Boolean Operations on 3D Surfaces Using Nef Polyhedra

Miguel Granados Susan Hert Lutz Kettner Michael Seel

Max-Planck Institut für Informatik, Saarbrücken

1 Introduction

We describe our efforts in CGAL to implement a data structure for three-dimensional polyhedra. Our efforts are based on W. Nef's foundational work on polyhedra [5] that are closed under boolean operations and can represent all degeneracies and possible non-manifold situations. We give a short summary here, see also H. Bieri's introductory paper [1]. Note also that we already provide in CGAL an implementation for planar Nef-polyhedra in dimension two, see the detailed implementation report [7].

Definition 1.1 (Nef-polyhedron). A *Nef-polyhedron* in dimension d is a point set $P \subseteq \mathbb{R}^d$ generated from a finite number of open halfspaces by set complement and set intersection operations.

Set union, difference and symmetric difference can be reduced to intersection and complement. Set complement changes between open and closed halfspaces, thus the topological operations *boundary*, *interior*, *exterior*, *closure* and *regularization* are also in the modeling space of Nef-polyhedra.

Although the distinction of closed and open boundaries and the potential unboundedness of Nef-polyhedra might not be necessary in many applications, it is a good general framework and we do not have to argue about invalid intermediate representations as they occur in other descriptions. In the following we refer to Nef-polyhedra whenever we say polyhedra.

Definition 1.2 (Local pyramid). A point set $K \subseteq \mathbb{R}^d$ is called a *cone with apex* 0, if $K = \mathbb{R}^+ K$ (i.e., $\forall p \in K, \forall \lambda > 0 : \lambda p \in K$) and it is called a *cone with apex* x, $x \in \mathbb{R}^d$, if $K = x + \mathbb{R}^+(K - x)$. A cone K is called a *pyramid* if K is a polyhedron.

Now let $P \in \mathbb{R}^d$ be a polyhedron and $x \in \mathbb{R}^d$. There is a neighborhood $U_0(x)$ of x such that the pyramid $Q := x + \mathbb{R}^+((P \cap U(x)) - x)$ is the same for all neighborhoods $U(x) \subseteq U_0(x)$. Q is called the *local pyramid* of P in x and denoted $\mathrm{Pyr}_P(x)$.

Definition 1.3 (Face). Let $P \in \mathbb{R}^d$ be a polyhedron and $x, y \in \mathbb{R}^d$ be two points. We define an equivalence relation $x \sim y$ iff $\mathrm{Pyr}_P(x) = \mathrm{Pyr}_P(y)$. The equivalence classes of $\sim$ are the *faces* of P. The dimensions of a face s is the dimension of its affine hull, $\dim s := \dim \mathrm{aff}\, s$.

In other words, a *face* s of P is a maximal non-empty subset of $\mathbb{R}^d$ such that all of its points have the same local pyramid Q denoted $\mathrm{Pyr}_P(s)$. This definition of a face partitions $\mathbb{R}^d$ into faces of different dimension. A face s is either a subset of P, or disjoint from P. We use this later in our data structure and store a selection mark in each face indicating its set membership.

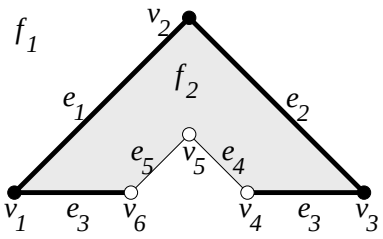

Figure 1: Planar example of a Nef-polyhedron. The shaded region, bold edges and black nodes are part of the polyhedron, thin edges and white nodes are not.

Faces do not have to be connected. There are only two full-dimensional faces possible whose local pyramids are the space $\mathbb{R}^d$ itself or the empty set. All lower-dimensional faces form the *boundary* of the polyhedron. As usual we call zero-dimensional faces *vertices* and one-dimensional faces *edges*. In the case of polyhedra in space we call two-dimensional faces *facets* and the full-dimensional faces *volumes*. Faces are *relative open* sets, e.g., an edge does not contain its endvertices.

Example 1.4. We illustrate the above definitions with an example in the plane. Given the closed halfspaces

$$h_1 : y \geq 0, \quad h_2 : x - y \geq 0, \quad h_3 : x + y \leq 3,$$
$$h_4 : x - y \geq 1, \quad h_5 : x + y \leq 2,$$

we define our polyhedron $P := (h_1 \cap h_2 \cap h_3) - (h_4 \cap h_5)$. Figure 1 illustrates the polyhedron with its partially closed and partially open boundary, i.e., vertex v_4, v_5, v_6, and edges e_4 and e_5 are not part of P. The local pyramids for the faces are $\mathrm{Pyr}_P(f_1) = \emptyset$ and $\mathrm{Pyr}_P(f_2) = \mathbb{R}^2$. Examples for the local pyramids of edges are the closed halfspace h_2 for the edge e_1, $\mathrm{Pyr}_P(e_1) = h_2$, and the open halfspace $\mathrm{cpl}\, h_4$ for the edge e_5, $\mathrm{Pyr}_P(e_5) = \{(x, y) | x - y < 1\}$. The edge e_3 consists actually of two disconnected parts, both with the same local pyramid $\mathrm{Pyr}_P(e_3) = h_1$. However, in our data structure we will represent the two connected components of the edge e_3 separately. Figure 2 lists all local pyramids for this example.

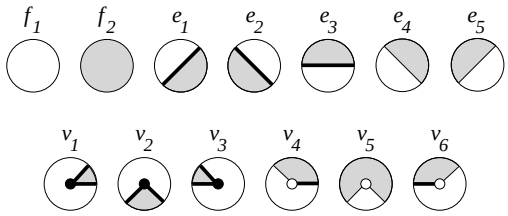

Figure 2: Sketches of the local pyramids of the planar Nef-polyhedron example. The local pyramids are indicated in the relative neighborhood in a small disc.

A first data structure proposed for Nef-polyhedra was the *Würzburg Structure* that stored all faces in the form of their local pyramids. It was observed that this representation lacked information about the relation between the different local pyramids, and the *Extended Würzburg Structure* added *incidences* for that purpose. We observe that for two faces, s and t, of a polyhedron P we have either that $s \cap \operatorname{clos} t = \emptyset$ or $s \subset \operatorname{clos} t$.

Definition 1.5 (Incidence relation). A face s is *incident* to a face t of a polyhedron P iff $s \subset \operatorname{clos} t$. This defines a partial ordering $\prec$ such that $s \prec t$ iff s is incident to t.

It actually suffices for the completeness of the data structure to store only the local pyramids of the minimal elements in the incidence relation $\prec$, the so-called *Reduced Würzburg Structure* [2]. For bounded polyhedra the minimal elements are always vertices. For unbounded, we might also get lines or planes in the three-dimensional case. It is worth noting that all variants of the Würzburg Structure are unique representations—two polyhedra are identical if and only if they have the same Würzburg Structure.

2 Bounding Nef-Polyhedra

The Reduced Würzburg Structure already indicates that it would be beneficial to have the polyhedron bounded, or more precisely, its boundary bounded. Another reason is that it would simplify the data types, for example, instead of segments, rays, and lines we will have only segments as edges.

Two approaches are possible: We extend the idea of an *infimaximal box* that has been used in the two-dimensional implementation to three dimensions [8], or we restrict the construction of polyhedra to objects of bounded boundary, which are also closed under the set operations and topological operations.

3 Sphere-Map Data Structure

We restrict the discussion now to three dimensions.

Using infimaximal boxes, or limiting the type of objects we start with, we can use the Reduced Würzburg structure and store only the local pyramids of vertices.

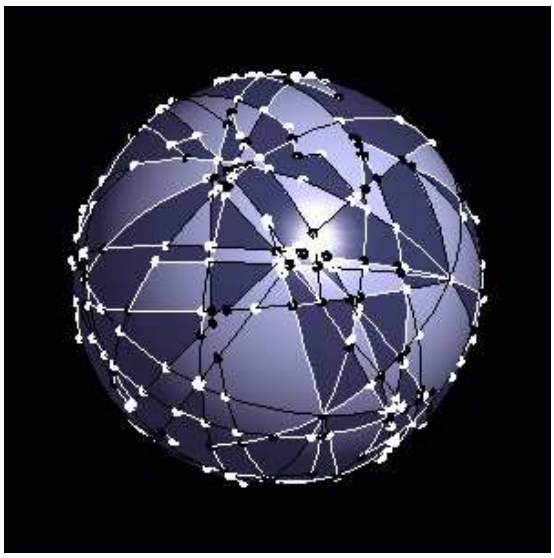

Figure 3: An example of a sphere map.

These local pyramids are represented by conceptually intersecting the local neighborhood with a small ε-sphere. This intersection forms a planar map on the sphere, see Figure 3, which together with the set-selection mark for each item forms a two-dimensional Nef-polyhedron embedded in the sphere. We add the set-selection mark for the vertex and call the resulting structure the *sphere map* of the vertex.

We use the prefix s to distinguish the elements of the sphere map from the three-dimensional elements; An *svertex* corresponds to an edge intersecting the sphere. An *sedge* corresponds to a facet intersecting the sphere. Geometrically the edge forms a great arc that is part of the great circle in which the supporting plane of the facet intersects the sphere. An *sface* corresponds to a volume.

We have extended the implementation of the planar Nef-polyhedra in CGAL to the sphere map. We do not actually project the geometry onto the sphere, but we work with three-dimensional operations on the original vertices. For the map overlay, we use the sweep-line algorithm separately on two hemispheres and simplify the result at the boundary later. The last distinction from planar polyhedra is the special situation of just one facet intersecting the sphere in a great circle. Currently, we represent this as a special case of a loop, called *sloop*, going around the sphere without any incident vertex. There is at most one *sloop* per vertex. A second *sloop* would intersect the first.

4 Selective-Nef-Complex Data Structure

Having sphere maps for all vertices of our polyhedron is a sufficient but not easily accessible representation of the polyhedron. We enrich the data structure with

more explicit representations of all the faces and incidences between them. We also depart slightly from the definition of faces in a Nef-polyhedron and represent the connected components of a face individually. We also do not implement additional bookkeeping to recover the original faces, e.g., all edges on a common supporting line with the same local pyramid. The faces and incidences added are in the order of increasing dimension; see Figure 4 for an example:

edges: We store two oppositely oriented edges for each edge and have a pointer from one oriented edge to its opposite edge. Such an oriented edge can be identified with an *svertex* in a sphere map; it remains to link one *svertex* with the corresponding opposite *svertex* in the other sphere map.

edge uses: An edge can have many incident facets (non-manifold situation). We introduce two oppositely oriented edge-uses for each incident facet; one for each orientation of the facet. An edge-use points to its corresponding oriented edge and to its oriented facet. We can identify an edge-use with an oriented *sedge* in the sphere map, or, in the special case also with an *sloop*. We won't mention it further, but all references to *sedge* can also refer to *sloop*.

facets: We store oriented facets as their boundary cycles of oriented edge-uses. We have a distinguished outer boundary cycle and several (or maybe none) inner boundary cycles representing holes in the facet. Boundary cycles are linked in one direction. We can access the other traversal direction when we switch to the oppositely oriented facet, i.e., by using the opposite edge-use.

shells: Facets around an edge form a radial order. This order is captured in the sphere map with the order of *sedges* around an *svertex*. Using this information, we can find from one oriented facet the next oriented facet along a common edge. This adjacency information of facets defines equivalence classes of facets, the *shells*. A shell is just represented with a starting point, i.e., one oriented facet.

volumes: A volume is defined by a set of shells, one outer shell containing the volume and several (or maybe none) inner shells excluding voids from the volume.

For each face we store the set-selection mark, which indicates whether the face is part of the modeled point set or if it is excluded. We call the resulting data structure *Selective Nef Complex*, *SNC* for short.

Dobrindt et al. [3] describe a data structure similar to the sphere-map data structure. Gursoz et al. [4] describe a data structure that similarly records the complete topology in the neighborhood of vertices. They also explain why the radial edge data structure by

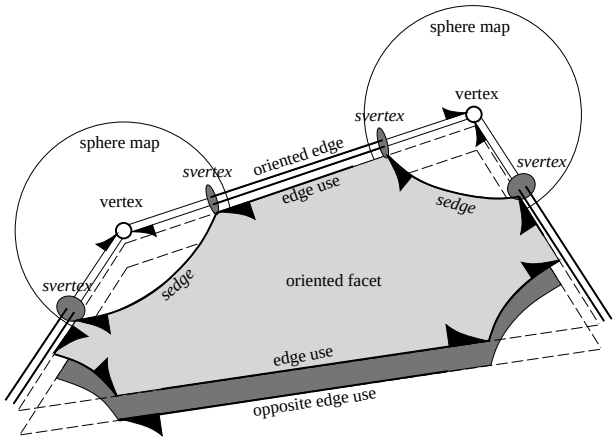

Figure 4: An example of a Selective Nef Complex. We show only one facet with two vertices, their sphere maps, the connecting edges, and both oriented facets. Shells and volumes are omitted for this example.

Weiler [9] is not sufficient in describing topology. Selective geometric complexes (SGC) by Rossignac and O'Connor are similar to Nef, but for general semi-algebraic varieties [6]. If we assume halfspaces as varieties, we get the connected components of Nef faces as cells in their data structure.

5 Synthesizing the SNC from Sphere Maps

An important step is the synthesis of the SNC from a set of sphere maps. It also can help in understanding the SNC data structure. The synthesis works in the order of increasing dimension:

1. We identify *svertices* that we want to link together as edges. We form an encoding for each *svertex* consisting of: (a) a normalized line representation for the supporting line, e.g. the normalized Plücker coordinates of the line, (b) the vertex coordinates, (c) a $+1$ or -1 indicating whether the normalization of the line equation reversed its orientation compared to the orientation from the vertex to the *svertex*. We sort all encodings lexicographically. Consecutive pairs in the sorted sequence form an edge (if we got the ± 1 sign right).
2. Edge-uses correspond to *sedges*. They form cycles around *svertices*. The cycles around two *svertices* linked as an edge have opposite orientations. Thus, corresponding *sedges* are easily matched up and we have just created all boundary cycles needed for facets.
3. We sort all boundary cycles by their normalized, oriented plane equation. We find the nesting relationship for the boundary cycles in one plane with a conventional two-dimensional sweep line algorithm.

4. Shells are found with a graph traversal. The nesting of shells is resolved with ray shooting from the lexicographically smallest vertex. Its sphere map also gives the set-selection mark for this volume by looking at the mark in the sphere map in $-x$ direction. This concludes the assembly of volumes.

6 Boolean Operations

We represent Nef-polyhedra in an SNC data structure. We can trivially construct an SNC for a halfspace. We can also construct it from a polyhedral surface representing a closed 2-manifold by constructing sphere maps first and then synthesizing the SNC as explained in the previous section.

Based on the SNC data structure, we can implement the boolean set operations. For the set complement we just reverse the set-selection mark for all vertices, edges, facets, and volumes. For the binary boolean set operations we find all vertices of the result with their sphere maps and synthesize the SNC. We consider all vertices of the original polyhedra, as well as all edge-edge and edge-face intersections.

For each vertex, we locate its position in the other polyhedra. If that position happens to be a vertex as well, we can apply our boolean set operation to the two sphere maps. The resulting sphere map can simplify to a situation that is not a local pyramid of a vertex anymore (but, for example, of a facet or volume, which are exactly those cases also used in the simplification procedure mentioned above), in which case we drop the vertex. Otherwise we keep it for our synthesis algorithm.

If the located position is not a vertex, we first construct a sphere map of the situation found, i.e., an edge, facet, or volume, before we continue in the same manner as for the vertex case.

For edge-edge or edge-facet intersections, we construct sphere maps for the edge or facet in each polyhedra, and continue in the same manner as for the vertex case.

We can also easily implement the topological operations *boundary*, *closure*, *interior*, *exterior*, and *regularization*. For example, for the boundary we deselect all volume marks and simplify the remaining SNC. The uniqueness of the representation implies that the test for the empty set is trivial. As a consequence, we can implement for polyhedra P and Q the subset relation as $P \subset Q \equiv P - Q = \emptyset$, and the equality comparison as the symmetric difference $P = Q \equiv (P - Q) \cup (Q - P) = \emptyset$.

The simplification works just as it does for the planar case; the cases we can locally detect and simplify are just a bit different. We also need more union-find data structures to maintain facets, shells and volumes efficiently during the simplification.

7 State of the Implementation

The sphere maps and SNC data structures are finished. We can construct polyhedra from bounded polyhedral surfaces, i.e., `CGAL::Polyhedron_3`. The extended geometry kernel is not done yet, but poses no particular problems. It can be based on the building blocks we can reuse from the planar Nef-polyhedra.

We have the synthesis algorithm finished except for the volume assignment; it is used in the construction from bounded polyhedral surfaces. We have inefficient but complete implementations for the ray-shooting required in the synthesis algorithm, and for the point location and intersection location in the boolean operations. These implementations test all possible combinations.

We plan to finish the synthesis algorithm and the simplification algorithm next. This allows us to realize and test all operations on Nef-polyhedra in space. From there, we will work on improving the performance of the ray-shooting, the point location, and the intersection location. The goal is a complete, exact, correct, and efficient implementation of boolean operations on polyhedra in space. However we should say that such a general structure will not be able to compete with specialized structures for 2-manifold meshes, such as the polyhedral surface in CGAL.

References

[1] H. Bieri. Nef polyhedra: A brief introduction. *Computing Suppl. Springer Verlag*, 10:43–60, 1995.

[2] H. Bieri. Two basic operations for Nef polyhedra. In *CSG 96: Set-theoretic Solid Modelling: Techn. and Appl.*, pages 337–356, Winchester, April 1996. Information Geometers.

[3] K. Dobrindt, K. Mehlhorn, and M. Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *LNCS*, pages 314–324, 1993.

[4] E. L. Gursoz, Y. Choi, and F. B. Prinz. Vertex-based representation of non-manifold boundaries. *Geom. Modeling for Product Engineering*, 23(1):107–130, 1990.

[5] W. Nef. *Beiträge zur Theorie der Polyeder.* Herbert Lang, Bern, 1978.

[6] J. R. Rossignac and M. A. O'Connor. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In M. Wozny, J. Turner, and K. Preiss, editors, *Geom. Modeling for Product Engineering.* North-Holland, 1989.

[7] M. Seel. Implementation of planar Nef polyhedra. Research Report MPI-I-2001-1-003, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, August 2001.

[8] M. Seel and K. Mehlhorn. Infimaximal frames: A technique for making lines look like segments. Research Report MPI-I-2000-1-005, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, December 2000.

[9] K. Weiler. The radial edge structure: A topological representation for non-manifold geometric boundary modeling. In M. J. Wozny, H. W. McLaughlin, and J. L. Encarnaçao, editors, *Geom. Modeling for CAD Appl.*, pages 3–36, Rensselaerville, NY, May 12–16 1988. IFIP, North Holland.