk. Removing a vertex includes the removal of all adjacent edges and faces. The medial graph is a quad-graph and removing a vertex implies removing four edges and four faces. The four circles of the removed faces are two vertex and two face circles. Figure 1.4 shows the resulting graph of the cube combinatorics after the removal of $v_\infty$. The circles of the four removed faces intersect orthogonally, so that the resulting circle pattern is bounded by a rectangle. Figures 1.5 and 1.6 show the circle patterns of the cube and the icosahedron respectively. This proceeding is also described in [**Zie04**].

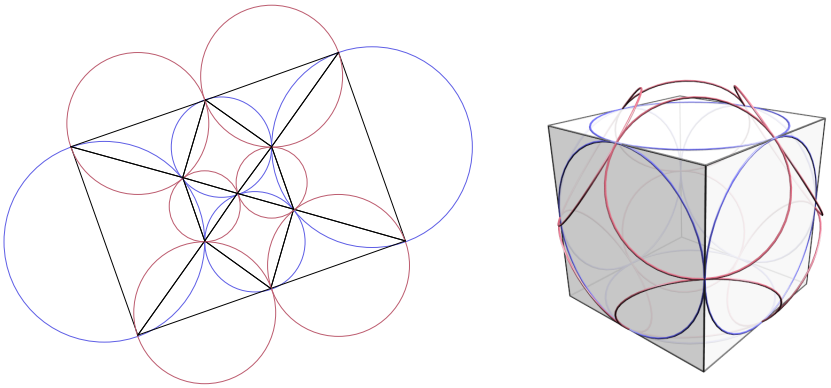


FIGURE 1.5. The medial combinatorics and circle pattern of the cube in the Euclidean plane

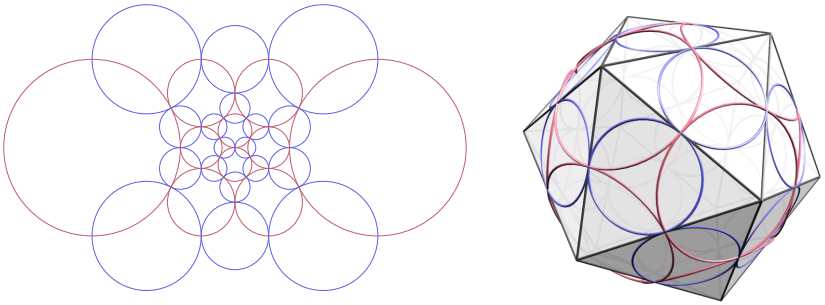


FIGURE 1.6. The circle pattern of the icosahedron

Assuming circles of infinite radius at the boundary, we can compute circle patterns for the whole Euclidean plane. The angles $\theta^*$ then define intersection angles with the boundary. Some quantities in the calculation of $S_{Euc}$ however are only defined for internal edges. To calculate the functional the sums in 4, 5, and 6 are taken over internal edges only. For the faces we set

$$\Phi_f = \begin{cases} 2\pi & f \text{ is interior edge} \\ 2\pi - \sum_{e_b} 2\theta^*_{e_b} & f \text{ is boundary face} \end{cases}$$

where the sum is taken over all boundary edges of $f$ which are also boundary edges of the combinatorics.

Now we can calculate circle patterns on the sphere using the Euclidean circle pattern functional and stereographic projection. The projection that maps points from the plane to the sphere is given by

$$\begin{aligned} \pi_{vertex} : \mathbb{R}^2 &\to S^2 \\ \begin{pmatrix} x \\ y \end{pmatrix} &\mapsto \begin{pmatrix} 2x \\ 2y \\ x^2+y^2-1 \\ x^2+y^2+1 \end{pmatrix}. \end{aligned} \tag{7}$$

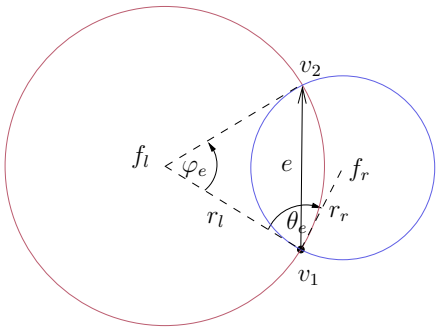



FIGURE 1.7. Edge layout

To get the Koebe polyhedron of a circle pattern, we have to compute the poles of the corresponding circles. A pole of a plane that intersects the unit sphere is the apex of the cone that touches the sphere at the circle of intersection. Let $ax + by + cz + dw = 0$ be the plane through the circle of face $f_i$. The pole of this plane with respect to the unit sphere is given by $(a, b, c, -d)$.

## 1.2. Implementation

**1.2.1. The Circle Pattern.** The gradient of the Euclidean circle pattern functional is a function

$$S'_{Euc} : \mathbb{R}^F \to \mathbb{R}^F.$$

We solve the equation $S'_{Euc}(\rho) = 0$ by using Newton's method. Taylor expansion gives

$$S'_{Euc}(\rho^0 + \rho^s) \approx S'_{Euc}(\rho^0) + S''_{Euc}(\rho^0) \cdot \rho^s.$$

Therefore Newton's step requires solving the linear system

$$S'_{Euc}(\rho^0) = -S''_{Euc}(\rho^0)\rho^s.$$

Since the resulting circle pattern is unique up to scale and Euclidean motion, we choose a reference face $f_0$ and the set $\rho_0 = 0$. We reduce the system to size $F - 1$ to keep the reference radius fixed. To speed up convergence, we use the Armijo rule for controlling the step-size in Newton's method. Choose $0 < \alpha < 0.5$, $0 < \beta < 1$ and for every step find $n \in \mathbb{N}$ such that

$$S_{Euc}(\rho^0 + \beta^n \rho^s) \leq S_{Euc}(\rho^0) + \alpha < S'_{Euc}(\rho^0), \beta^n \rho^s > .$$

Then set $\rho^{n-1} := \rho^n + \beta^n \rho^s$ for the next step. We use $\alpha = 0.2$ and $\beta = 0.5$. See [**BV04**] pp. 463-466 for a comprehensive description of this step-size controller. We utilize the numerical algorithms of the "Matrix Toolkits for Java API [**Hei**]" and the package jtem [**TBa**] for linear system solving and matrix factorization.

Minimizing the Euclidean functional produces correct radii for the given combinatorics in the plane. We calculate the coordinates (centers and intersection points) of the circle pattern with an iterative layout algorithm. Starting with an arbitrary face, we enumerate the combinatorics in a depth-first manner. Figure 1.7 shows the layout at edge $e$. Given the center of $f_l$

and the start vertex $v_1$, the position of $f_r$ is obtained by rotation of $f_l$ by $\theta^*$ around $v_1$ with the scale factor is $\frac{r_r}{r_l}$. The position of $v_2$ is the position of $v_1$ rotated by $\varphi_e$ around $f_l$.

The orthogonal circle pattern on the sphere is obtained by stereographic projection to the unit sphere with center at the origin. The point $\widehat{p_0} = (0, 1, 0, 1)$ is the center of projection. Intersection point of the circles map to points on the sphere by the Mapping 7. In the implementation we interchange the $y$ and $z$ coordinates. As of the medial graph construction there are face circles and vertex-circles belonging to faces and vertices of the initial combinatorics. The Centers of vertex-circles map to the pole of the corresponding circle on the sphere and form the vertices of the Koebe polyhedron. Let $r$ be the radius of the mapped circle. The mapping is

$$\begin{array}{rcl} \pi_{center} : \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ \begin{pmatrix} x \\ y \end{pmatrix} & \mapsto & \begin{pmatrix} 2x \\ 2y \\ x^2+y^2-r^2-1 \\ x^2+x^2-r^2+1 \end{pmatrix}. \end{array}$$

The removed faces of the medial graph belong to circles through infinity. They are projected to the north pole of the sphere. The pole of such a circle can be expressed in terms of the vertices of a boundary edge. Let $p_i$ and $p_j$ be those vertices, and let $ax + by + cz + dw = 0$ be the plane through the three points $p_i$, $p_j$ and $\widehat{p_0}$. Calculation yields

$$\begin{array}{rcl} \det \begin{bmatrix} x_i & x_j & 0 & x \\ 0 & 0 & 1 & y \\ y_i & y_j & 0 & z \\ 1 & 1 & 1 & w \end{bmatrix} & = & (y_j - y_i)x + (x_i y_j + y_i x_j)y \\ & & +(x_i - x_j)z + (y_i x_j - x_i y_j)w. \end{array}$$

Then the pole is given by

$$(p_i, p_j) \mapsto \begin{pmatrix} y_j - y_i \\ x_i y_j + y_i x_j \\ x_i - x_j \\ -(y_i x_j - x_i y_j) \end{pmatrix}$$

where $p_i = (x_i, y_i)$, $p_j = (x_j, y_j)$.

**1.2.2. Normalization.** The resulting polyhedron is unique up to projective transformations that fix the sphere. [**Spr05**] describes a unique representation with the property that the barycenter of the vertices is the center of the unit sphere. Figure 1.8 shows a cube Koebe polyhedron in a normalized and non-normalized representation. Let $p_1, ..., p_n$ be the vertices on the sphere in homogeneous coordinates $p_i = (x_i, y_i, z_i, w_i)$. Define the scalar product

$$< p_i, p_j >:= x_i x_j + y_i y_j + z_i z_j - w_i w_j.$$

Minimize the function

$$F(p) = \sum_{i=1}^{n} - \log \left( - \frac{< p, p_i >}{< p, p >} \right)$$

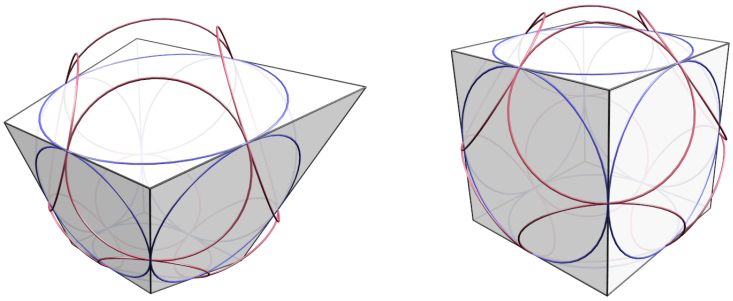


FIGURE 1.8. The cube Koebe polyhedron non-normalized and normalized version

to end up with a point $\hat{p}$ inside the sphere. Normalize $\hat{p}$ such that $< p, p >= -1$. It is $p = \frac{\hat{p}}{\sqrt{-<\hat{p},\hat{p}>}}$. Use the Gram-Schmidt orthogonalization method and the vectors $p$, $(0,0,1,0)$, $(0,1,0,0)$, $(1,0,0,0)$ to construct an orthonormal transformation matrix $A$. Calculate the inverse of that matrix with respect to the scalar product $<,>$. It is

$$A^{-1} = E^T A E$$

where

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Transform the points $p_1, ..., p_n$ with this inverse to arrive at the normalized point set.

The minimization of $F$ is carried out using the same algorithm as for the Euclidean circle pattern functional. This approach can fail as the functional is non-convex and there are examples which produce incorrect results. To improve the results of the minimization, we try to start at a good guess. We introduce a pre-conditioning by translation and scaling of the circle pattern before the projection. Ensure that one half of the points gets mapped to the upper hemisphere and the other to the lower one after stereographic projection. In vast majority of examples this treatment is sufficient and the minimization succeeds.

## 1.3. The Program

The goal of the implementation was to design a program that could create and edit Koebe polyhedra. The user should enter an embedded graph which is automatically translated into the corresponding polyhedron. The Java software, which has been created for this project, is included on CD-ROM (Section 4.2).

The Application consists of the graph editor on the left side and a 3D viewer on the right. If the input graph is valid the algorithm is run and the resulting polyhedron is displayed. There are several viewing options which include

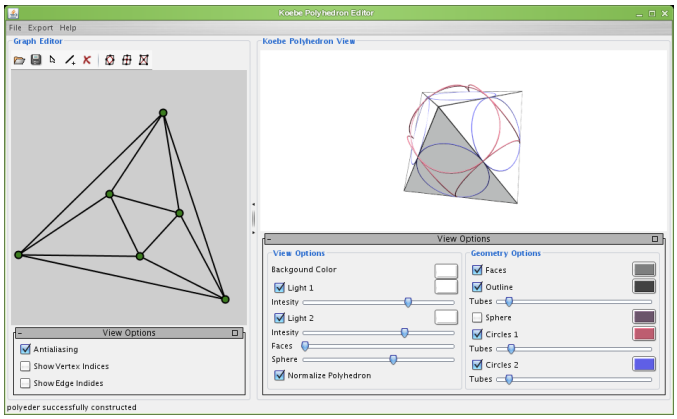



FIGURE 1.9. The Koebe polyhedron editor

check-boxes for circles, mesh, and faces (Figure 1.9). Internally the graph is represented by a half-edge data structure which is described in Section 4.1.

CHAPTER 2

# Discrete Minimal Surfaces

This chapter discusses the construction of discrete minimal surfaces. We use the method developed by Bobenko, Hoffmann, and Springborn in [**BHS06**].

## 2.1. Theory

A reasonable definition of discrete minimal surfaces carries as much properties as possible from the smooth to the discrete case. As described in the article, the duality transform of minimal surfaces has a discrete analog. [**BHS06**] defines the Christoffel transform of a discrete minimal surface to be the discrete analog of the Gauss map. This definition leads to a construction method once this discrete Gauss map can be computed. As defined in the article, the Gauss map of a discrete minimal surface corresponds to a part of a Koebe polyhedron. Thus we can compute Koebe polyhedra and dualize them to obtain discrete minimal surfaces. To understand the properties of the duality transformation of our surfaces, we need to have a look at a more general class of surfaces first.

In the smooth world minimal surfaces are a subclass of isothermic surfaces. In the discrete case discrete minimal surfaces are a subclass of discrete isothermic surfaces. Discrete minimal surfaces are quad-meshes whose quadrilaterals exhibit the following property.

Definition 2.1.1. Four points $p_1, \ldots, p_4$ form a *conformal square* if and only if there exists a Möbius transformation such that the transformed points form a square.

Discrete isothermic surfaces are now defined in terms of these conformal squares.

Definition 2.1.2. Let $\mathcal{D}$ be a quad-graph such that the degree of every interior vertex is even. (That is, every vertex has an even number of edges.) Let $V(\mathcal{D})$ be the set of vertices of $\mathcal{D}$. A function

$$f : V(\mathcal{D}) \to \mathbb{R}^3$$

is called a discrete isothermic surface if for every face of $\mathcal{D}$ with vertices $v_1$, $v_2$, $v_3$, $v_4$ in cyclic order, the points $f(v_1)$, $f(v_2)$, $f(v_3)$, $f(v_4)$ form a conformal square.

In [**BHS06**] discrete minimal surfaces are defined as discrete isothermic surfaces which are dual to Koebe polyhedra. The following proposition from [**BHS06**] defines this duality transformation.

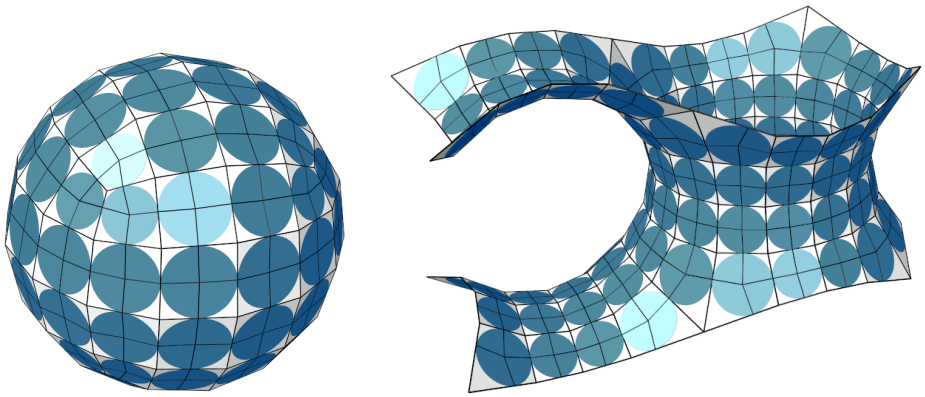


FIGURE 2.1. The quad-graph of a Koebe polyhedron and the dualized surface

PROPOSITION 2.1.3. *Let $f : V(\mathcal{D}) \to \mathbb{R}^3$ be a discrete isothermic surface, where the quad-graph $\mathcal{D}$ is simply connected. Then the edges of $\mathcal{D}$ may be labeled "+" and "−" such that each quadrilateral has two opposite edges labeled "+" and the other two opposite edges labeled "−". The dual discrete isothermic surface is defined by the formula*

$$\Delta f^* = \pm \frac{\Delta f}{\| \Delta f \|^2} \tag{8}$$

*where $\Delta f$ denotes the difference of neighboring vertices and the sign is chosen according to the edge label.*

DEFINITION 2.1.4. Let $f : V(\mathcal{D}) \to \mathbb{R}^3$ be a discrete isothermic surface. $f$ is a discrete minimal surface if and only if the the dual discrete isothermic surface corresponds to a Koebe polyhedron.

That means if the dual of a discrete isothermic surface is the quad-graph of a Koebe polyhedron, it is a discrete minimal surface. The quad-graph of a Koebe polyhedron consists of all conformal squares, thus it is a discrete isothermic surface. Figure 2.1 (left) shows such a quad-graph. All quadrilaterals are kites with at least two right angles. These kites are always conformal squares.

We now construct discrete minimal surfaces by dualizing the quad-graph of our Koebe polyhedron from Part 1 of this work. Figure 2.1 (right) shows the dualized quad-graph of the polyhedron on the left. The vertices with degree of three correspond to singularities of the resulting surface. Here we cut the combinatorics to resolve the singularities. To gain more control over the shape of the result we have to deal with boundary conditions and umbilic points.

The goal of the implementation was to create an application that could build discrete minimal surfaces with a prescribed combinatorics of curvature lines. Additionally we want to create umbilic points at the boundary. The boundary of the combinatorics is either a curvature line or an asymptotic line depending of the boundary style. It is not possible to control such

behavior using the Euclidean functional and stereographic projection. For that reason we use the spherical functional described in [**BHS06, Spr03**].

Here as in the Euclidean case the parameters of the functional are the radii of the corresponding circle pattern in the sphere. For the radius $r_f$ of face $f$ define

$$\rho_f := \log \tan \frac{r}{2}.$$

The spherical functional is

$$\begin{aligned} S_{sph}(\rho) &= \sum_{f_i \circ \bullet \circ f_j} \lambda_e \left[ p_e(\rho_{f_k} - \rho_{f_j}) - s_e(\rho_{f_k} + \rho_{f_j}) - \pi(\rho_{f_k} + \rho_{f_j}) \right. \\ &\quad + \mathrm{Cl}_2(\theta_e^* + p_e) + \mathrm{Cl}_2(\theta_e^* - p_e) - \mathrm{Cl}_2(2\theta_e^*) \\ &\quad \left. - \mathrm{Cl}_2(\theta_e + s_e) + \mathrm{Cl}_2(\theta_e - s_e) - \mathrm{Cl}_2(2\theta_e) \right] \\ &\quad + \sum_{\circ f} \Phi_f \rho_f. \end{aligned} \tag{9}$$

Here $p_e$ is defined like above and $s_e$ is

$$s_e = 2 \arctan \left( \tan \frac{\theta_e^*}{2} \tanh \frac{\rho_{f_k} + \rho_{f_j}}{2} \right).$$

The sum in Equation 9 is taken over all edges including the boundary. For a boundary edge $e_b$ set $\rho = 0$ for the missing face. $\lambda_e$ is defined for all undirected edges. It is

$$\lambda_e := \begin{cases} \frac{1}{2} & f_e \text{ and } f_{-e} \text{ are boundary faces} \\ 1 & \text{else} \end{cases}$$

where a face $f$ is said to be a boundary face if the boundary of $f$ contains a boundary edge. See Section 2.1.2 for a detailed description of parameter $\lambda_e$. The derivatives of the functional are

$$\begin{aligned} \frac{\partial S_{sph}}{\partial \rho_f} &= \Phi_f - \sum_b \lambda_b (p_b + s_b + \pi) \\ S''_{sph} &= \sum_{f_j \circ \bullet \circ f_k} \lambda_e \left( \frac{\sin \theta_e}{\cosh(\rho_{f_k} - \rho_{f_j}) - \cos \theta_e} (\mathrm{d}\rho_{f_j} - \mathrm{d}\rho_{f_k})^2 \right. \\ &\quad \left. - \frac{\sin \theta_e}{\cosh(\rho_{f_k} + \rho_{f_j}) + \cos \theta_e} (\mathrm{d}\rho_{f_j} + \mathrm{d}\rho_{f_k})^2 \right). \end{aligned}$$

Again like for the Koebe polyhedron $\frac{\partial S_{sph}}{\partial \rho_f} = 0$ implies that the angles $\varphi_e$ make the circles fit. It is

$$\varphi_e = p_e + s_e + \pi.$$

Therfore we set $\Phi_f = 2\pi$ for all faces $f$. For a boundary edge it is $p_b + s_b = 0$ so

$$\varphi_b = \pi$$

holds for all boundary edges.

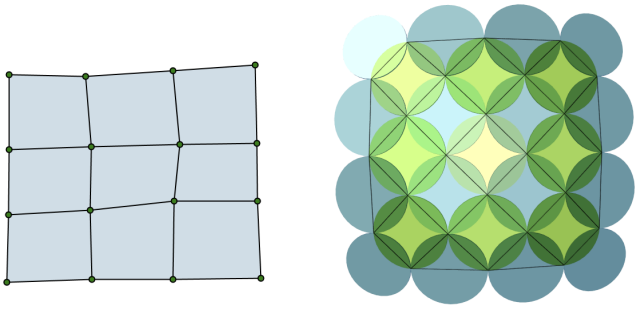


FIGURE 2.2. Curvature line sketch (left) and the corresponding circle pattern in the sphere (right)

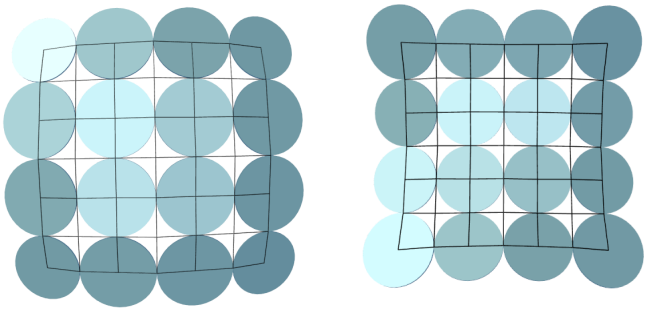


FIGURE 2.3. Koebe polyhedron and dual minimal surface

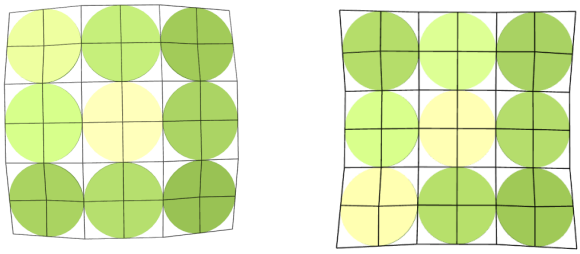


FIGURE 2.4. Koebe polyhedron and dual

**2.1.1. Construction.** The construction of a discrete minimal surface is carried out in 4 steps. For the first step the curvatures line pattern for the resulting surface or a fundamental piece of the surface is created. Figure 2.2 (left) shows the sketch of a simple quad grid curvature line pattern.

As for the Koebe polyhedra we construct the medial combinatorics and the respective circle pattern in the sphere. This is step two of the creation process. Figure 2.2 (right) shows the medial combinatorics and the circle pattern for the sketched curvature lines.

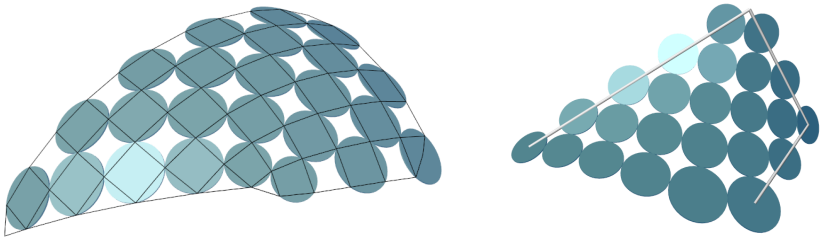


FIGURE 2.5. Boundary conditions dual polyhedron

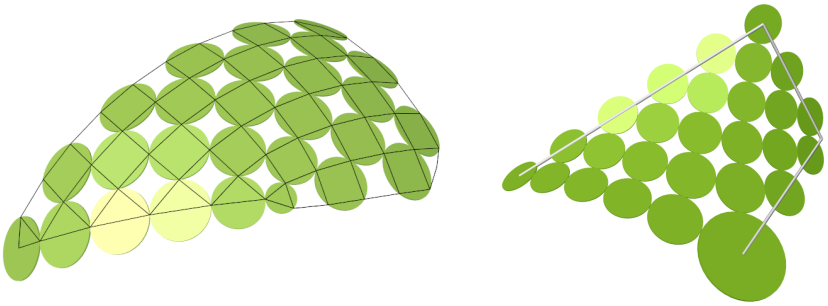


FIGURE 2.6. The possible boundary combinatorics

Step three produces a part of a Koebe polyhedron. Here we choose one of the two possible polyhedra. The standard polyhedron has flat faces corresponding to faces of the curvature line pattern, each consisting of four faces of the quad-graph. The other polyhedron's faces correspond to vertices of the curvature line pattern. Figure 2.4 shows the standard polyhedron. A face in the curvature line pattern results in a circle of the minimal surface. Figure 2.3 shows the dual Koebe polyhedron, where vertices of the curvature line pattern correspond to circles of the minimal surface. Technically we use a vertex-face subdivision of the medial combinatorics to arrive at the quad-graph of the Koebe polyhedron.

The last step assigns "+" and "-" edge labels according to the dualization rule to the quad-graph. Then dualize the quad-graph with the discrete Christoffel transform (Equation 8).

**2.1.2. Boundary Conditions.** The Koebe polyhedron is the discrete analogon of the Gauss map of the minimal surface. Therefore straight asymptotic lines and planar curvature lines are mapped to great circles by the Gauss map. Thus we want the circle pattern on the sphere to be bounded by great circles. Then the boundary of the resulting minimal surface will be either a planar curvature line or a asymptotic line depending on the combinatorics of the boundary.

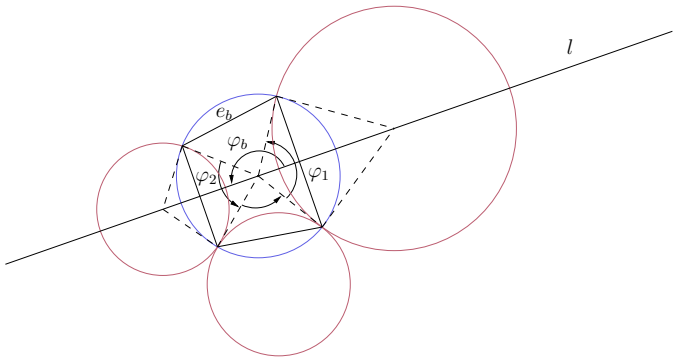



FIGURE 2.7. Asymptotic boundary

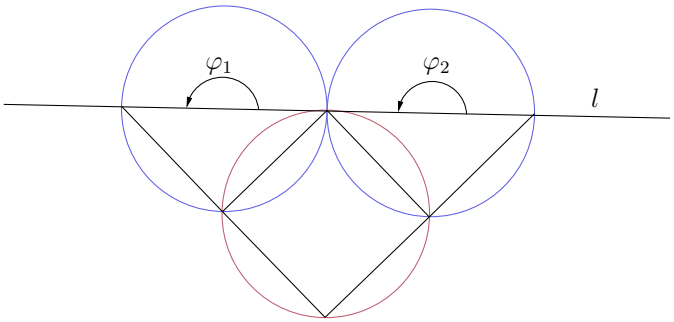



FIGURE 2.8. Curvature line boundary

An asymptotic boundary line is produced by orthogonally intersecting circles whose centers lie on a great circle. Figure 2.7 shows such configuration. Since $e_b$ is a boundary edge, the corresponding angle $\varphi_b$ is equal to $\pi$. To make the circles fit at such a boundary the parameter $\lambda_e$ is equal to $\frac{1}{2}$ for the edges of $\varphi_1$ and $\varphi_2$. The centers of the circles after dualization lie on a straight line. Figure 2.6 and 2.5 show an example for this kind of boundary. Three of the four boundaries are straight asymptotic lines.

A curvature line boundary results from the combinatorics shown in Figure 2.8. Here the intersections of the circles at the boundary lie on a great circle $l$. Again the angles at the boundary $\varphi_1$ and $\varphi_2$ are equal to $\pi$. Thus the line $l$ is a great circle and the dualized circles on the boundary form a planar curvature line.

If the minimal surface is bounded by a straight asymptotic line, then it can be continued smoothly by 180°-rotation about it.

If the surfaces boundary is a planar curvature line, then reflection in this plane continues the surfaces smoothly.

**2.1.3. Umbilic Points.** At an umbilic point a finite number of curvature lines end. We model umbilic points at the boundary of our combinatorics and have to adjust the respective $\Phi_f$. Decreasing $\Phi_f$ results in a smaller angle of the circle pattern at the corner. A boundary edge takes up $\pi$ of the available angle. Thus having an extra vertex at a corner needs

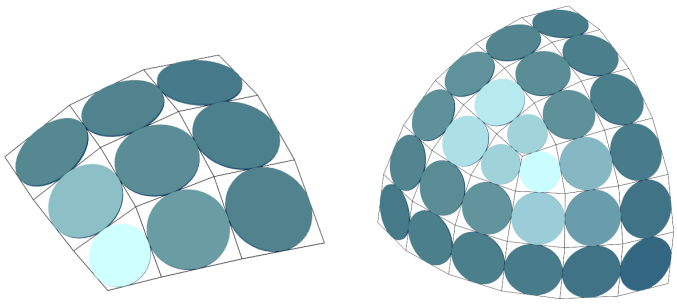


FIGURE 2.9. Umbilic point under the discrete Gauss map

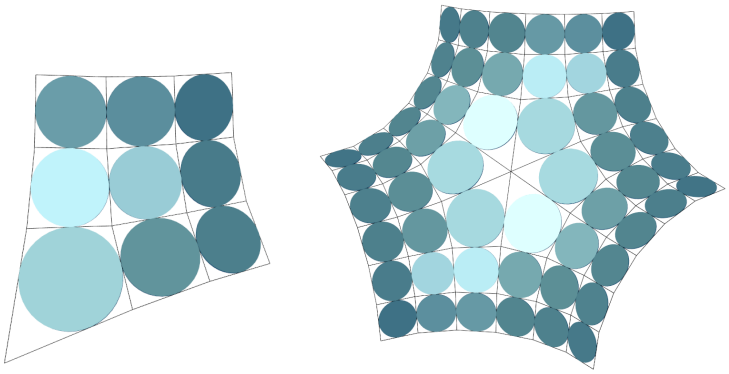


FIGURE 2.10. Umbilic point with six discrete curvatures lines originating

us to add $\pi$ to the respective face. Figure 2.9 shows an umbilic point at the corner of a quad grid mesh. Figure 2.10 is the corresponding discrete minimal surface. Here six parts are used to form the umbilic point.

**2.1.4. Visualization.** A discrete minimal surface consists of conformal squares, where each four of them form a quadrilateral with an inscribed circle. The edges between centers and the intersections of touching circles are a discrete analog of curvature lines. Drawing only the circles is enough to provide information about the shape of surface. There is another way to visualize the properties of the surface. The following Lemma gives the motivation for this.

LEMMA 2.1.5. ***(Touching Coins Lemma)*** *Whenever four circles in* 3*-space touch cyclically but do not lie on a common sphere, they intersect the sphere which passes through the points of contact orthogonally.*

Now additionally to the circles we can draw spheres intersecting the circles orthogonally. In Figure 2.11 the circles and the spheres of a part of the Schwarz P surface are drawn.

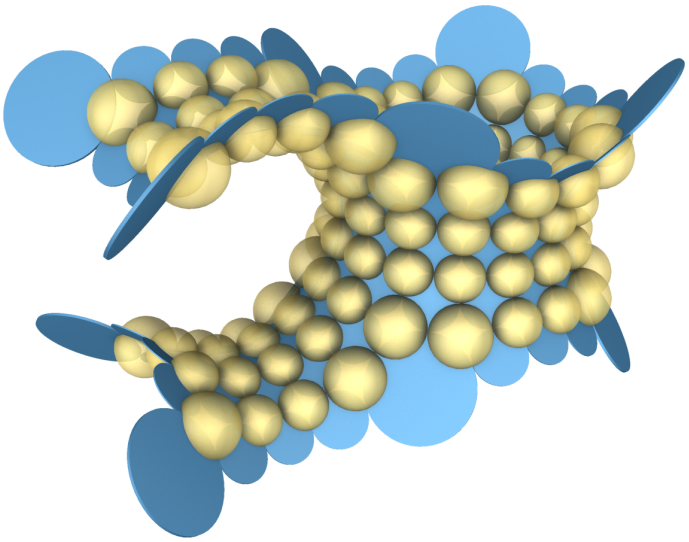


FIGURE 2.11. Orthogonally intersecting circles and spheres

## 2.2. Implementation

We use the same algorithm as in the Koebe project to minimize the spherical functional. Like in the Euclidean case, we use Newton's method for optimization. Since the spherical circle pattern functional is non-convex, we have to start at a good initial value. Setting all $\rho$ to $-5$ has proved to be a good guess.

Once the correct radii for the circle patterns are computed, we determine the positions on the sphere by an iterative layout algorithm. This algorithm is essentially the same algorithm like in the Euclidean case. The only difference is the rotation and the scaling which is carried out on the sphere. Here we use the Möbius class of the jtem package [**TBa**] to perform the spherical rotation and the scaling.

## 2.3. The Program

The program is designed to construct, visualize, and export discrete minimal surfaces. After the program has been started the main window is displayed (Figure 2.12). The leftmost section is the action library. It contains the parts and tools of the creation process. You can double-click or drag actions from the library to the creation plan in the center of the main window. The creation plan is a list of actions which will be processed from top to bottom. It is necessary to add the "Load Combinatorics" action at the top of the creation plan. Below the creation plan there are buttons for deleting or for changing the order of the actions. The right panel shows options for the currently selected action of the creation plan. The lower section provides

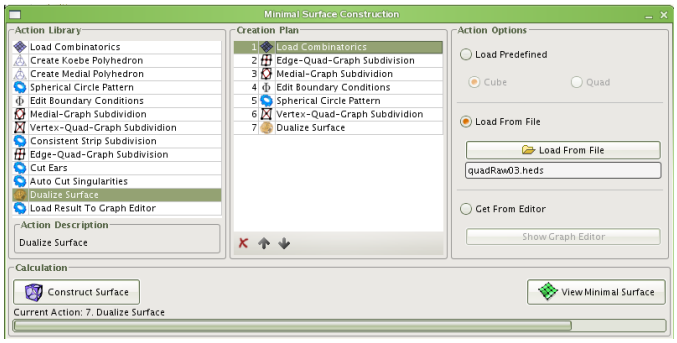



FIGURE 2.12. The minimal surface program

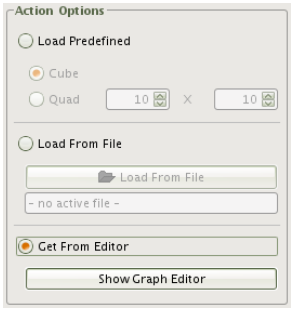



FIGURE 2.13. Load combinatorics options

the button to start the calculation process and shows the currently active action as well as a process bar.

There are two ways to construct a discrete minimal surface. The first constructs a complete Koebe polyhedron using the Euclidean circle pattern functional and dualizes it to create a minimal surface. The second method constructs parts of Koebe polyhedra via the spherical functional. Thereby boundary conditions can be specified.

**2.3.1. Complete Koebe Polyhedron Method.** To illustrate this method we describe the creation of the Scherk tower as proposed in [**BHS06**]. We start with the combinatorics and add the "Load Combinatorics" action to the creation plan. From the options section we choose the "Get From Editor" option and press the "Show Graph Editor" button to bring up the editor (Figure 2.13). Choose the edge tool ( ) from the tool bar and draw the combinatorics of the Scherk tower (Figure 2.14). Now drag some vertices to create an embedded version of the graph (Figure 2.15). We have to add faces to get a ready-to-calculate structure ( ).

The next step in the creation process is the quad-graph of the Koebe polyhedron. Technically we compute the circle pattern on the sphere via the "Create Medial Polyhedron" action ( ). Then we subdivide using a "Vertex-Face-Subdivision" rule ( ) to arrive at the quad-graph of the Koebe polyhedron.

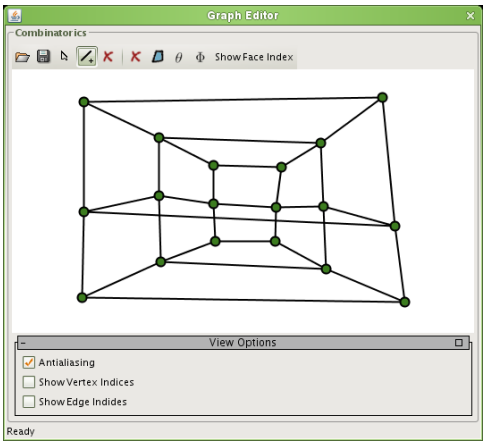



FIGURE 2.14. Combinatorics editor and the Scherk tower graph

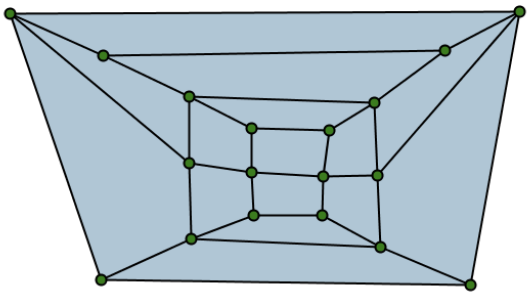


FIGURE 2.15. Embedded Scherk tower combinatorics

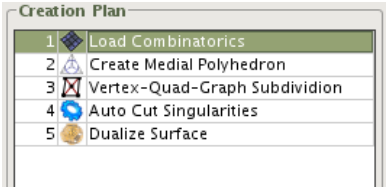



FIGURE 2.16. The Scherk tower creation plan

Finally we need to dualize the surface. Since the dualization step requires the assignment of consistent edge labels (Proposition 2.1.3), we have to introduce boundaries. The "Auto Cut Singularities" cuts the surface along paths which include the singularities. Thus the edge labeling at an odd-valent vertex becomes valid due to the double labeling of the split edge. For the Scherk tower creation we omit the insertion of a double vertex as proposed in [**BHS06**] and don't calculate the asymptotic planes. At last add the "Dualize Surface" ( ) action to the creation plan.

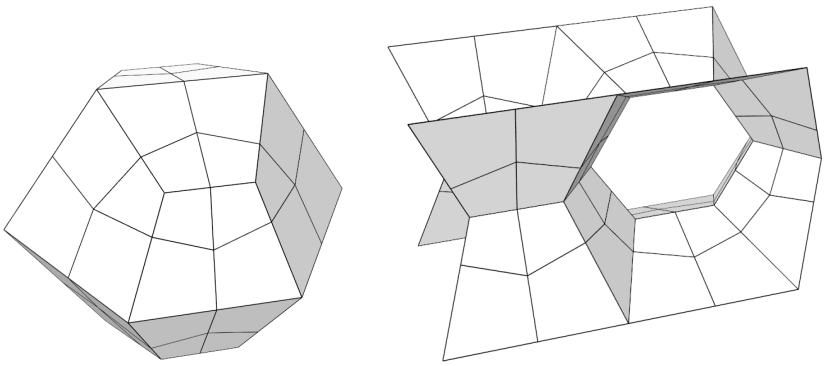


FIGURE 2.17. The Koebe polyhedron quad-graph and the dual discrete minimal surface of the Scherk tower using a coarse discretization

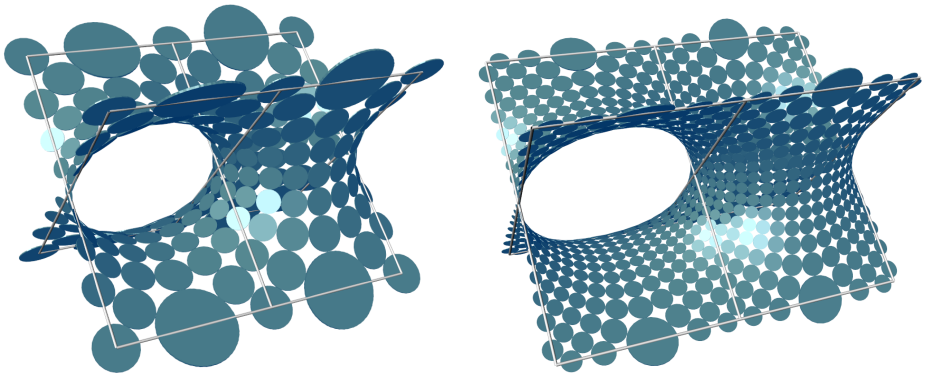


FIGURE 2.18. Schwarz' CLP surface

Figure 2.16 is the final plan for the Scherk tower using the "Complete Polyhedron Method". Press the "Construct Surface" button to invoke the calculation and show the surface rendering. The resulting surface and its Koebe polyhedron are shown in Figure 2.17.

Having dualized the Koebe polyhedron, one obviously acquires only a part of the desired surface. Since the Gauss map's image is a multi-cover of the sphere being branched at the singularities, one can reflect the surface about its singular points to continue smoothly. The Scherk tower is simply periodic so it can be continued in one direction. In the viewing window there are tools for point reflection, reflection at a plane, and 180°-rotation about a straight line.

**2.3.2. Spherical Calculation Method.** This method is a little more involved due to the different boundary conditions. What came for free in the last method has to be specified precisely here. We describe the creation of Schwarz's CLP surface (Figure 2.18) .

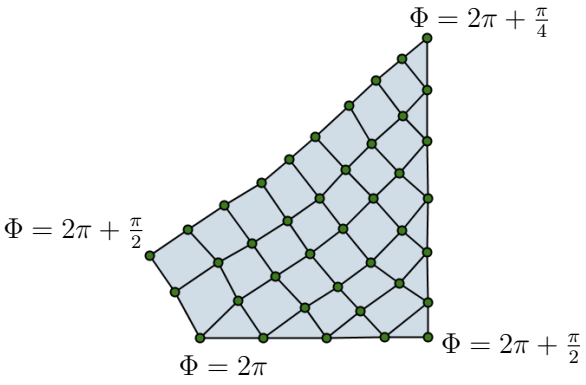



FIGURE 2.19. The medial combinatorics and boundary angles of the Schwarz CLP surface

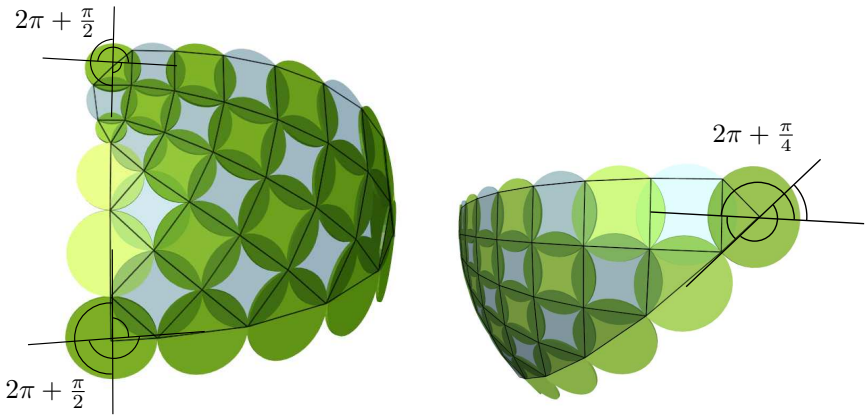



FIGURE 2.20. Boundaries and umbilic points of the Schwarz CLP surface under the discrete Gauss map

The medial combinatorics and the angles of the Schwarz CLP surface are shown in Figure 2.19. We start with the medial graph and omit the medial subdivision that was part of the "Complete Koebe Method", as the subdivision step cannot handle the boundary correctly. In Image 2.20 the angles are shown schematically. The $\Phi = 2\pi$ angle in Figure 2.19 produces a boundary which is a part of a great circle while changing the combinatorial style from curvature line to asymptotic line. This behavior produces a 180° turn of the boundary since the polyhedron is the discrete analog of the Gauss map and the asymptotic line will be perpendicular to the plane of the curvature line.

To specify the angles during calculation add the "Edit Boundary Condition ($\Phi$)" action to the creation plan. Alternatively we can enter the angles directly using the tool in the graph editor.

Like before we now have to create the Koebe polyhedron by "Edge-Face-Subdivision" and dualize the result to get a part of the Schwarz CLP surface. Here we can choose either of the two possible polyhedra. The options of the

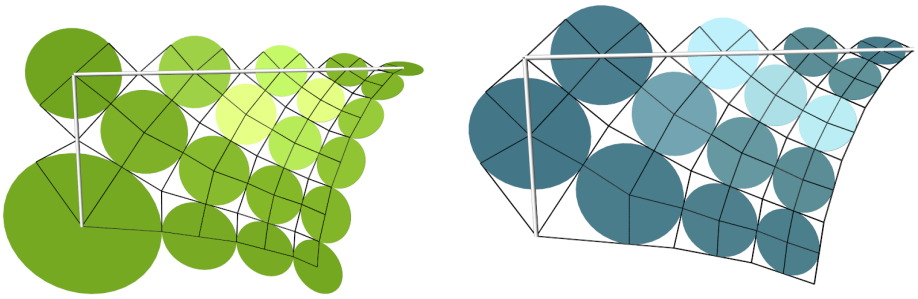


FIGURE 2.21. A fundamental piece of the Schwarz CLP surface

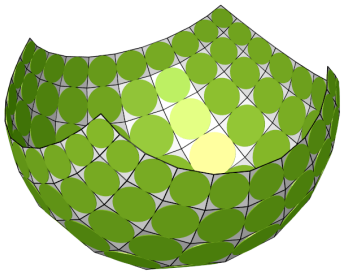


FIGURE 2.22. A part of the Koebe polyhedron of Enneper's surface

subdivision action decide whether to create circles for vertices or for faces of the combinatorics. Both versions of a fundamental piece of the Schwarz CLP surface are shown in Figure 2.21.

Again Schwarz' CLP surface can be continued smoothly by reflection at the planes of the boundary curvature lines or 180°-rotation about the boundary asymptotic lines.

## 2.4. Examples

To create a discrete analog of a smooth minimal surface, the combinatorics of the curvature lines of a fundamental piece is required. The examples are constructed using the spherical method, so the combinatorics of asymptotic lines of the surface is needed. If the surface has umbilic points, then we have to know the angle at which the curvature lines meet at this point.

**2.4.1. Enneper's Surface.** The circle pattern for Enneper's surface is the stereographic projection (Equation 7) of the plane quadrilateral grid with circles of the same radius (Figure 2.22). Figure 2.23 shows the minimal surface of two different parts of this circle pattern. The program has an extra action for Enneper type circle patterns on the sphere ( ).

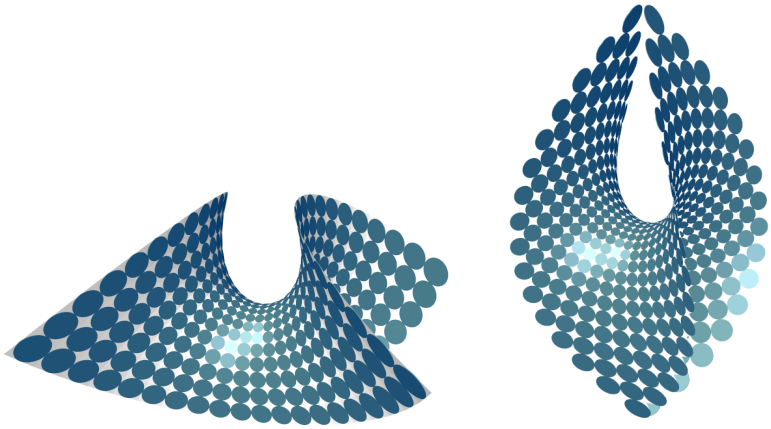


FIGURE 2.23. Enneper's surface

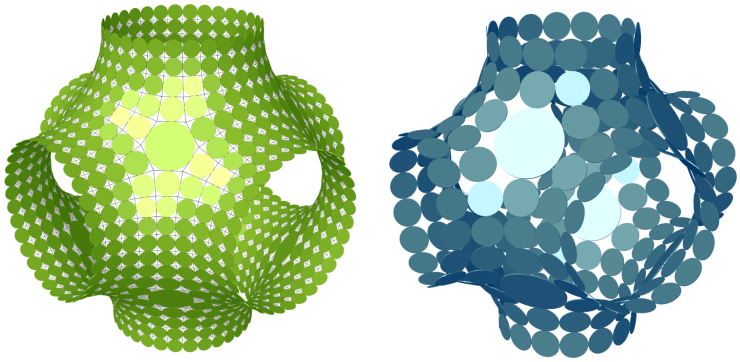


FIGURE 2.24. The Schwarz P surface in different discretizations

**2.4.2. Schwarz' P Surface.** A fundamental piece of the Schwarz P surface is a square quad grid with three right angles and $\frac{\pi}{3}$ at the remaining corner. At this umbilic point six curvature lines join. Figure 2.9 and 2.10 show the circle pattern and the surface at this point. Figure 2.24 shows the usual visualization of the surface. Here the surface can be smoothly continued by translation in either of the spatial directions.

Figure 2.25 shows unsymmetric versions created with the complete polyhedron method. Here we use different discretization resolutions in all of the directions (left) and in two directions (right).

**2.4.3. The Scherk Tower.** The creation of Scherk's tower with the polyhedron method is described in Section 2.3.1. There are some other ways to approximate Scherk's tower using the spherical method.

The surface is simply periodic and asymptotic to two intersecting planes. The angle between the planes creates a 1-parameter family of surfaces. The image of the Gauss map of a fundamental piece of the Scherk tower is a hemisphere with four special points on the boundary. The special points

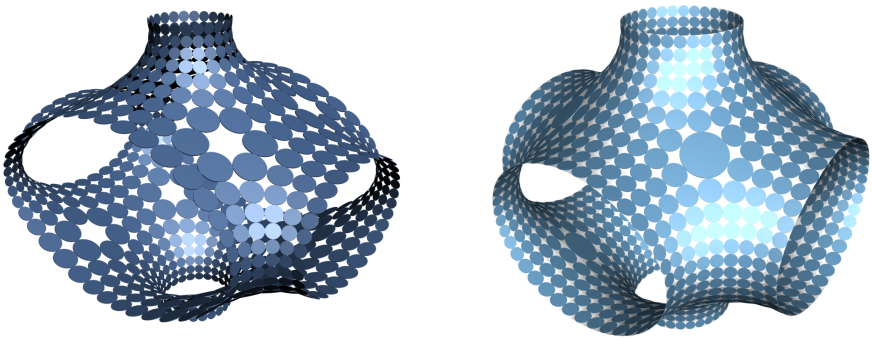


FIGURE 2.25. Unsymmetric Schwarz P surface

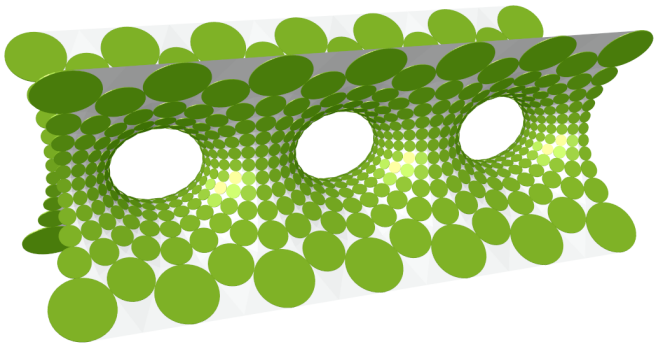


FIGURE 2.26. Scherk's tower symmetric version

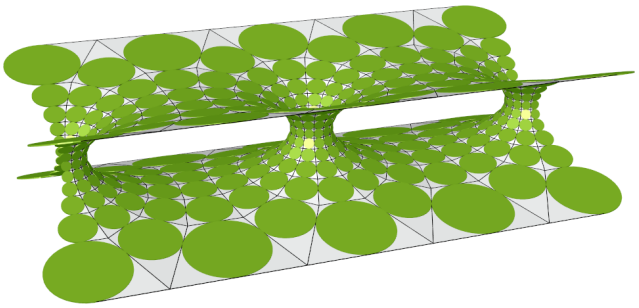


FIGURE 2.27. Scherk's tower unsymmetric version

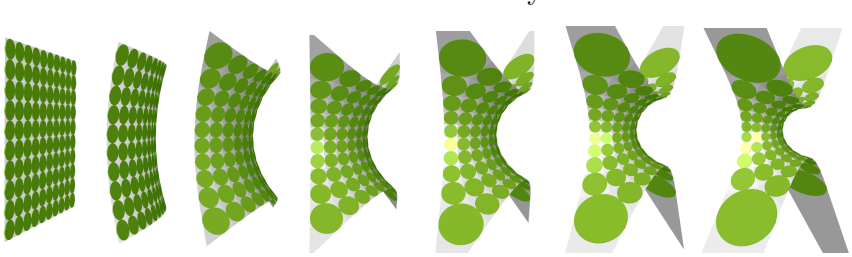


FIGURE 2.28. From the plane to Scherk's tower

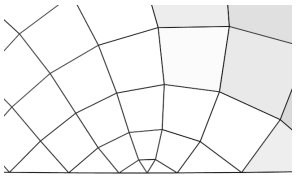


FIGURE 2.29. Scherk tower, medial combinatorics of a singularity

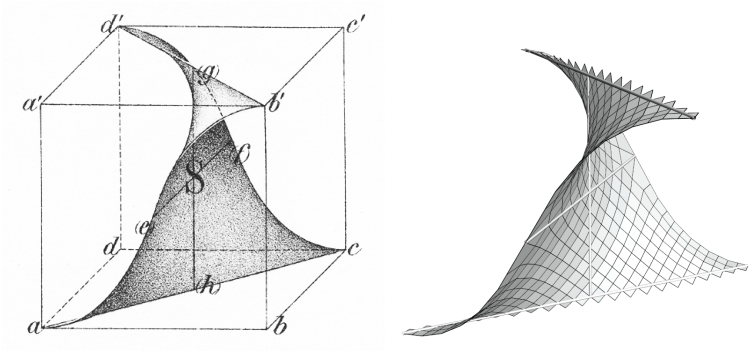



FIGURE 2.30. Gergonne's surface by H. A. Schwarz [**Sch90**] (left) and discrete analog (right)

correspond to points at infinity. Figure 2.28 shows an attempt to cope with this behavior. As the angles at the corners approach $\pi$ the vertices go to infinity and form the asymptotic planes. These images are calculated using the default square grid combinatorics and the "Medial Subdivision" action. The resulting double edges at the corners have an angle of $\Phi = 2\pi - \frac{\pi}{2}$ to get the plane and theoretically $\Phi = 2\pi$ to obtain the Scherk tower. The rightmost image of Figure 2.28 was created with an angle of $\Phi = 6.24$. The angle between the asymptotic planes is controlled by the ratio of columns and rows of the combinatorics. In Figure 2.28 we use the symmetric version with $m = n$.

We cannot calculate the surface for $\Phi = 2\pi$ because dualization maps the infinitely small face on the hemisphere boundary to an infinitely large face that forms a part of the asymptotic plane. To overcome this problem we can use a different combinatorics. We delete the decreasing face from the graph and connect the remaining faces. A part of the resulting combinatorics is shown in Figure 2.29. This combinatorics however results in a three-valent vertex after quad-graph generation. Use the "Auto Cut Singularities" action and the "To Boundary Only" option to get a dualizable surface.

The unsymmetric example in Figure 2.27 has parameters $m = 12$, $n = 7$.

**2.4.4. Gergonne's Surface.** Gergonne's surface is a triply periodic surface whose fundamental piece divides the cube into two equal regions (Figure 2.30). We construct only a fourth of this part as the symmetry allows to rebuild the surface by reflection. The height of the cube is controlled by the ratio of rows and columns in the combinatorics (Figure 2.32). The most

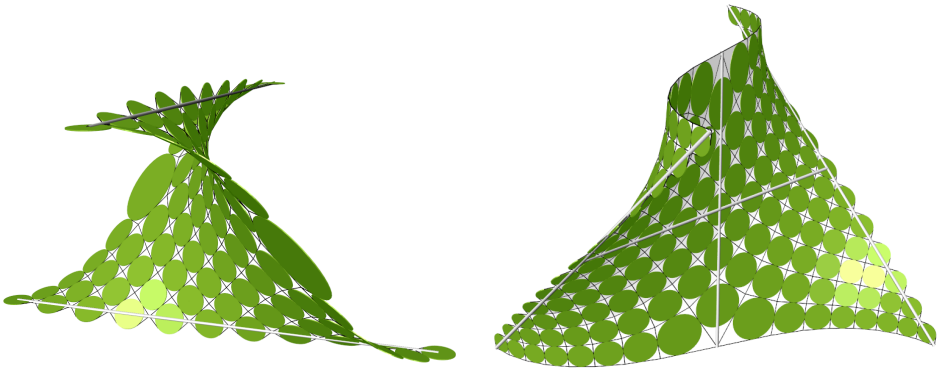


FIGURE 2.31. Gergonne's Surface $m < n$ (left) and $m = n$ (right)

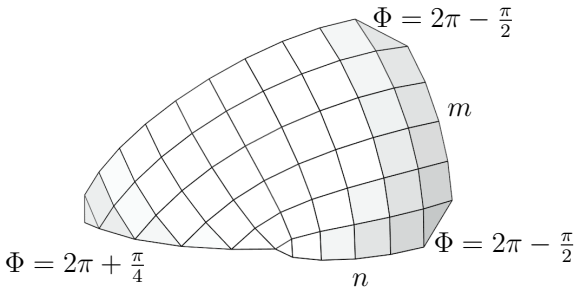



FIGURE 2.32. The medial combinatorics and angles of Gergonne's surface

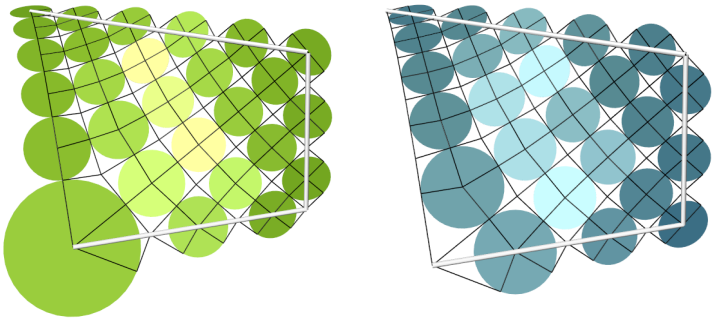


FIGURE 2.33. Fundamental piece of Gergonne's surface

symmetric surface is the case $m = n$, in Figure 2.31 it is $m < n$ on the left and $m = n$ on the right.

**2.4.5. Schwarz' D Surface.** The Schwarz D surface is a triply periodic surface whose fundamental piece is bounded by the edges of a cube. It has an umbilic point at the center (Figure 2.34 right). Again we exploit the symmetry to reduce the complexity of the combinatorics for the calculation. Here we introduce a singular point that does not lie at a corner of the

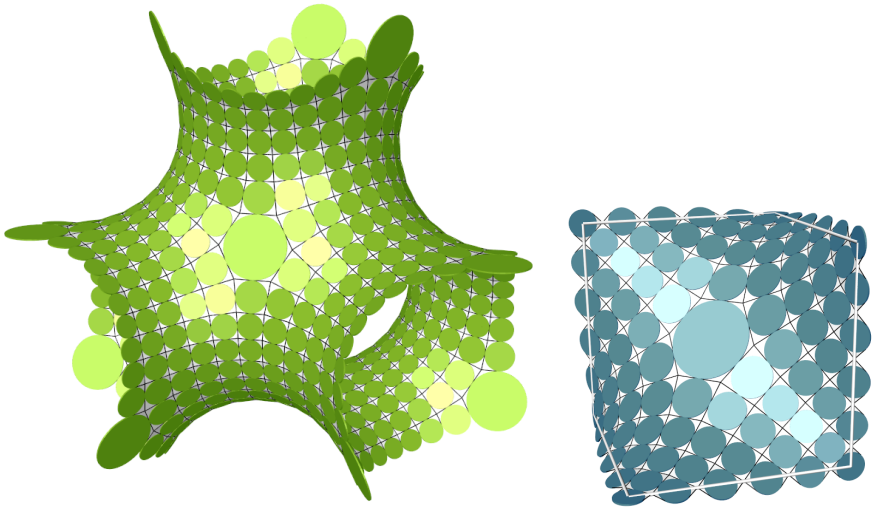


FIGURE 2.34. Schwarz' D Surface, fundamental piece (right) and larger surface obtained by reflection (left)

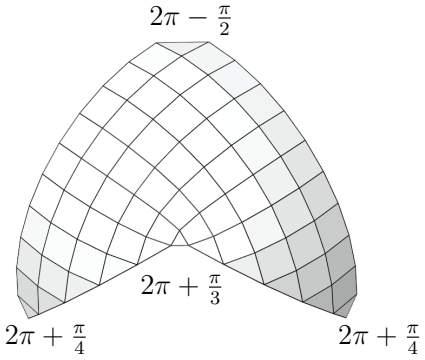



FIGURE 2.35. Medial combinatorics and angles of the Schwarz D surface

graph (Figure 2.35). There are no ratio parameters in this surface, only the resolution of the discrete surface can be defined.

The Schwarz D surface can also be created using the "Complete Polyhedron" method. Here the combinatorics is a subdivided tetrahedron.

**2.4.6. Schwarz' H Surface.** Like the P and D surface the Schwarz H surface is triply periodic. A fundamental piece is bounded by two parallel equilateral triangles. The distance of the triangles creates a 1-parameter family of surfaces. The medial combinatorics is similar to Gergonne's surface. Gergonne's quad-graph is the Schwarz H medial graph and vice versa. Figure 2.37 shows the combinatorics of the circle pattern on the sphere. The parameters in Figure 2.36 are $n = 3$, $m = 5$ (left) and $n = 5$, $m = 3$ (right).

**2.4.7. Neovius' Surface.** Neovius' surface is triply periodic. The fundamental piece is similar to the one of the Schwarz P surface only the angles

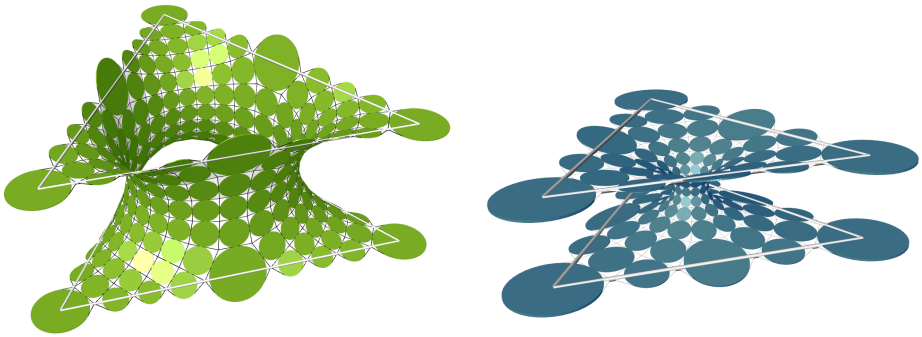


FIGURE 2.36. Schwarz' H surface

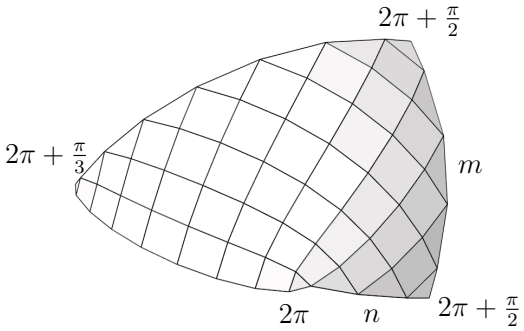



FIGURE 2.37. Medial combinatorics and angles of the Schwarz H surface

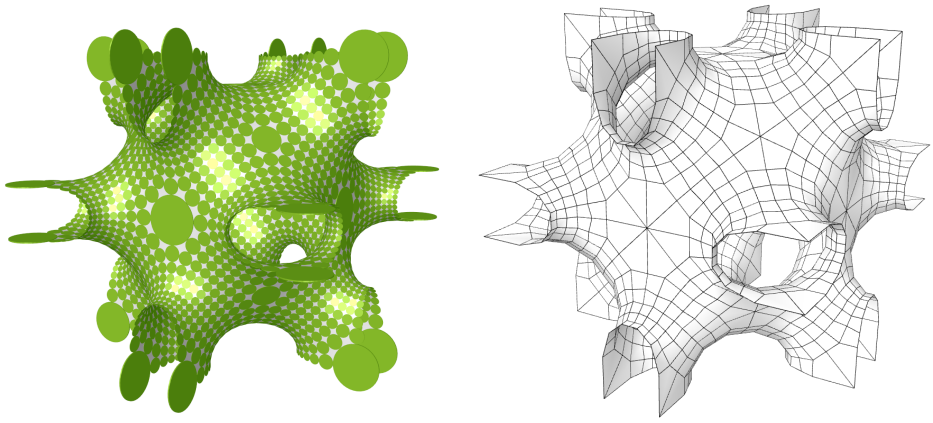


FIGURE 2.38. Neovius' surface

are different (Figure 2.39). There are three umbilical points at the corners of a fundamental piece after reflection. Two of those have eight curvature lines meeting and the third has six (Figure 2.40).

**2.4.8. Quadrilateral Frame.** Given the aspect ratio and the angles of a quadrilateral, what is the minimal surface with this boundary? The curvature line pattern is easy to draw since the four asymptotic lines are

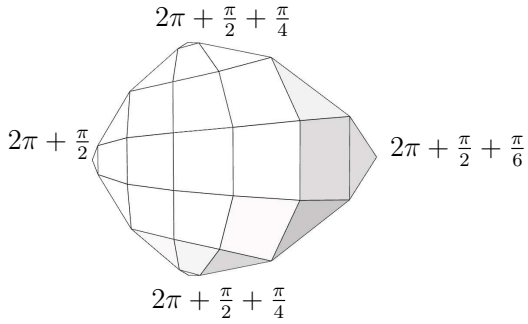



FIGURE 2.39. Circle pattern combinatorics of Neovius' surface

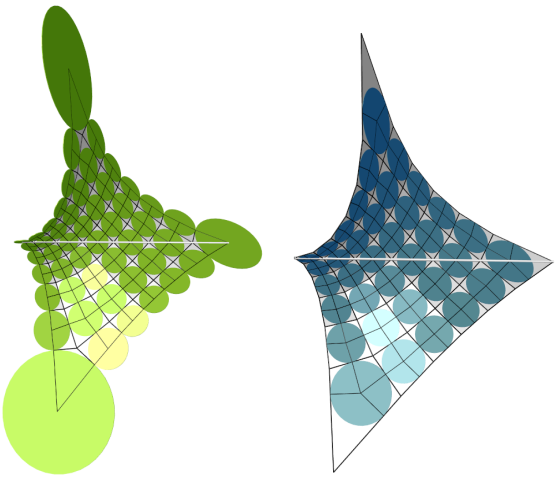


FIGURE 2.40. Fundamental piece of Neovius' surface

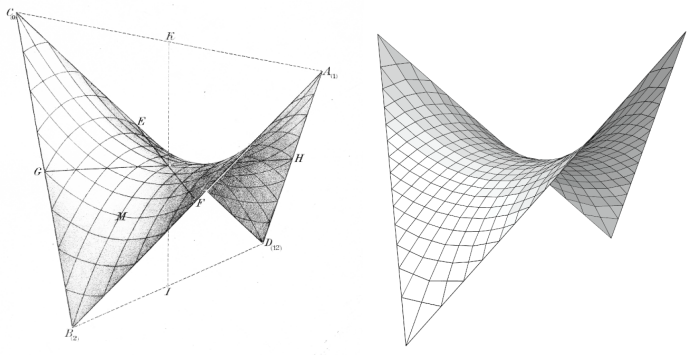


FIGURE 2.41. Quadrilateral minimal surface by H. A. Schwarz [**Sch90**] (left) and discrete analog (right)

already fixed. The medial combinatorics, which is also the asymptotic line pattern, is shown in Figure 2.43.

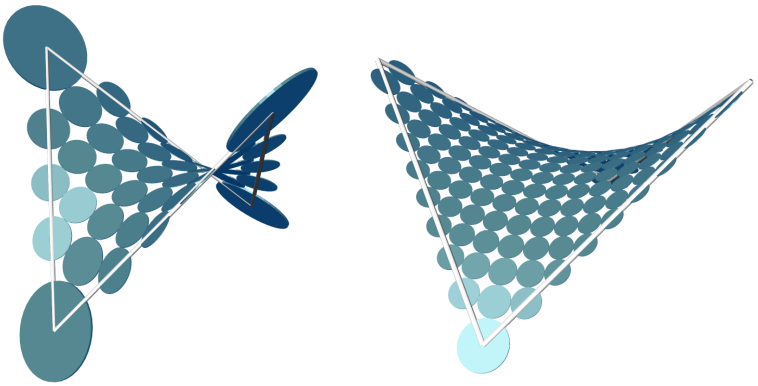


FIGURE 2.42. Quadrilateral boundary frame

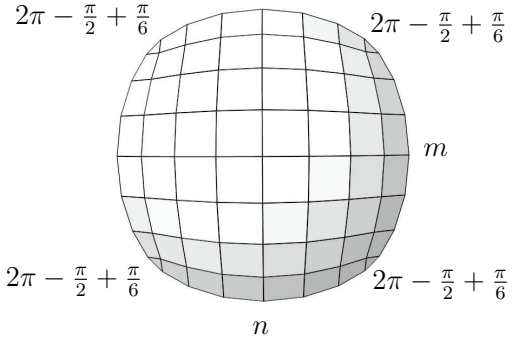



FIGURE 2.43. Medial combinatorics of the quadrilateral surface

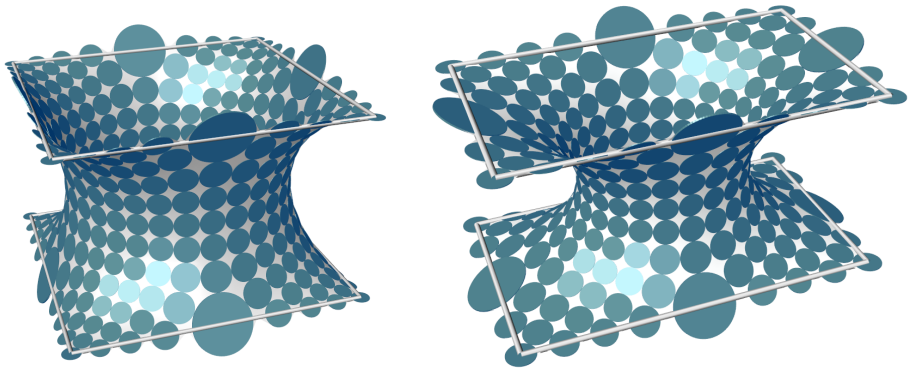


FIGURE 2.44. Schoen's I-6 surface (left) and generalized version (right)

**2.4.9. Schoen's I-6 Surface.** Take two parallel copies of a rectangle. What is the minimal surface bounded by these polygons. By construction Schoen's I-6 surface has three planes of symmetry. Thus we need to calculate only an eighth of the surface and obtain the remainder by reflection if we take squares to start with. The medial combinatorics of Schoen's I-6 surface is the

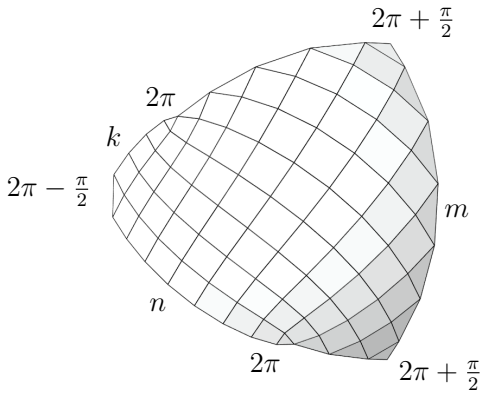



FIGURE 2.45. Medial combinatorics of the generalized Schoen I-6 surface.

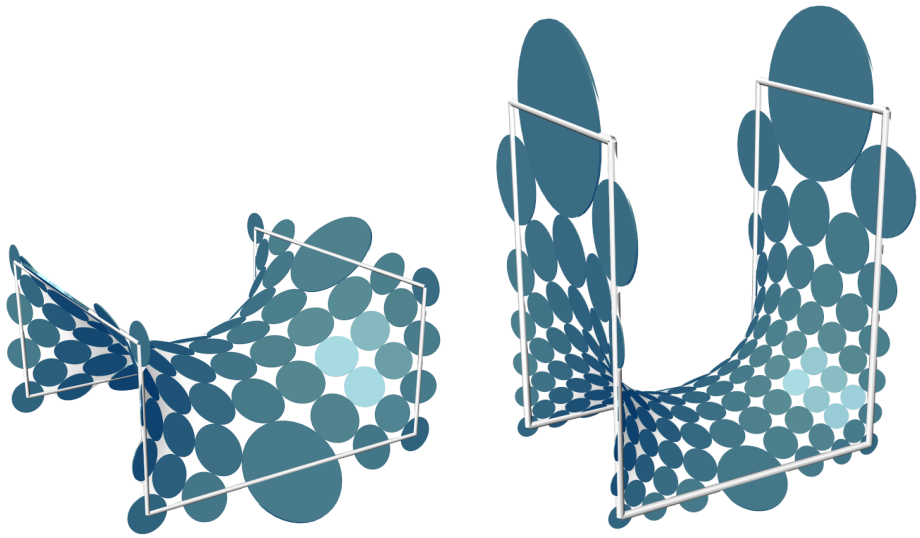


FIGURE 2.46. Cuboid boundary polygons

same as for Schwarz' H surface. Just the angle on left becomes $2\pi + \frac{\pi}{4}$ to get a fundamental piece of Schoen's surface. If we want a general rectangle we have to use a different combinatorics. Figure 2.45 shows the combinatorics of a general Schoen I-6 surface. The ratio of the rectangle is controlled by the ratio of $n$ and $k$. The distance of the rectangles depends on $m$. The right example in Figure 2.44 has $n = 9$, $m = 7$ and $k = 5$.

**2.4.10. Cuboid Boundary Frame.** Figure 2.46 shows discrete minimal surfaces with a boundary of a cuboid. These surfaces have two planes of symmetry and we need to calculate a fourth to get a fundamental piece. Figure 2.47 illustrates the corresponding combinatorics.

**2.4.11. Catenoid Approximations.** A catenoid can be constructed as the minimal surface spanned by two parallel circles of the same radius.

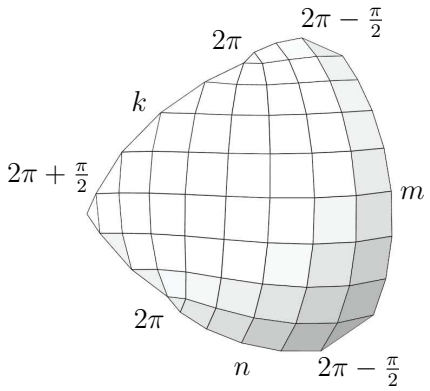



FIGURE 2.47. Medial combinatorics of the cuboid boundary surface

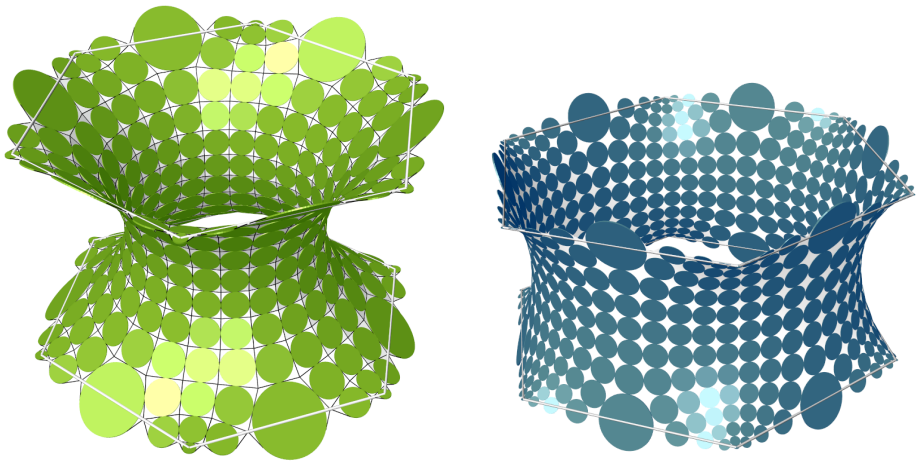


FIGURE 2.48. Catenoid approximations

We approximate these circles by using regular $n$-gons and the medial combinatorics of the Schwarz H surface or the Schoen I-6 surface. In fact the Schwarz H or the Schoen I-6 surface are approximations to the discrete catenoid as well. Here the ratio of $m$ and $n$ specifies the calculated part of the catenoid.

For a generalization of this approximation process we can use non-regular polygons as a discrete analog of the circle (Figure 2.49). Here we use the combinatorics of the generalized Schoen I-6 surface.

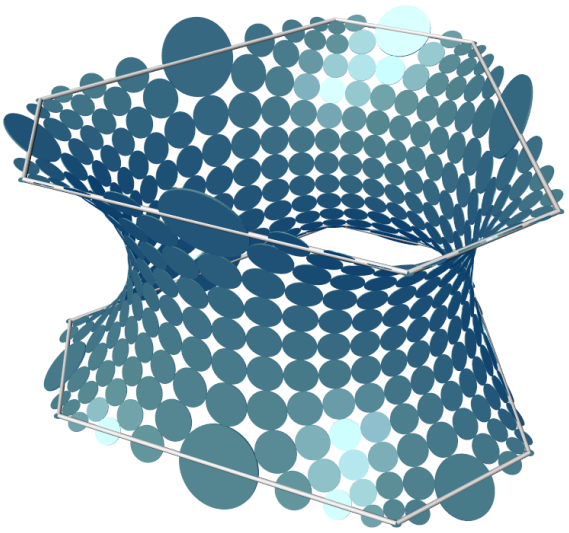


FIGURE 2.49. Non-regular catenoid approximation

CHAPTER 3

# Alexandrov's Theorem

The article [**BI06**] provides a new poof of Alexandrov's theorem. We implemented an algorithm following the constructive proof of the article. In the end the user could enter some metric via editor or file and calculate the corresponding polyhedron which realizes this metric on its boundary. In the first section we summarize the theorems and definitions needed for the implementation. We give the necessary equations to calculate angles and lengths.

## 3.1. Theory

The central theorem of this part is Alexandrov's theorem.

THEOREM 3.1.1. *Let $M$ be a sphere with a convex Euclidean polyhedral metric. Then there exists a convex polytope $P \subset \mathbb{R}^3$ such that the boundary of $P$ is isometric to $M$. Besides, $P$ is unique up to a rigid motion.*

The goal of our implementation is to construct the polytope for the given metric. Fist we have to clarify some definitions to understand how we can represent such a metric structure. In [**BI06**] we find the corresponding definitions:

DEFINITION 3.1.2. An Euclidean polyhedral metric on a surface $M$ is a metric structure such that any point $x \in M$ possesses an open neighborhood $U$ with one of the following properties. Either $U$ is isometric to a subset of $\mathbb{R}^2$, or there is an isometry between $U$ and an open subset of a cone with angle $\alpha \neq 2\pi$ such that $x$ is mapped to the apex. In the first case $x$ is called a regular point, in the second case it is called a singular point of $M$. The set of singular points is denoted by $\Sigma$.

If for any $x \in \Sigma$ the angle at $x$ is less that $2\pi$, then the Euclidean polyhedral metric is said to be convex.

Usually one defines such a metric with the help of a triangulation of the sphere which has a length assigned to each edge. Here the triangle inequality has to hold for every triangle of the triangulation. Then a point in the interior of a triangle is clearly surrounded by a part of $\mathbb{R}^2$. The vertices of the triangles play the role of the cone points. The cone angle is just the sum of angles in the triangles at this point. Only vertices with a cone angle $\neq 2\pi$ are considered, then these vertices are the singularities of the metric. A point on an edge of the triangulation is contained in an Euclidean neighborhood as well since we don't change distances if we flatten this edge.

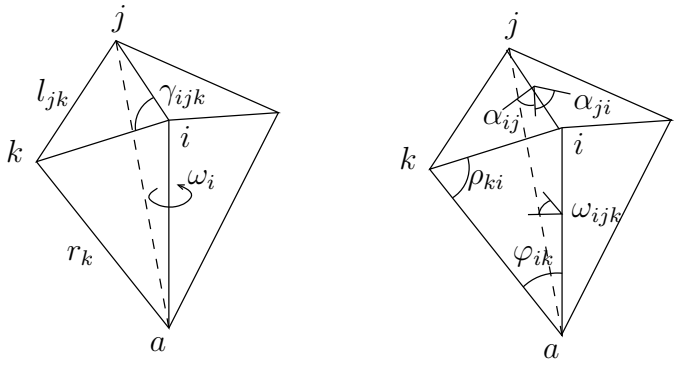



FIGURE 3.1. Angles and lengths of a generalized polytope

We can easily provide an editor or a file format for such a representation, see the program section for a description.

In the article such a construction is called a *geodesic triangulation.* Using this object we can define a structure essential to the construction process.

The singular points in $\Sigma$ are denoted by $i, j, k, \ldots$ . An edge connecting vertex $i$ and vertex $j$ is denoted by $ij$. Consequently we write $ijk$ denoting the triangle with the vertices $i$, $j$, and $k$. Assign radii $r = r_1, \ldots, r_n$ such that there exist pyramids with base $ijk$ and edge lengths $r_i, r_j, r_k$. These pyramids are then glued following the combinatorics of the triangulation. Since consecutive radii end at the same peak, all radii are glued to one point called the apex of the generalized polytope. See Figure 3.1 for notations.

Let $T$ be a geodesic triangulation.

DEFINITION 3.1.3. A generalized polytope with boundary $M$ is an equivalence class of couples $(T, r)$ as above, where two couples are equivalent if and only if there is an isometry between the resulting polyhedra which is identical on the boundary and maps the apex to the apex. A generalized convex polytope is a generalized polytope such that $\theta_{ij} \leq \pi$ for any edge $ij$ in some (and hence in any) associated triangulation.

A generalized convex polytope is uniquely defined by it's radii, see [**BI06**]. Let $\mathcal{P}(M)$ be the space of generalized convex polytopes. The radii of the polytope then define a parametrization on this space. But not all possible radii create a generalized convex polytope, thus there are constraints on the set of possible radii. Therefore we identify $\mathcal{P}(M)$ with the set of admissible radii in the remainder of this text. We are now able to define the curvature of a generalized convex polytope.

For an interior edge connecting a vertex $i$ and the apex $a$ the curvature at this edge is defined as

$$\kappa_i := 2\pi - \omega_i.$$

Furthermore define

$$\theta_{ij} := \alpha_{ij} + \alpha_{ji}.$$

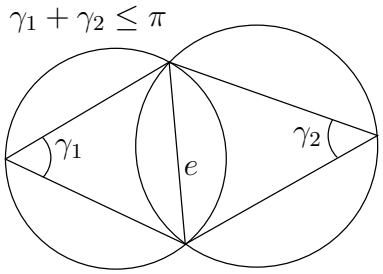



FIGURE 3.2. Local Delaunay condition

The curvature of a generalized polytope is a vector valued function which takes the radii as arguments. It is

$$\kappa : \mathcal{P}(M) \to \mathbb{R}^n$$

where $n$ is the number of vertices in the triangulation.

To compute the curvature at a given vertex we need expressions for $\omega_i$. It is

$$\cos \omega_{ijk} = \frac{\cos \gamma_{ijk} - \cos \rho_{ij} \cos \rho_{ik}}{\sin \rho_{ij} \sin \rho_{ik}}$$

and $\omega_i$ is the sum of all angles at edge $ia$. The formula for $\rho$ is

$$\cos \rho_{ij} = \frac{l_{ij}^2 + r_j^2 - r_i^2}{2 l_{ij} r_j}.$$

The Jacobian of the curvature is given by

$$\begin{aligned} \frac{\partial \kappa_i}{\partial r_j} &= \frac{\cot \alpha_{ij} + \cot \alpha_{ji}}{l_{ij} \sin \rho_{ij} \sin \rho_{ji}} \\ \frac{\partial \kappa_i}{\partial r_i} &= -\sum_{i \neq j} \cos \varphi_{ij} \frac{\partial \kappa_i}{\partial r_j}, \end{aligned}$$

where

$$\cos \alpha_{ij} = \frac{\cos \rho_{ik} - \cos \gamma_{ijk} \cos \rho_{ij}}{\sin \gamma_{ijk} \sin \rho_{ij}}$$

and $\frac{\partial \kappa_i}{\partial r_j} = 0$ if $ij$ is not an edge of the triangulation. If there are multiple edges or loops, then the formulas have to be modified, see [**BI06**].

The algorithm described in this work operates on generalized convex polytopes. If for a generalized convex polytope $\kappa_i = 0$ holds for all $i$, the polytope can be embedded into $\mathbb{R}^3$ and thus be visualized as a convex polytope. This is also the idea of the algorithm. Starting with sufficiently large and equal $r_i$, we modify these radii so that the curvature decreases for all internal edges. To start with a convex polytope we need a special triangulation, the Delaunay triangulation.

DEFINITION 3.1.4. A geodesic triangulation of a surface with a polyhedral metric is called a *Delaunay triangulation* if every edge is locally Delaunay. An edge is called locally Delaunay if the following condition holds

$$\gamma_1 + \gamma_2 \leq \pi$$

where $\gamma_1$ and $\gamma_2$ are angles in the adjacent triangles of the edge (Figure 3.2).

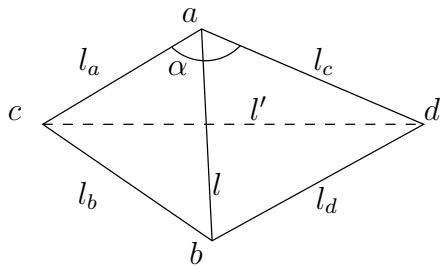



FIGURE 3.3. Flipping $ab$ to $cd$

**Algorithm 1** DELAUNAYTRIANGULATION

**Require:** stack $S$ with all marked edges
**while** $S$ is non-empty **do**
  pop $ab$ from $S$ and unmark it
  **if** $ab$ not locally Delaunay **then**
    FLIP $ab$ to$cd$
    **for** $xy \in ac, cb, bd, da$ **do**
      **if** $xy$ is not marked **then**
        mark $xy$ and push it on $S$
      **end if**
    **end for**
  **end if**
**end while**

There is an algorithm described in [**Ede01**] pp. 7-9 that transforms an arbitrary triangulation into a Delaunay triangulation (Algorithm 1). The algorithm uses the flip transformation. The length of the flipped edge is then

$$l'^2 = l_a^2 + l_d^2 - 2l_b l_c \cos\alpha. \tag{10}$$

Figure 3.3 explains the notations at the affected triangles.

If $T$ is a Delaunay triangulation, there is an $R > 0$ such that the generalized polytope with radii $(R, \ldots, R)$ is convex. For a convex polytope it is always

$$\operatorname{rank}\frac{\mathrm{d}\kappa}{\mathrm{d}r} = n$$

if the following condition holds

$$0 < \kappa_i < \delta_i \ \forall_i \tag{11}$$

where $\delta_i$ is the angle defect at vertex $i$, it is $\delta_i := 2\pi - \sum_{jk \in \mathrm{lk}i} \gamma_{ijk}$. This is important as we want to solve equations of the form $\kappa(\tilde{r}) = \tilde{\kappa}$ by linearization.

It is shown that there is always a deformation $\kappa \rightsquigarrow (1-\delta)\kappa$ that reduces the curvature at every vertex such that the generalized polytope $(T, r)$ exists and is convex up to a single edge which may be locally concave. This edge can be made convex by flipping and resizing using Equation 10. Here $\delta$ is some factor in the range $(0, 1)$, thus the condition 11 stays valid during deformation. In symmetric cases there may be situations in which there

are more than one edges getting concave at the same time. These cases can be handled by a slightly modified version of the algorithm, see the implementation section.

The algorithm runs now as follows: From the given triangulation $T$ and the assigned edge lengths, which are assumed to define a convex polyhedral metric, we calculate the Delaunay triangulation and determine radii $R$ such that the generalized polytope $(T, R)$ is convex. In [**BI06**] it is shown that this is always possible. Then calculate the curvature $\kappa(r)$ and choose a $\tilde{\kappa} < \kappa(r)$. Solve the equation $\kappa(\tilde{r}) = \tilde{\kappa}$ with a Newton solver. If the generalized polytope $(T, \tilde{r})$ is convex, then we can proceed and set $r \leftarrow \tilde{r}$. If not, but there is exactly one concave edge, then we may fix the convexity by flipping this edge and proceed normally. If we find more than one edges to be locally concave, then we have to choose an $\tilde{\kappa}$ which is nearer to $\kappa(r)$ and repeat the last steps. [**BI06**] shows that there is always a path that takes the curvature $\kappa$ to zero while preserving the convexity of the generalized polytope.

Once arrived at $\kappa \approx 0$, we are no longer forced into imagining some abstract polytope. We can start to construct (embed) a convex polytope that realizes the input metric on its boundary. The construction algorithm is an iterative calculation which starts with an edge and enumerates the boundary by successive determination of the opposite vertex.

### 3.2. Implementation

**Algorithm 2** AlexandrovsPolyhedron

**Require:** $T'$, $\varepsilon$, $\delta_{max}$
1: $T \leftarrow$ DelaunayTriangulation($T'$)
2: $r \leftarrow (R, \ldots, R)$, such that $(T, r)$ is convex
3: $\delta \leftarrow \delta_{max}$
4: $\tilde{\kappa} \leftarrow (1 - \delta) \cdot \kappa(r)$
5: **while** $\| \kappa(r) \| > \varepsilon$ **do**
6:  **if** $\kappa(\tilde{r}) = \tilde{\kappa}$ is solvable for $T$ **then**
7:  **if** $(T, \tilde{r})$ is convex **then**
8:  $(T, r) \leftarrow (T, \tilde{r})$
9:  $\delta \leftarrow \delta_{max}$
10:  **else if** $(T, \tilde{r})$ has exactly one concave edge $ij$ **then**
11:  Flip $ij$ in $T$
12:  **continue**
13:  **else if** $(T, \tilde{r})$ has more that one concave edge **then**
14:  $\delta \leftarrow \delta^2$
15:  **end if**
16:  **else**
17:  $\delta \leftarrow \delta^2$
18:  **end if**
19:  $\tilde{\kappa} \leftarrow (1 - \delta) \cdot \kappa(r)$
20: **end while**
21: **return** $(T, r)$

Algorithm 2 shows the algorithm as a direct translation of the ideas of the theory section. Line 6 of AlexandrovsPolyhedron requires solving the non linear system

$$\kappa(\tilde{r}) = \tilde{\kappa}.$$

This system is solved using a Newton solver. Here for every step solve the linearized system

$$J_\kappa(\tilde{r}) \cdot r = \tilde{\kappa}.$$

Then choose a step-width $\delta$ such that the residual decreases and continue until $\tilde{r}$ is sufficiently near at the real solution in terms of a small residual. This algorithm works for many examples, but is rather slow when many edge flips occur. For the creation of large models we use a modification of this algorithm.

**Algorithm 3** AlexandrovsPolyhedronFast

**Require:** $T'$, $\varepsilon$, $\delta_{max}$
1: $T \leftarrow$ DelaunayTriangulation($T'$)
2: $r \leftarrow (R, \ldots, R)$, such that $(T, r)$ is convex
3: $\delta \leftarrow \delta_{max}$
4: $\tilde{\kappa} \leftarrow (1 - \delta) \cdot \kappa(r)$
5: **while** $\| \kappa(r) \| > \varepsilon$ **do**
6:  **if** $\kappa(\tilde{r}) = \tilde{\kappa}$ is solvable for $T$ **then**
7:  $C \leftarrow$ set of concave edges
8:  **if** $C$ is empty **then**
9:  $(T, r) \leftarrow (T, \tilde{r})$
10:  $\delta \leftarrow \delta_{max}$
11:  **else**
12:  **for** $ij \in C$ **do**
13:  Flip $ij$
14:  **end for**
15:  **continue**
16:  **end if**
17:  **else**
18:  undo flips from the last iteration if there are any
19:  $\delta \leftarrow \delta^2$
20:  **end if**
21:  $\tilde{\kappa} \leftarrow (1 - \delta) \cdot \kappa(r)$
22: **end while**
23: **return** $(T, r)$

It appears that in most steps the generalized polytope remains convex even if we flip more that one edge to recover the local convexity. This leads to the idea to try flipping all concave edges and undo these flips if we did not succeed in creating a convex generalized polytope. Algorithm 3 illustrates this idea. In fact this approach has proved to be substantially faster than Algorithm 2. This algorithm can handle the symmetric cases mentioned above.

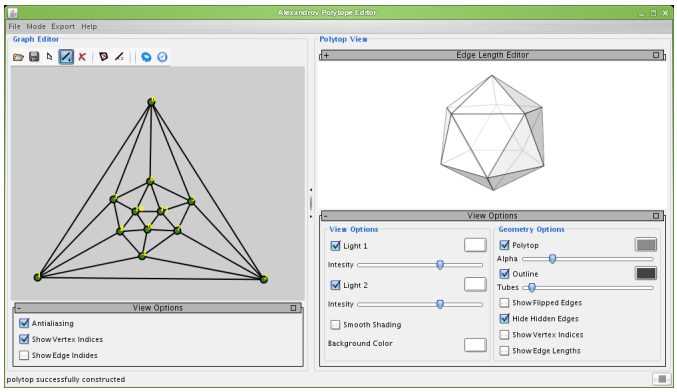



FIGURE 3.4. Screenshot of the Alexandrov polytope editor program

LISTING 3.1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE convexPolyhedralMetric>
<cpml description="default tetrahedron">
        <edgelist>
                <edge length="1.0">
                        <property name="hidden" type="boolean" value="true"/>
                </edge>
                <edge length="1.0"/>
                <edge length="1.0"/>
                <edge length="1.0"/>
                <edge length="1.0"/>
                <edge length="1.0"/>
        </edgelist>
        <trianglelist>
                <triangle a="0" b="1" c="4"/>
                <triangle a="2" b="5" c="1"/>
                <triangle a="3" b="2" c="0"/>
                <triangle a="5" b="3" c="4"/>
        </trianglelist>
</cpml>
```

## 3.3. The Program

The program "Alexandrov Polytope Editor" is laid out like the "Koebe Polyhedron Editor". The left section shows the graph designer where the combinatorics and edge lengths are entered. The right section is the 3D viewer for the resulting polytope (Figure 3.4).

There is a file format to specify a convex polyhedral metric. The CPML (Convex Polyhedral Metric Markup Language) file format is an XML format that defines such a polyhedral metric using triangles and edge lengths. Listing 3.1 shows an example of the file format.

**3.3.1. Testing the Algorithm.** The first thing to do after the algorithm is completely debugged is to establish a testing environment. We use the combinatorics of the icosahedron for our tests. Then we assign random lengths in the range of $\frac{1}{3}$ to 1 to the edges. If the metric is convex, then we start the calculation and display the resulting polyhedron. To initiate the tests press the button in the toolbar of the graph editor ( ).

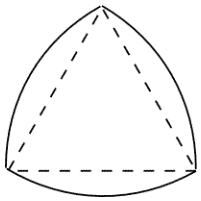


FIGURE 3.5. A Reuleaux Triangle

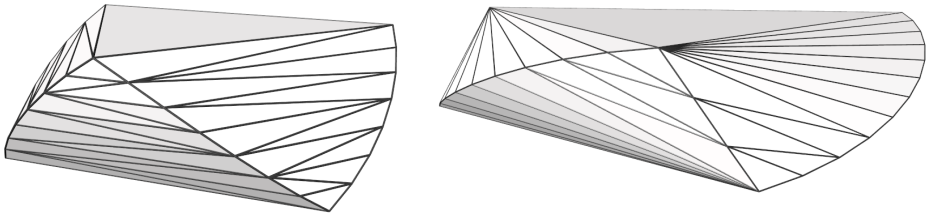


FIGURE 3.6. The Reuleaux-Triangle D-Form, here two Reuleaux triangles are glued together to form a D-Form

## 3.4. Examples

**3.4.1. D-Forms.** A D-Form is a convex three dimensional shape whose boundary consists of two developable surfaces bordered by closed curves of equal length. See [**Bou**] for a detailed description of this construction process and further examples. Since the individual parts of a D-Form are developable, the metric is flat and does not need to have vertices at other places that the boundary. Therefore we construct such a surface using a triangulation of the area inside the flat curves forming the boundary of each part. We use the Euclidean metric of the plane to assign edge lengths. The parts need to have the same number of edges on the boundary. For the sake of simplicity we use a discretization of the boundary which has constant edge length.

Figure 3.6 shows Reuleaux D-Forms. Here two Reuleaux triangles are identified along their boundary. A Reuleaux triangle is a shape of constant width, Figure 3.5. In the left part of Figure 3.6 the vertices of one Reuleaux triangle are glued to the midpoints of the sides of the other. In the right part of the figure a different position for the points on the edges is chosen.

The next example glues a circle to an equilateral triangle. Again the triangle and the circle have the same boundary length and the same number of edges at the boundary. Figure 3.7 shows the resulting polytope.

**3.4.2. The Reuleaux-Triangle-Tetrahedron.** The Reuleaux-Triangle-Tetrahedron is a generalization of the D-Form concept. Here we take four Reuleaux triangles and identify them along their sides. Three vertices of individual triangles meet at a point to give a curved tetrahedron. We created a model of this surface using a 3D printed (Figure 3.9).

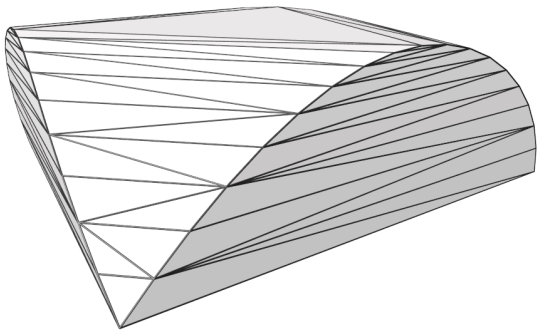


FIGURE 3.7. A D-Form created by glueing a circle to a triangle

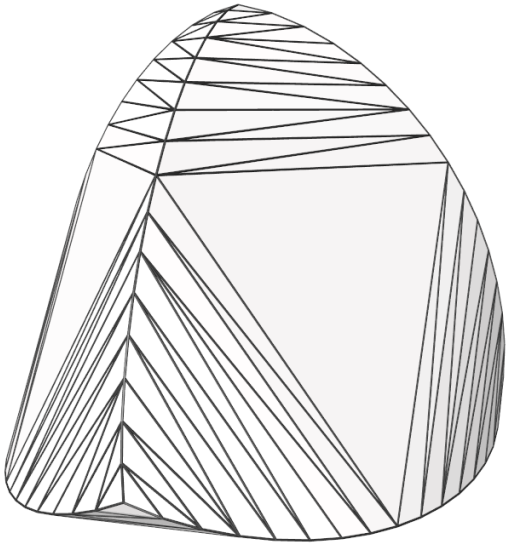


FIGURE 3.8. The Reuleaux-Triangle-Tetrahedron

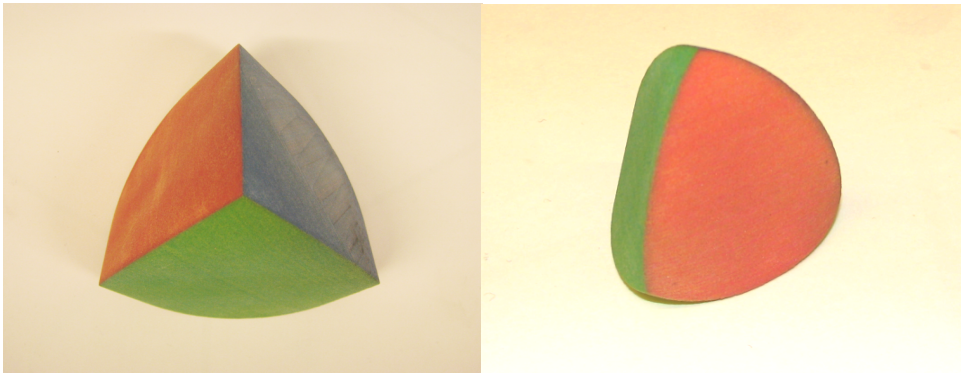


FIGURE 3.9. Models of the Reuleaux-Triangle-Tetrahedron printed by a 3D plaster printer

The program provides generators for all three examples. Change the mode of the application to “CPML Editor” by the selecting the “Mode” menu entry from the menu bar. Then select the combinatorial type you want to generate (,,).

CHAPTER 4

# The Software

## 4.1. The Half-Edge Data Structure API

The presented algorithms work on polyhedral data, and the usual way of representing such data is the indexed face set. This representation directly maps to current computer hardware thus provides fast rendering methods. On the other hand it doesn't encode combinatorial information directly. Such information is for example edge-vertex or face-vertex incidences. To handle such tasks efficiently we use a half-edge data structure. Figure 4.1 shows the Weiler graph of the data structure used in this implementation. An edge of the graph represents data saved in the start-node of the edge. Figure 4.2 shows the data stored in the nodes. The half-edge data structure API is contained in the package "halfedge" which is available on CD-ROM (Section 4.2). We believe this implementation can be helpful to other projects

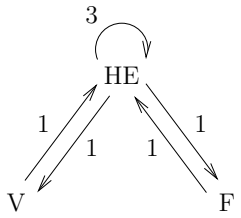
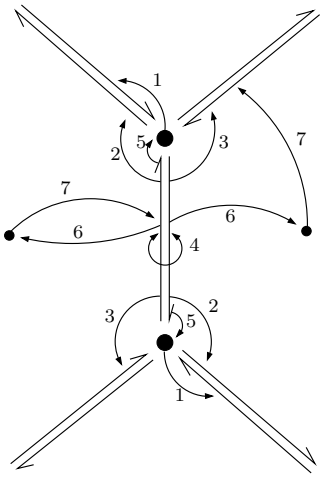



FIGURE 4.1. Weiler graph of the half-edge data structure




1 - Incident edge
2 - Next half-edge
3 - Previous half-edge
4 - Opposite half-edge
5 - The target vertex
6 - The left face of the half-edge
7 - One boundary edge of the face

FIGURE 4.2. The half-edge data structure

Listing 4.1

```
import halfedge.HalfEdgeDataStructure;
public class MyHEDS extends HalfEdgeDataStructure <
        MyVertex,
        MyEdge,
        MyFace
> {
        private static final long
                serialVersionUID = 1L;

        public MyHEDS() {
                super(MyVertex.class, MyEdge.class, MyFace.class);
        }
}
```

Listing 4.2

```
import halfedge.Vertex;

public class MyVertex extends Vertex<MyVertex, MyEdge, MyFace> {
        private static final long
                serialVersionUID = 1L;

        protected MyVertex getThis() {
                return this;
        }
}
```

Listing 4.3

```
import halfedge.Edge;

public class MyEdge extends Edge<MyVertex, MyEdge, MyFace> {
        private static final long
                serialVersionUID = 1L;

        protected MyEdge getThis() {
                return this;
        }
}
```

as well so we describe the main functionality and a few getting started steps. In the remainder of this section class names are printed in CAPITAL letters.

The HALFEDGEDATASTRUCTURE class is the container that stores all the nodes of the graph. It manages the creation and removal of nodes. All vertices, edges, and faces are stored in array-lists thus have constant time access by index. The container class is a generic class and is not intended to be used as is. The standard usage is to subclass the HALFEDGEDATASTRUCTURE class and use your own implementations of VERTEX, EDGE, and FACE. Listings 4.1-4.4 show an implementation example of the needed classes. An example of the usage is Listing 4.5. You can use the FACEBYFACEGENERATOR by Markus Schmies to construct a valid surface successively.

To access adjacency information, the node classes contain various methods which are implemented in the abstract base classes. The following list describes the vertex, edge, and face adjacency actions. The vertex adjacency methods are:

**getConnectedEdge():** a half-edge with the vertex as target
**getVertexStar():** all neighbor vertices

Listing 4.4

```
import halfedge.Face;

public class MyFace extends Face<MyVertex, MyEdge, MyFace> {
        private static final long
                serialVersionUID = 1L;

        protected MyFace getThis() {
                return this;
        }
}
```

Listing 4.5

```
import halfedge.generator.FaceByFaceGenerator;

public class Test {
        public static void main(String[] args) {
                MyHEDS heds = new MyHEDS();
                MyVertex v1 = heds.addNewVertex();
                MyVertex v2 = heds.addNewVertex();
                MyVertex v3 = heds.addNewVertex();
                MyVertex v4 = heds.addNewVertex();

                FaceByFaceGenerator<MyVertex, MyEdge, MyFace>
                        gen = new FaceByFaceGenerator
                                <MyVertex, MyEdge, MyFace>(heds);
                gen.addFace(v1, v2, v3, v4);

                System.out.println(heds);
        }
}
```

**getEdgeStar():** all adjacent half-edges with the vertex as target
**getFaceStar():** all adjacent faces

The half-edge adjacency methods are

**getTargetVertex():** the target vertex
**getStartVertex():** the start vertex
**getNextEdge():** the next half-edge
**getPreviousEdge():** the previous half-edge
**getOppositeEdge():** the opposite half-edge
**getLeftFace():** the face on the left of the half-edge
**getRightFace():** the face on the right of the half-edge

And the face adjacency methods are

**getBoundaryEdge():** a half-edge from the boundary of the face
**getBoundary():** the half-edges that form the boundary.

If an half-edge is in the boundary of some face, then the latter is considered to be "left of" this edge. Thus the method getLeftFace() always returns a face whose boundary contains the edge.

The class hierarchy's root is the abstract class Node (Figure 4.3). The abstract classes Vertex, Edge, and Face are subclasses of this base class. To provide type safety these classes are generic. The signature of the type Node includes parameters for vertex, edge, and face types (Listing 4.6), and the derived classes extend the Node class but don't substitute the generic

Listing 4.6

```
public abstract class Node
<
        V extends Vertex<V, E, F>,
        E extends Edge<V, E, F>,
        F extends Face<V, E, F>
>
```

Listing 4.7

```
public abstract class Vertex
<
        V extends Vertex<V, E, F>,
        E extends Edge<V, E, F>,
        F extends Face<V, E, F>
>
extends Node<V, E, F>
```

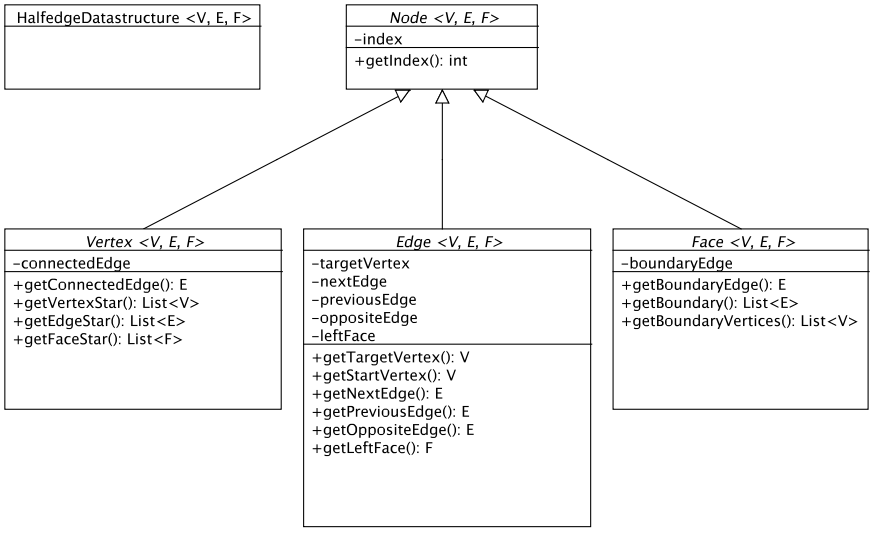



Figure 4.3. Half-edge Data Structure UML

parameters (Listing 4.7). The signatures of the classes Edge and Face are defined in the same way. To allow correct type inference the node classes are parametrized with the corresponding vertex-edge-face triple.

## 4.2. CD-ROM

The included CD-ROM contains all necessary libraries and programs to run the software described in Chapter 1, 2, and 3 with Windows™ or Linux. Figure 4.4 shows the directory structure of the CD-ROM. With Windows™ an autostart script runs the software directly after insertion of the CD-ROM. Use the bin/DiplomaThesis.bat script to start the program manually. With Linux you can use the start script bin/DiplomaThesis.sh to start the software. Figure 4.5 shows the start program, you can start the programs or view this document directly by pressing the corresponding button.

| | |
|---|---|
| / | Contains this document as PDF file |
| /bin | Start scripts |
| /data | Examples from the text |
| /lib | Java jar libraries |
| /src | Source files |
| /native | Native OpenGL binding libraries for Java |
| /jre | Java runtime for Windows™ |

FIGURE 4.4. The CD-ROM directories

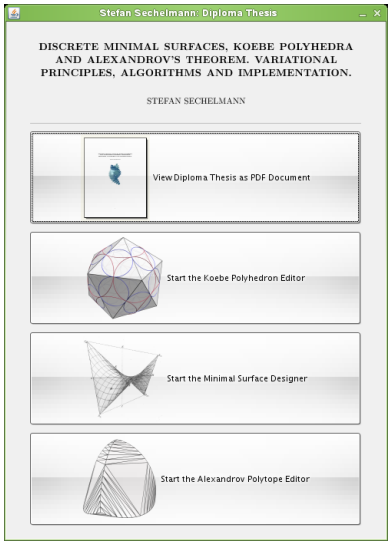



FIGURE 4.5. The starter program

APPENDIX A

# Zusammenfassung

In dieser Arbeit beschreiben wir die Theorie und Implementation von drei verschiedenen Computerprogrammen. Der erste Teil der Arbeit beschäftigt sich mit Koebe-Polyedern, das sind Polyeder, deren Kanten tangential zur Einheitssphäre sind. Die Berechnung solcher Polyeder basiert auf orthogonalen Kreismustern auf der Sphäre. Hierbei schneiden sich die Kreise orthogonal und bilden eine Überdeckung der ganzen Sphäre. Solche Kreismuster, sowie Variationsprizipien zur Berechnung, werden in [**BS04, Spr03**] beschieben. Ein orthogonales Kreismuster auf der Sphäre ist hierbei bis auf Rotation eindeutig durch die Radien der Kreise beschrieben. Einem orthogonalen Kreismuster ist nun ein eindeutiges Koebe-Polyeder mit gleicher Kombinatorik zugeordnet. Wir präsentieren Software zur Berechnung von Koebe-Polyedern, die auch auf der beigelegten CD enthalten ist.

Der zweite Teil der Arbeit beschreibt die Berechnung von diskreten Minimalflächen. Wir orientieren uns an den Ausführungen in [**BHS06**]. Eine diskrete Minimalfläche ist genau wie ihr glattes Vorbild dual zum Bild der Gauß-Abbildung. Die Dualitätstransformation ist analog zum glatten Fall die diskrete Christoffel-Transformation. Das diskrete Analogon zum Bild der Gauß-Abbildung sind kantentangentiale Polyeder also Koebe-Polyeder. Diese Definition bildet die Verbindung zwischen dem ersten und dem zweiten Teil. Aus diesem Grund benutzt die Software für diesen Teil die Algorithmen und Datenstrukturen des ersten Teils. Wir beschreiben die Funktionweise des Programms anhand von einigen Beispielen.

Im dritten Teil wenden wir uns Polyedern zu, die durch ihre Metrik auf dem Rand definiert sind. Der Satz von Alexandrov sagt aus, dass zu jeder konvexen polyedrischen Metrik ein eindeutiges konvexes Polyeder gehört, welches diese Metrik auf seinem Rand realisiert. [**BI06**] enthält einen konstruktiven Beweis dieses Satzes, mit dessen Hilfe ein Algorithmus zur Berechnung von Polyedern aus gegebener Metrik erstellt wurde. Wir präsentieren verschiedene Versionen dieses Algorithmuses und diskutieren Ergebnisse anhand von Beispielen. So berechnen wir einige D-Forms, das sind abwickelbare Flächen mit geschlossener Randkurve gleicher Länge, die am Rand identifiziert sind. Eine Veralgemeinerung hiervon ist das Reuleaux-Dreieck-Tetraeder, welches aus vier Reuleaux-Dreiecken zusammengeklebt ist.

Im letzten Teil beschreiben wir die Datenstruktur, die in allen Programmen verwendet wird, um polyedrische Daten zu speichern und zu verarbeiten. Hierbei handelt es sich um eine Half-Edge Datenstruktur. Diese hat die Eigenschaft die wichtigsten kombinatorischen Informationen in konstanter

Zeit liefern zu können. Wir demonstrieren anhand von Beispielimplementationen die Verwendung dieser Datenstruktur.

APPENDIX B

# Acknowledgment

I'd like to thank my advisor Prof Alexander I. Bobenko for convincing me to start working for the geometry group at Technische Universität Berlin and for always being a source of new ideas for my projects. Furthermore I would like to thank Boris Springborn for spending hours of explanation and debugging with me. His help was crucial to the Koebe and the minimal surfaces project. I'd like to say thanks to Ulrike Bücking who helped me with the curvature line patterns of the example surfaces. The Alexandrov part of this work has been created with the assistance of Ivan Izmestiev and I'd like to thank him for being a good advisor. At last I'd like to thank Steffen Weißmann for introducing me to the fabulous jReality Java library [**TBb**] that he's been developing. Most of the pictures in this work have been created with the help of this library. And last but not least I thank my beloved Franzi for checking my work for spelling errors and punctuation.

# Bibliography


[BHS06] Alexander I. Bobenko, Tim Hoffmann, and Boris A. Springborn, *Minimal surfaces from circle patterns: Geometry from combinatorics*, Annals of Mathematics **164** (2006), 231–264.

[BI06] Alexander I. Bobenko and Ivan Izmestiev, *Alexandrov's theorem, weighted delaunay triangulations, and mixed volumes*, `http://arxiv.org/abs/math.DG/0609447`, 2006.

[Bou] Paul Bourke, *D-forms*, `http://local.wasp.uwa.edu.au/~pbourke/surfaces_curves/dform/`.

[BS04] Alexander I. Bobenko and Boris A. Springborn, *Variational principles for circle patterns and koebe's theorem*, Trans. Amer. Math. Soc. **356** (2004), 659–689.

[BV04] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Camebridge University Press, 2004.

[Ede01] Herbert Edelsbrunner, *Geometry and topology for mesh generation*, Cambridge Monographs on Applied and Computational Mathematics, vol. 7, Cambridge University Press, Cambridge, 2001. MR MR1833977 (2002k:65206)

[Hei] Bjørn-Ove Heimsund, *Matrix toolkits for java (mtj)*, `http://rs.cipr.uib.no/mtj`.

[Koe36] P. Koebe, *Kontaktprobleme der konformen abbildung*, Sächs. Akad. Wiss. Leipzig Math.-Natur. **88** (1936), 141–146.

[Sch90] Hermann Amandus Schwarz, *Gesammelte mathematische abhandlungen*, Springer, Berlin, 1890.

[Spr03] Boris A. Springborn, *Variational principles for circle patterns.*, Ph.D. thesis, Technische Universität Berlin, 2003.

[Spr05] ———, *A unique representation of polyhedral types, centering via möbius transformations*, `http://arxiv.org/abs/math.MG/0401005`, December 2005.

[TBa] TU-Berlin, *Java tools for experimental mathematics*, `http://www.jtem.de`.

[TBb] ———, *jreality: a java 3d viewer for mathematics*, `http://www.jreality.de`.

[Zie04] Günter M. Ziegler, *Convex polytopes: Extremal constructions and f-vector shapes*, Park City Mathematical Institute (PCMI 2004) Lecture Notes. With an Appendix by Th. Schröder and N. Witte; Preprint, TU Berlin, 73 pages; `http://www.arXiv.org/math.MG/0411400`, November 2004.