Cauchy's theorem for orthogonal polyhedra of genus 0

Therese Biedl[*] Burkay Genc[†]

Technical report CS-2008-26

Abstract

A famous theorem by Cauchy states that a convex polyhedron is determined by its incidence structure and face-polygons alone. In this paper, we prove the same for orthogonal polyhedra of genus 0 as long as no face has a hole. Our proof yields a linear-time algorithm to find the dihedral angles.

1 Introduction

A famous theorem by Cauchy states that for a convex polyhedron, the incidence structure and the face-polygons determine the polyhedron uniquely. Put differently, if we are given a graph with a fixed order of edges around each vertex, and we are given the angles at every vertex-face incidence and edge lengths, then there can be at most one set of dihedral angles such that graph, facial angles, edge lengths and dihedral angles are those of a convex polyhedron. See for example the book by Aigner and Ziegler [1] for a proof.

Cauchy's theorem fails for polyhedra that are not convex. An easy example is a polyhedron where one face has a rectangular "hole" where a small box can be popped to the "inside" or "outside". But in fact, there are even so-called *flexible* polyhedra where the dihedral angles change continuously, though it has been shown that their volume must stay the same [5].

We show in this paper that Cauchy's theorem *does* hold for orthogonal polyhedra of genus 0, as long as we exclude these holes in faces. (Rather than defining holes, we will express this by saying that the graph of the polyhedron must be connected; see Section 2 for precise definitions.) Thus, while a big cube with a small cube attached on one face has two possible realizations, this is in essence the only way in which multiple realizations are possible.

Cauchy's theorem for convex polyhedra is proved by contradiction and does not lead to an algorithm; only very recently have exponential algorithms been found for this problem (see [11, 8, 4] and also the review paper by O'Rourke [10]), and no polynomial-time algorithm is known. In contrast to this, our proof of Cauchy's theorem for orthogonal polyhedra of genus 0 is algorithmic, and yields a simple linear-time algorithm to find the dihedral angles.

1.1 Roadmap

We first briefly outline the approach of this paper. Rather than proving uniqueness and then deriving an algorithm from the proof, we provide an algorithm that reconstructs an orthogonal polyhedron. There will never be any choice in the assignment of dihedral angles, except at one moment when we can choose one dihedral angle. Hence we obtain two sets of dihedral angles, and can argue that only one of them could possibly do; this then proves uniqueness.

Our algorithm proceeds in three steps. In the first step in Section 3, we only identify which dihedral angles must be flat, i.e., have value 180°. We do this by determining for each face whether it is perpendicular to the x-axis, y-axis or z-axis; the algorithm to do so is simple, but proving its correctness is not.[1] Two adjacent faces that are perpendicular to the same axis

[*]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, biedl@uwaterloo.ca

[†]Izmir University of Economics, Faculty of Computer Science, Sakarya Cad. No:156, Balcova, Izmir, Turkey, burkaygenc@gmail.com. This work was done while the author was a student at University of Waterloo and supported by NSERC.

[1]A preliminary version of this algorithm appeared in 2004 [2], but its correctness was shown only for orthogonally convex polyhedra for which all faces are rectangles.

necessarily must have a flat dihedral angle between them, so this determines all flat dihedral angles.

The problem hence reduces to reconstructing an orthogonal polyhedron where all dihedral angles are non-flat. In Section 3, we show that there are only 7 possible configurations of vertices for such a polyhedron. Moreover, if we fix one dihedral angle and know all the facial angles, this determines all other dihedral angles at a vertex, and hence with a simple propagation scheme, all dihedral angles can be computed as long as the graph is connected.

Finally, we study in Section 5 which of the two sets of dihedral angles obtained with the above can possibly be the correct set of dihedral angles. This is the only part of the algorithm that makes use of edge lengths. We conclude with remarks in Section 6, where we also study an "inverse" problem of reconstructing facial angles, given dihedral angles.

2 Definitions

A *polygonal curve* is a simple closed curve in the plane that consists of a finite number of line segments. A *polygon* is a set in a plane whose boundary is a polygonal curve. A *polygonal set* is an interior connected set in a plane that is the finite union of polygons. A *polyhedral surface* is a 2-manifold that is a finite union of polygonal sets. A *polyhedron* is a set in 3D whose boundary is a polyhedral surface. The polyhedron has *genus* g if its boundary is a surface of genus g.

A *face* of a polyhedron is a maximal polygonal set on the boundary of the polyhedron. A *vertex* is a point that belongs to at least three faces. An *edge* is a maximal line segment that belongs to two faces and contains no vertex other than its endpoints. A *facial angle* is the interior angle of a face at a vertex. A *dihedral angle* is the interior angle at an edge between two adjacent faces.

The incidences between vertices and edges of a polyhedron determine a graph called the *graph of the polyhedron*. Looking at the polyhedron from the outside fixes a cyclic order of edges around each vertex; this is called the *induced embedding of the graph*.

For a polyhedral surface, we can also define a graph by using as faces the polygonal sets that defined the polyhedral surface, and then carry over all other definitions (vertex, edge, graph, facial angles, dihedral angles). The main difference is that in a polyhedral surface, some dihedral angles may be flat, i.e., have value $180°$, while this is not possible in a polyhedron.

We will usually assume that we are given an *embedded graph*, i.e. a graph with a fixed cyclic order of edges around each vertex. From this order we can determine the *faces of the graph*, which are the cycles obtained by always taking the next edge in cyclic order. We will also assume that we are given *facial angles of the graph*, which are values at each incidence between a vertex and a face of the graph.

Given an embedded graph and facial angles (and sometimes also the lengths of the edges), we say that a polyhedral surface S *realizes* the input if its graph (with the induced embedding) is the given embedded graph, and its facial angles (and edge lengths, if given) are as prescribed in the input.

We will almost only study orthogonal polyhedra of genus 0 in this paper. A polyhedral surface/polygon is *orthogonal* if all its faces are perpendicular to a coordinate axis. This implies that all facial angles and all dihedral angles are multiples of $90°$. The polyhedral surface has genus 0 if and only if its graph is *planar*, i.e. it can be drawn in the plane without crossing.

3 Flat dihedral angles

In this section, we present an algorithm that, given an embedded planar graph and facial angles, determines which of the edges of the graph must have a flat dihedral angle in any realizing orthogonal polyhedral surface.

3.1 Algorithm

For each face of the input graph, the facial angles determine relative orientations of edges within the face. We write $e \parallel e'$ if e and e' are edges on one face and have the same orientation within that face. We can extend $\parallel$ into an equivalence relationship $\sim$ by defining that $e \sim e'$ if there exists a set of edges $e = e_1, \ldots, e_k = e'$ with $e_i \parallel e_{i+1}$ for $1 \leq i < k$. We define a *band* to be an equivalence class under this equivalence relationship $\sim$. Note that the bands can easily be computed in linear time from the embedded graph and the facial angles. Directly from the definition of || and $\sim$, the following is obvious:

Observation 1 *All edges in a band must be parallel in any realizing orthogonal polyhedral surface.*

In any orthogonal polyhedral surface all edges are parallel to one coordinate axis. In such a realization, it hence makes sense to say that band B has *orientation* x if all its edges are parallel to the x-axis, and similarly for the other two orientations.

Two bands $B_1 \neq B_2$ *cross* if there exists a face in the graph that contains edges from both. Since B_1 and B_2 were equivalence classes, this means that their edges are not parallel to each other. In particular, if B_1 had orientation α in some realization, $\alpha \in \{x, y, z\}$, then B_2 cannot have orientation α.

We are now ready to describe the algorithm. During the algorithm, we store a set $T(B_i)$ of possible orientations of band B_i. We fix orientations arbitrarily for two bands that cross, and then propagate along bands that cross them to eliminate orientations that cannot be correct for these bands. It is not clear why this should identify all orientations, but we will that it does. The precise algorithm is as follows:

```
Algorithm BANDORIENTATION
for all bands $B$
    set $T(B) = \{x, y, z\}$
    mark $B$ as "unidentified".
Let $B_1$ and $B_2$ be two bands that cross
    and set $T(B_1) = \{x\}$ and $T(B_2) = \{y\}$.
while there are unidentified bands
    find an unidentified band $B$ with $|T(B)| = 1$.
    if there is none, output an error message.
    else mark $B$ as identified
        for each band $B'$ that crosses $B$
            set $T(B') \leftarrow T(B') - T(B)$.
            if $T(B')$ is now empty
                output an error message.
```

This algorithm can be implemented in linear time if we pre-compute an auxiliary graph H of bands, which has a vertex for every band and an edge between two bands if and only if the bands cross. By storing bands in buckets by the size of $|T(B)|$, we can then in $O(1)$ time find the next band B to be identified, and in $O(\deg_H(B))$ time update all the bands that B crosses; this is $O(m+n)$ time overall since H has size $O(m+n)$.

We will show the following result in the next subsection:

Lemma 1 *Assume that an embedded planar graph with facial angles can be realized by an orthogonal polyhedral surface S. Then Algorithm* BANDORIENTATION *terminates without error message. Moreover, after applying a suitable rotation of S, for all bands B the unique value left in $T(B)$ at termination of the algorithm is the orientation of B in S.*

The band orientations then tell us all edge orientations (i.e., axis to which each edge is parallel), which in turn determines to which axis each face is perpendicular. We cannot determine dihedral angles directly from this (because there are still two possible directions for each face normal), but we can determine flat dihedral angles, since for any two adjacent co-planar faces, the dihedral angles at the edges shared by them must be 180°.

Theorem 1 *Given an embedded planar graph with facial angles, we can in $O(m+n)$ time*

- *report that no orthogonal polyhedral surface can realize this graph and facial angles, OR*
- *report all edges of the graph for which the dihedral angle must be 180° in any orthogonal polyhedral surface that realizes this graph and facial angles.*

3.2 Correctness

In this section, we prove correctness of Algorithm BANDORIENTATION, i.e., we prove Lemma 1. We assume throughout this section that some orthogonal polyhedral surface S exists that realizes the given graph and facial angles. We furthermore assume that S has been rotated such that all edges in band B_1 (the first band picked by Algorithm BANDORIENTATION) are parallel to the x-axis, and all edges in band B_2 are parallel to the y-axis. The following observation is then quite straightforward:

Lemma 2 *At any time during algorithm* BANDORIENTATION, *for any band B the orientation of B is in $T(B)$.*

Proof: This holds initially by the rotation of S and since $T(B) = \{x, y, z\}$ for all bands $B \neq B_1, B_2$. We only change a set if we update $T(B') \leftarrow T(B') - T(B)$, for some band B with $|T(B)| = 1$ and some band B' that crosses B. By induction $T(B)$ contains the orientation of B, and since $|T(B)| = 1$, the unique value in $T(B)$ is the orientation of B. Since B' crosses B, its edges are not parallel to the edges in B, so removing $T(B)$ from $T(B')$ does not remove the orientation of B'. $\square$

So, if Algorithm BANDORIENTATION ends without error message, then all band orientations have indeed been determined as desired. It cannot ever terminate when $T(B)$ becomes empty for some band B, because by the above the orientation of B remains in $T(B)$ if S exists.

The remaining possibility for Algorithm BANDORIENTATION to terminate is when there are no unidentified bands with $|T(B)| = 1$ left. We claim that this cannot happen if the realizing polyhedral surface S has genus 0.

It will be easier to assume that S is quadrangulated, i.e., every face is a rectangle. This is not a restriction: If S is not quadrangulated, we can add vertices and edges to obtain a quadrangulated orthogonal polyhedral surface S'. Any band of S corresponds to the union of bands of S', so if Algorithm BANDORIENTATION identifies all bands of S', it also identifies all bands of S.

Since S is quadrangulated, bands (which were defined as a set of edges) actually can be interpreted naturally as a sequence of faces instead, see Figure 1. Let e_0 be an edge in a band B, and let f_0 be an incident face of e_0. Since f_0 is a rectangle, there exists only one other edge e_1 on f_0 that is parallel to e_0. let f_1 be the other face incident to e_1. Continue in this manner until we return to edge e_0.

Any two consecutive edges of the band have the same edge length and span the same range of coordinates since they are on a face that is a rectangle; therefore all edges of a band have the same edge length and span the same range of coordinates. (A set of faces which are connected and bounded by two planes has been called a band elsewhere [6], hence our name for the set of edges.)

Assume Algorithm BANDORIENTATION stops with some bands not identified. Then there are three types of faces: those where 0, 1 or 2 of the two bands containing the face have been identified. No two adjacent faces can be of type 0 and 2, because otherwise the band that contains them both is both identified and unidentified. Since the surface is connected, there must exist faces of type 1, i.e., there is a face for which one band B^* containing the face is identified, and the other band (which crosses B^*) is not identified. We will show that this is impossible.

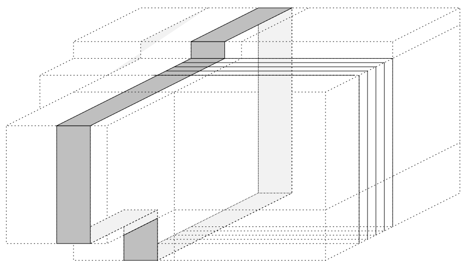
Figure 1: A band of a quadrangulated orthogonal polyhedral surface (gray) and one of its chords (lined).

Lemma 3 *Let B^* be a band that has been identified. Then all bands crossing B^* also will be identified by Algorithm* BANDORIENTATION.

Proof: We first give an outline of the proof, which is by contradiction. Assume there exists a maximal sequence of faces of B^* where the crossing bands have not been identified. Using genus 0, we can argue that some band must cross B^* once within this sequence and once not within this sequence. Then we argue that this band must "interleave" with another band that has been identified. We also argue that interleaving bands must cross, which means that there are three bands (one of them is B^*) that pairwise cross and two have been identified. This means that Algorithm BANDORIENTATION will also identify the third, a contradiction.

The precise proof will not work with bands directly, but instead will only consider subsequences of bands that stay "on one side" of B^*. Define the *cycle $c(B^*)$ of band B^** to be the cycle on surface S obtained by connecting the midpoints of consecutive edges in B^*. Because S has genus 0, $c(B^*)$ splits S into two pieces; arbitrarily pick one of them and call it the *interior* of $c(B^*)$.

A *chord* is a set $f_1, \ldots, f_k$ of faces such that $f_1, \ldots, f_k$ are consecutive faces of a band B that crosses B^*, f_1 and f_k are on B^* and $f_2, \ldots, f_{k-1}$ are not on B^* and in the interior of $c(B^*)$. See also Figure 1. Every band that crosses B^* defines at least one chord. We will use all terminology defined for a band B (such as "orientation" and "identified") also for chords defined by B.

The first and last face of a chord is called a *chord-pair*. From the definition, a chord-pair of a chord defined by band B is a pair of faces that belong to both B^* and B. Since all faces of a band span the same range in one of the coordinate axis, the two faces of a chord-pair span the same range in two of the coordinate axes.

We now consider a restricted version of Algorithm BANDORIENTATION, where we only propagate orientations along chords whose chord-pairs alternate along B^*. More precisely, two chord-pairs $\{u_k, v_k\}$, $k = 1, 2$ are said to *interleave* if their order along B^* is u_1, u_2, v_1, v_2 or u_1, v_2, v_1, u_2. We say that two chords interleave if their chord-pairs interleave.

Claim 1 *Chords that interleave cross each other in a face in the interior of $c(B^*)$.*

Proof: Let $c(B^*)$ be the cycle of band B^*. Let C_1 and C_2 be two chords that interleave, and for $k = 1, 2$ let c_k be the path obtained by connecting the midpoints of the faces of C_k in order. By definition of chord, c_k is a curve on the surface of S that connects two points on $c(B^*)$ and is otherwise in the interior of $c(B^*)$. Since C_1 and C_2 interleave, the curves $c(B^*), c_1$ and c_2 form K_4, the complete graph on four vertices. If c_1 and c_2 do not cross, then we have a planar drawing of K_4 (on S, which has genus 0) such that all vertices are drawn on one face (namely, $c(B^*)$). Since K_4 is not an outer-planar graph, this is not possible. See Figure 2. So c_1 and c_2 do cross, and the face that contains that crossing point is common to C_1 and C_2. □

There exists at least one other band that crosses B^* and whose orientation has been identified. (This holds if B^* is one of the initial two bands, because they cross each other, and also holds if B^* was identified later, because then $|T(B^*)|$ became 1 due to some crossing identified band.) Fix an arbitrary chord C^0 of this identified band.

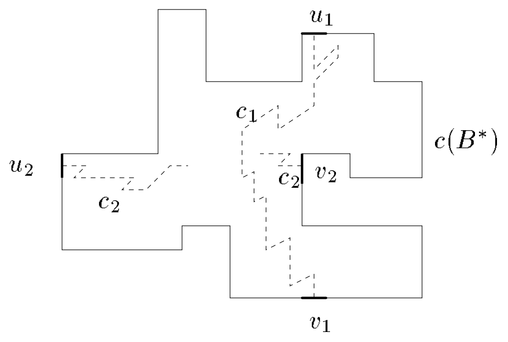

Figure 2: Chord-pair $\{u_1, v_1\}$ interleaves chord-pair $\{u_2, v_2\}$.

Starting from C^0, we now mark all chords that can be reached via interleaving chords. More precisely, mark all chords C^1 that interleave C^0, then in turn mark all chords that interleave C^1, and so on until no more chords can be marked.

Claim 2 *All marked chords are identified by Algorithm* BANDORIENTATION.

Proof: This holds for C^0 by choice of C^0 as a chord defined by an identified band. Assume chord C_j was marked because it interleaves some previously marked chord C_i. By induction C_i has been identified. By Claim 1, C_i and C_j cross. So there are three bands (B^* and the two bands for C^i and C^j) that mutually cross, and two of them are identified. This will also identify the third, because the orientations of the first two (which are different since they cross) are both removed from the list of possible orientations for the third by Algorithm BANDORIENTATION. □

We use the term "marked" also for a chord-pair (which is marked if and only if the chord of it is marked), and even for faces of B^*. Note that every face f of B^* belongs to a chord, since there is a band crossing B^* at this face, and the part of the band that enters the interior of $c(B^*)$ forms a chord. On the other hand, every face of B^* belongs to only one chord, since no three bands meet in a face. So we call a face of B^* marked if and only if the unique chord that contains it is marked.

If all faces of B^* are marked, then by Claim 2 all chords, and hence all bands that cross B^* are identified and we are done. So assume not all faces of B^*

are marked, and let U be a maximal contiguous set of faces of B^* that is not marked.

Claim 3 *There is a chord-pair $\{f, f'\}$ with $f \in U$ and $f' \notin U$.*

Proof: Note that cycle $c(B^*)$ lies within one plane; we will now consider $c(B^*)$ as a 2D polygonal curve within that plane. Let u be the intersection of U with $c(B^*)$. Then $c(B^*) - u$ forms an open curve in a plane. By considering the region between the two endpoints of that open curve, we can find a horizontal or vertical line ℓ that intersects $c(B^*) - u$ an odd number of times. See Figure 3(a).

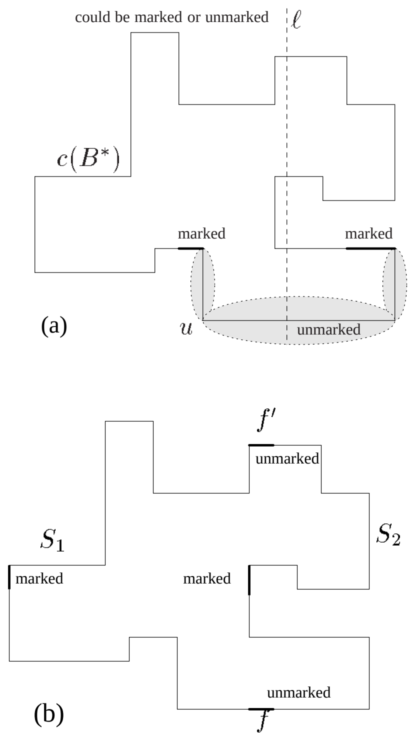

Figure 3: (a) There must be a horizontal or vertical line ℓ that intersects $c(B^*) - u$ an odd number of times. (b) Along this line, we can find an unmarked chord-pair that interleaves a marked chord-pair.

Consider all edges of $c(B^*)$ that are intersected by ℓ; this corresponds to a set L of faces of B^* that are intersected by ℓ (when considering ℓ as a line in 3D.) Any chord-pair that contains one face in L must also contain another face in L, because as observed earlier, the two faces of a chord-pair have the same range along two coordinate axes. By definition of ℓ, $L \cap (B^* - U)$ contains an odd number of faces, so by parity one of the chord-pairs that have faces in L can have only one of its faces in $B^* - U$, and the other face must be in U. □

By definition of U, $f \in U$ means that f was not marked. The chord-pair $\{f, f'\}$ splits B^* into two subsequences S_1, S_2 of faces, and both of them must contain at least one marked face, since $f' \notin U$ and U was chosen as a maximal sequence of unmarked faces. See Figure 3(b). During the iterative process of marking chords, there must have been a first time when we had marked faces on both S_1 and S_2. The chord-pair that caused this to happen hence had one face in S_1 and the other in S_2. But then this chord-pair interleaves with $\{f, f'\}$, which means $\{f, f'\}$ should have been marked as well, a contradiction.

This finishes the proof of Lemma 3, and hence the proof of Lemma 1. □

4 Non-flat dihedral angles

If we know all flat dihedral angles, we can delete the corresponding edges in the graph, and then delete the resulting isolated vertices and contract the resulting vertices of degree 2 into their neighbours. Doing this merges all faces of a polyhedral surface S until they become faces of the polyhedron bounded by S, and the resulting graph is the graph of the polyhedron. In this section, we are interested in determining the remaining dihedral angles, and we can thus assume that we are given the graph of the polyhedron.

Let v be a vertex of an orthogonal polyhedron. The incident 8 octants of v may or may not be occupied by the polyhedron within a small neighbourhood of v, yielding 2^8 possible configurations at vertex v. Of those, many cannot occur in an orthogonal polyhedron, since the resulting surface is not a 2-manifold. Some more have a flat dihedral angle. Eliminating all these cases and omitting rotational symmetries, we are left with only 7 vertex configurations, which are given in Figure 4.

In particular, each vertex of an orthogonal polyhedron can have three, four or six incident edges (so it has degree 3, 4 or 6 in the graph.) In Figure 5, we give the vertex configurations together with the facial angles and dihedral angles in the graph. The reader

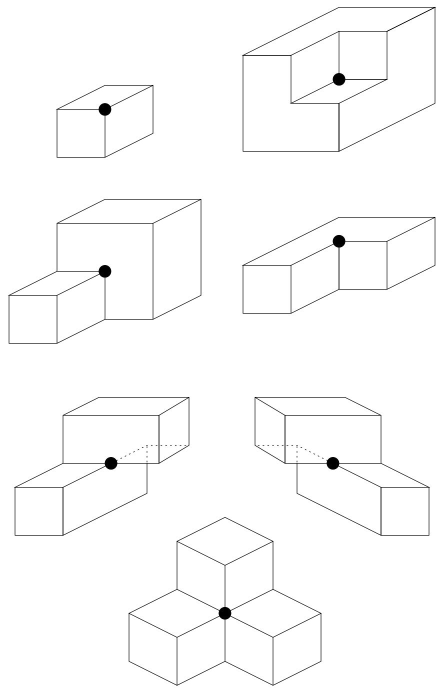
Figure 4: All vertex configurations of an orthogonal polyhedron.

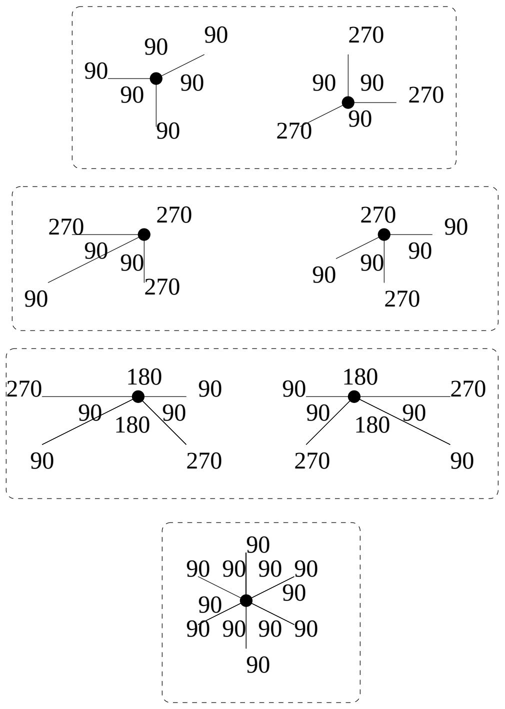

Figure 5: The vertex configurations with facial and dihedral angles.

should at this point start to forget the geometry and view this as an embedded graph with facial angles and labels on all edges.

We group the 7 configurations into four groups; configurations in different groups have different degrees or different facial angles. Within each group, any mapping from one configuration to the other that preserves order and facial angles maps every dihedral angle α to its opposite $360° - \alpha$. Since $\alpha \neq 180°$, this implies the following:

Observation 2 *All dihedral angles at a vertex v are determined by the degree of v, the facial angles at v, and one dihedral angle of an edge incident to v.*

all dihedral angles can hence be computed if one initial dihedral angle is fixed, as follows:

```
Algorithm DihedralAngles
let $e^*$ be an arbitrary edge of the graph $G$
set $d_1(e^*) = 90°$, $d_2(e^*) = 270°$
compute a traversal of $G$, starting at one endpoint of $e^*$
for all vertices $v$, in order of traversal
    let $e_v$ be an incident edge of $v$ for which $d_1(e_v)$
        and $d_2(e_v)$ have been determined.
        ($e_v = e^*$ for the first vertex, and the edge
        along which $v$ was reached otherwise.)
    for all edges $e \neq e_v$ incident to $v$
        set $d_1(e)$ to be the unique dihedral angle
            determined by the degree of $v$, the facial
            angles of $v$, and $d_1(e_v)$.
        if none of the configurations matches, or if
            this changes a previously assigned value
            of $d_1(e)$, output an error message.
        Similarly set $d_2(e)$ using $d_2(e_v)$.
```

The computation of $d_1(.)$ and $d_2(.)$ can be integrated into the traversal, and the running time is hence $O(m + n)$ time.

5 Selecting among two sets

At this point, we have computed two possible sets of dihedral angles, and we now need to determine which of them is the correct one.

These two sets are in fact opposite to each other, i.e., $d_1(e) = 360° - d_2(e)$ for all edges e. This clearly holds for the initial edge, and by induction also for the other edges, since the two configurations within a group in Figure 5 have opposite dihedral angles. So if the set $\{d_1(.)\}$ is realized by an orthogonal polyhedron P, then $d_2(e)$ is the outside angle between the faces adjacent to e in P; we could thus call $\{d_2(.)\}$ the *outside dihedral angles.*

To determine which of the two sets are the inside and which the outside dihedral angles, we use edge lengths and reconstruct the coordinates of all vertices. To be precise, pick some vertex of degree 3, assign it to be located at the origin, and arbitrarily assign its three incident edge to be directed along the x-axis, y-axis and z-axis. Using the facial angles, edge lengths, and the dihedral angles from $\{d_1(e)\}$, we can then easily compute all coordinates of all vertices in $O(m+n)$ time. (If this assigns two different coordinates to the same vertex, output an error message; the edge lengths cannot have been correct.)

Now find a vertex v with maximal x-coordinate (breaking ties arbitrarily), and let f be a face adjacent to v and perpendicular to the x-axis. Since there are no flat dihedral angles, the edges incident to f must have dihedral angle $90°$, otherwise there would be a vertex with even larger x-coordinate. This decides which of $\{d_1(.)\}$ and $\{d_2(.)\}$ was correct, and only one of them can be correct.

Putting all three algorithms together, we hence obtain the following:

Theorem 2 *Given an embedded planar graph with facial angles and edge lengths, we can in $O(m+n)$ time*

- *find the dihedral angles of any orthogonal polyhedral surface that has this graph, facial angles and edge lengths, OR*
- *report that this graph and facial angles can only belong to an orthogonal polyhedral surface for which the polyhedron bounded by it has a disconnected graph, OR*
- *report that no orthogonal polyhedral surface can realize this graph, facial angles and edge lengths.*

Moreover, if a realizing orthogonal polyhedral surface exists, then it is unique.

6 Remarks

We assumed that we are given a graph, facial angles and edge lengths, and that the reconstructed orthogonal polyhedron has a connected graph and genus 0. We now briefly discuss these assumptions.

- Inspection of the proof of Cauchy's theorem shows that it does not use edge lengths, so for a convex polyhedron the graph and facial angles determine the dihedral angles. Our proof also does not use edge lengths, except at the very last step where we determine which of two possible sets of dihedral angles is the correct one.

  It seems exceedingly likely that this step could be done without using edge lengths. In particular, in the corresponding 2D problem (given a set of angles, can this be the set of angles of an orthogonal polygon?) there is a simple solution: the set of n angles can be realized if and only if it adds up to $180°(n+2)$. If any band happens to have only rectangular faces, then the cycle of the band (as defined in Section 3) lies within a plane, and studying the dihedral angles of this band will tell us which set is correct. But for arbitrary bands the incidence structure of the faces on it is more complicated. Can we use it somehow to determine the correct set of dihedral angles without using edge lengths?
- We demanded that the graph of the polyhedron is connected, i.e., no face has holes. If this condition is dropped, then testing whether a realizing polyhedral surface exists becomes NP-hard even for genus-1 orthogonal polyhedra [2], and the proof can easily be modified to genus-0 orthogonal polyhedra [9]. We show in the appendix that the problem is in fact NP-hard in the strong sense, and in particular also holds for polyhedral surface where every face is a unit rectangle.
- We demanded that the orthogonal polyhedral surface has genus 0. This was used frequently throughout the proof of correctness of Algorithm BANDORIENTATION. Already for genus 1, Algorithm BANDORIENTATION may not identify all bands (depending on how the initial bands are chosen); see Figure 6 for an example.

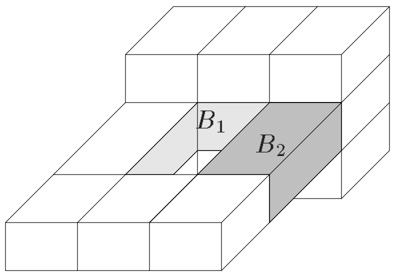

Figure 6: An orthogonal polyhedral surface of genus 1 where Algorithm BANDORIENTATION cannot identify any bands after the initial two.

The other algorithms work without modification for surfaces of higher genus, so Cauchy's theorem holds for higher genus orthogonal polyhedra (not polyhedral surfaces, i.e., no flat dihedral angles are allowed), as long as they have a connected graph.

- Our algorithm computes the set of dihedral angles, and as a by-product also vertex coordinates, but it does not check whether the resulting surface is indeed a 2-manifold. This can be done in polynomial time (see [3]).

6.1 An inverse problem

One could also ask an inverse question to Cauchy's theorem: Can we reconstruct facial angles, given dihedral angles (as well as embedded graph and edge lengths)? To our knowledge, this problem has not been studied before, and in particular, it is open whether the set of facial angles is unique for a convex polyhedron. We study here orthogonal polyhedra.

Assume first that none of the (given) dihedral angles is flat. We show that in this case we can reconstruct facial angles in $O(n)$ time, with an approach quite similar to the one in Section 4. Again the crucial idea is that there are a constant number of configurations at a vertex of an orthogonal polyhedron, see Figure 5. If a vertex has degree 3 or degree 6, then the dihedral angles determine all facial angles. To resolve degree-4 vertices, we need an observation:

Lemma 4 *Let F be a face with at most three unknown facial angles. Then there is a unique set of facial angles that can realize F, and it can be found in $O(\deg(F))$ time.*

Proof: Let v_1, v_2, v_3 be three vertices of F where the facial angle is not necessarily known. These divide the boundary of F into three polygonal chains C_1, C_2, C_3 such that all facial angles in the interior of the chains are known. Since we know edge lengths as well, each chain C_i is determined up to rotation and translation. In particular, chain C_i gives the distance between its endpoints, and hence we know the triangle formed by v_1, v_2, v_3. The angles of this triangle determine the relative rotations of the three chains to each other, and hence the facial angles. [2] $\square$

The algorithm to determine facial angles is now very simple: First determine all facial angles at vertices of degree 3 or 6. Then, for as long as there is a face F with at most three undetermined facial angles, determine its facial angles. For any vertex v incident to F for which now one facial angle is determined, update all other facial angles.

We claim that for a planar graph this determines all facial angles, and prove this as follows. Define an auxiliary graph H by using a node in H for every vertex and face of the graph of the polyhedron, and an edge from vertex v to face F if F is adjacent to v. Clearly H is bipartite and planar. Delete from H all nodes where the corresponding vertex/face has all facial angles identified during the algorithm, and let H' be the remaining graph.

Any node in H' that corresponds to a face F must have degree at least 4, otherwise the algorithm would have identified the facial angles of F. Any node in H' that corresponds to a vertex v must have degree 4, because v had degree 4 in G, and for a vertex either all or none of its facial angles are identified. So H' has minimum degree 4, contradicting that any planar bipartite graph has a vertex of degree at most 3.

Theorem 3 *Given an embedded planar graph with non-flat dihedral angles and edge lengths, we can in $O(m+n)$ time find the facial angles of any realizing orthogonal polyhedron, or report that no such polyhedron exists. Moreover, if a realizing polyhedron exists, then it is unique.*

This theorem involves edge lengths due to Lemma 4. A possible direction for future research is

[2]Computing the facial angles from the distances requires infinite precision arithmetic. This can be avoided by trying instead all possible orthogonal rotations of the chains.

to improve this lemma and avoid using edge lengths. We conjecture that the facial angles are determined solely by the graph and the dihedral angles.

Also, we assumed that the given dihedral angles are non-flat. If we allow flat dihedral angles, we can still easily reconstruct the facial angles at vertices that are vertices of the polyhedron (eliminate all flat dihedral angles by merging faces, and then apply the above algorithm.) On the other hand, reconstructing the facial angles at vertices in the interior of a face of the polyhedron is NP-hard, by reduction from the problem of reconstructing an orthogonal polygon from its edge lengths, which is NP-hard [3].

References

[1] M. Aigner and G. Ziegler. *Proofs from THE BOOK*. Springer Verlag, 1998 (1st edition), 2004 (3rd edition).

[2] T. Biedl and B. Genc. When can a graph form an orthogonal polyhedron? In *Canadian Conference on Computational Geometry (CCCG'04)*, pages 53–56, August 2004.

[3] T. Biedl, A. Lubiw, and J. Sun. When can a net fold to a polyhedron? *Computational Geometry: Theory and Applications*, 31(3):207–218, 2005.

[4] A. I. Bobenko and I. Izmestiev. Alexandrov's theorem, weighted Delaunay triangulations, and mixed volumes. http://arxiv.org/abs/math/0609447, 2006. arXiv:math/0609447v1 [math.DG].

[5] R. Connelly, I. Sabitov, and A. Walz. The bellows conjecture. *Beiträge zur Algebra und Geometrie*, 38(1):1–10, 1997.

[6] M. Damian, R. Flatland, and J. O'Rourke. Epsilon-unfolding orthogonal polyhedra. *Graphs and Combinatorics*, 23:179–194, 2007.

[7] P. Eades and S. Whitesides. The logic engine and the realization problem for nearest neighbor graphs. *Theoretical Computer Science*, 169:23–37, 1996.

[8] Maksym Fedorchuk and Igor Pak. Rigidity and polynomial invariant of convex polytopes. *Duke Math. Journal*, 129(2):371–404, 2005.

[9] Burkay Genc. *Reconstruction of Orthogonal Polyhedra*. PhD thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2008.

[10] Joseph O'Rourke. Computational geometry column 49. *ACM SIGACT News*, 38(2), June 2007.

[11] I. Kh. Sabitov. The volume of a polyhedron as a function of its metric and algorithmical solution of the main problems in the metric theory of polyhedra. In *International School–Seminar Devoted to the N.V. Efimov's Memory*, pages 64–65. Rostov University, 1996.

[12] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Ann. ACM Symp. Theory Computing*, pages 216–226, 1978.

A Disconnected graphs

In this appendix, we show that reconstructing the dihedral angles, given an embedded planar graph with edge lengths and facial angles, is NP-hard if the graph of the polyhedron is not connected. This was already shown earlier [2], but the proof there was from Partition, and hence left the possibility that the problem may be pseudo-polynomial in the edge lengths. We here use a reduction from NAE-3SAT, and hence establishes that the problem is in fact strongly NP-hard.

We use an approach called the *logic engine* [7], with a reduction from NAE-3SAT, which is the following problem: Given n boolean variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$ of three literals each, test whether there exists an assignment of values to variables such that each clause contains at least one true and at least one false literal. This problem is well-known to be NP-hard [12].

We explain how to construct the graph of the polyhedron (given an instance of NAE-3SAT) by showing parts of the polyhedron for which the graph is connected; by Theorem 2 there is no realization of this graph other than the one provided in the construction.

The first part of the polyhedron is the *frame*. This consists of a box that is $6n + 4$ units wide, $2m + 3$ units high and 5 units deep. At the leftmost and rightmost end of the top face, we place *towers* that are $2m + 2$ units high, 1 unit wide and 5 units deep. The top face of the box also contains n pairs of holes (one for each variable), which are 3 units wide, 1 unit deep and 1 unit apart. They are spaced 1 unit away from the towers and 3 units away from each other. See also Figure 7 (which is to scale for $n = m = 4$.)

The next part of the polyhedron is an *armature*. There are two types of armature, a *positive armature* and the *negative armature.* An armature is a block of height $2m + 1$, width 3 and depth 1, with an small $3 \times 1 \times 1$-box added at the bottom that connects it

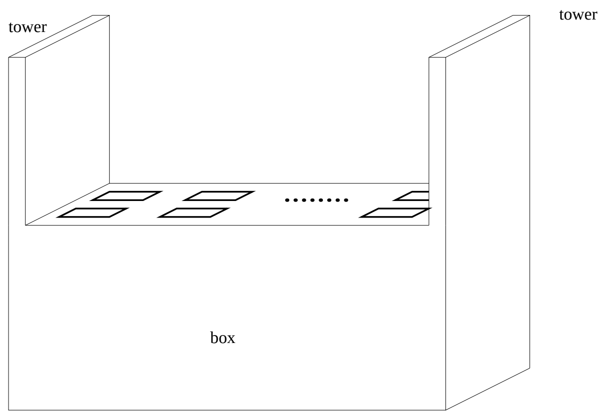

Figure 7: The frame. Thick lines mark a hole.

to the hole on the top of the box. This small box is placed at the front for the positive armature and at the back for the negative armature. See Figure 8.

For the ith pair of holes on top of the block, we attach a positive armature at the hole in the front, and a negative armature at the hole in the back; we call these the armatures of literal x_i and $\overline{x_i}$, respectively.

There are up to m holes each on the left and the right side of an armature. Each hole has height 1 and depth 1, and is spaced 1 unit away from all neighbouring edges or holes. We will discuss in a moment which holes actually exist. Figure 8, which is to scale for $m = 4$, shows an armature that has all possible holes.

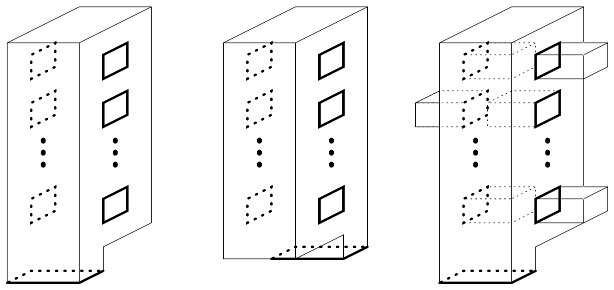

Figure 8: A positive armature, a negative armature, and an armature with flags attached. Thick lines mark a hole.

Label the possible position of holes from bottom to top as $1, 2, \ldots, m$. (Each such position is actually a pair of holes, one on the left and one on the right of the armature; either both or none will exist.) The holes at the jth position of the armature of literal ℓ_i exists if and only if ℓ_i does *not* occur in clause c_j.

Finally, into each hole of an armature, we place a *flag*, which is simply a $1 \times 2 \times 1$-cube.

We will now show that if the polyhedron can be realized without overlap, then the instance of NAE-3SAT is satisfiable. The other direction also holds and is proved similarly.

So assume the polyhedron can be realized without overlap. Set each literal ℓ_i to be true if and only if the armature of ℓ_i is up, i.e., the dihedral angles at the hole connecting the armature to the box are $270°$. Note that the armatures of x_i and $\overline{x_i}$ cannot both be up or both be down (otherwise they would intersect), so this is well-defined.

Consider clause c_j, and all flags in the jth possible position at the armatures that are up. The left tower so close to the leftmost armature that no flag can be between them. Likewise no flag can be between the rightmost armature and the right tower. Finally, the two flags of an armature cannot be both inside the armature, since they would overlap. This leaves space for only $2n - 1$ flags: there are $n - 1$ gaps between armatures, and n armatures. So if there is no overlap, then at least one armature has no flags in the jth position, so this literal occurs in c_j and has been assigned a true value, so c_j contains a true literal. By similarly arguing about the armatures that are down, we see that c_j contains at least one false literal, which proves the reduction.

In our reduction all edges have integer edge lengths, and hence by subdividing faces we can create an instance where all faces are unit squares; we hence establish NP-hardness of reconstructing so-called polycubes from their graph, a question raised by Craig Gotsman (private communication.)

Theorem 4 *Given a planar graph for which all faces are unit-length quadrangles, it is NP-hard to decide whether this graph is the graph of an orthogonal polyhedral surface.*