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 Σ.

If for any $x \in \Sigma$ the angle at x is less that 2π, 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.

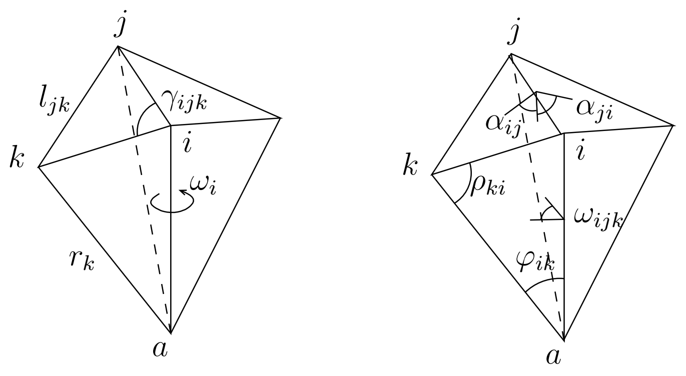

FIGURE 3.1. Angles and lengths of a generalized polytope

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 Σ 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

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

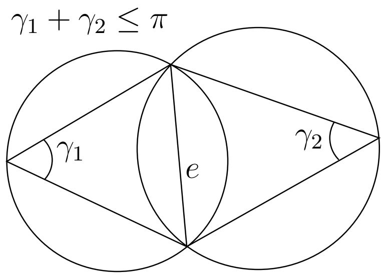

FIGURE 3.2. Local Delaunay condition

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 ω_i. It is

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

and ω_i is the sum of all angles at edge ia. The formula for ρ 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 γ_1 and γ_2 are angles in the adjacent triangles of the edge (Figure 3.2).

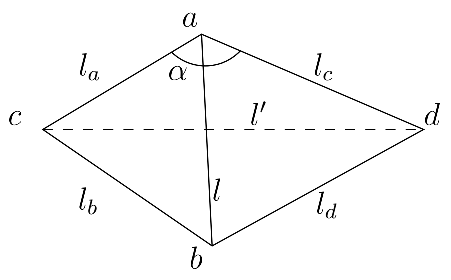

FIGURE 3.3. Flipping ab to cd

Algorithm 1 DELAUNAYTRIANGULATION

```
Require: stack $S$ with all marked edges
 1: while $S$ is non-empty do
 2:   pop $ab$ from $S$ and unmark it
 3:   if $ab$ not locally Delaunay then
 4:     FLIP $ab$ to$cd$
 5:     for $xy \in ac, cb, bd, da$ do
 6:       if $xy$ is not marked then
 7:         mark $xy$ and push it on $S$
 8:       end if
 9:     end for
10:   end if
11: 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

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

if the following condition holds

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

where δ_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 δ 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 κ 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: $r \leftarrow (R, \dots, R)$, 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 δ 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.