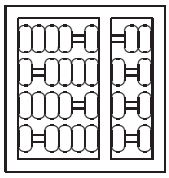

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

General convex hull using the gem data structure

Arnaldo J. Montagner *Jorge Stolfi*

Technical Report - IC-06-011 - Relatório Técnico

May - 2006 - Maio

General convex hull using the gem data structure*

Arnaldo J. Montagner† Jorge Stolfi †

Abstract

We describe in detail a general algorithm for constructing the convex hull of a finite set of points in Euclidean space of arbitrary dimension n. The algorithm handles degenerate situations, such as non-simplicial faces and point sets contained in a lower-dimensional subspace. The topology of the hull is kept in a *graph encoded map* (*gem*) *data structure*, a novel representation for n-dimensional triangulations. The gem representation, which was introduced as a mathematical device by S. Lins in 1982, extends the *cell-tuple* (or *generalized map*) representation proposed by Brisson and Lienhardt to maps that are not barycentric subdivisions, to manifolds with borders, and to non-manifold (but triangulable) topological spaces.

1 Introduction

Computing the convex hull of a set of points is a central problem in computational geometry, not only for its importance in practical applications [3, 6, 11, 13], but also for being a natural and intuitive concept. In this article we describe a program that computes the convex hull of a finite set of points $S \subset \mathbb{R}^n$, for any $n \geq 1$, and represents it by a data structure called *gem*.

1.1 Challenge

The challenge of our implementation is to treat the occurrence of more than $k + 1$ points contained in the same k-dimensional space — for example, 3 co-linear points, 4 co-planar points, etc.

Many geometric algorithms assume that the input points are in a non-degenerate position. Sweep line algorithms, for example, might require that no two points share the same x coordinate, and most Voronoi diagram algorithms assume that no four points are on the same circle. This assumption simplifies the algorithm, eliminating the necessity of treating many exceptional cases.

The occurrence of such degenerate cases is generally handled by the *perturbation method* introduced by Edelsbrunner [5] and elaborated by many others, such as Yap [14] and Emiris [7]. This approach consists of adding symbolic infinitesimal values to the points' coordinates, eliminating any degeneracy in their position. The main disadvantage of this

*Research supported in part by FAPESP

†Institute of Computing, University of Campinas, 13081-970 Campinas, SP.

method is that the final result of the algorithm depends on the perturbations used. Moreover, handling the perturbations can increase considerably the complexity of the geometric primitives.

1.2 Our solution

Our application deals with the degenerate cases without using perturbations. We implemented a recursive convex hull algorithm that allows the occurrence of non-simplicial faces. The input points have integer coordinates and the geometric computations are performed with exact integer arithmetic.

2 Basic concepts

2.1 Polytopes

Convex hull: The *convex hull* of a set of points $S \subset \mathbb{R}^n$ is the smallest convex set containing S.

Polytope: A subset of $\mathbb{R}^n$ is called a *convex polytope*, or simply *polytope*, if it is the convex hull of a finite set of points of $\mathbb{R}^n$. A *k-polytope* is a polytope of dimension k.

Simplex: A *k-simplex* is a k-polytope that is the convex hull of $k+1$ points. Note that those points must be affinely independent, that is, not contained in a $(k-1)$-dimensional affine space.

Supporting hyperplane: A hyperplane $h \subset \mathbb{R}^n$ *supports* a set of points $S \in \mathbb{R}^n$ if $h \cap S \neq \emptyset$ and S is contained in one of the closed half-spaces determined by h.

Faces of a polytope: Let P be an n-polytope. $F \subseteq P$ is a *face* of P if F is $\emptyset$, or P itself, or the intersection of P and a supporting hyperplane. In that case we say that P is *incident* to F. If F has dimension k, we say it is a *k-face* of P. The faces with dimension 0, 1, $n-2$ and $n-1$ are called, respectively, *vertices*, *edges*, *ridges* and *facets* of P.

Notice that each k-face F of a polytope P is a k-polytope. This implies that F has its own faces of dimension $\leq k$, which are also faces of P. The incidence relation is therefore a partial order on the faces of P, with one minimal element ($\emptyset$) and one maximal element (P).

2.2 Complexes

Polytope complex: A *complex of n-polytopes*, or *n-complex*, is a set of n-polytopes $C = \{P_1, \ldots, P_m\}$ such that:

(1) if P_i and P_j are distinct elements of C, then $P_i \cap P_j$ is $\emptyset$ or a k-face of both, with $k < n$;

(2) if F is a facet of $P_i \in C$, there is at most one other polytope of C incident to F.

Faces of a complex: The set of faces of an n-complex $C = \{P_1, \ldots, P_m\}$ is the union of the sets of faces of the polytopes $P_1, \ldots, P_m$.

Quasi-complex: An *n-quasi-complex* is a set of n-polytopes $C = \{P_1, \ldots, P_m\}$ that satisfies the condition (1) of complexes, but not the condition (2). That is, if P_i and P_j are distinct elements of C, then $P_i \cap P_j$ is $\emptyset$ or a k-face of both, with $k < n$; however, if F is a facet of $P_i \in C$, there can be two or more polytopes of C, distinct from P_i, incident to F.

Border of a complex: The *border* of an n-complex C, denoted by $\partial^\star C$, is the quasi-complex formed by the $(n-1)$-faces of C that are not shared by two elements of C. By extension, the border $\partial^\star P$ of a single n-polytope P is the $(n-1)$-complex consisting of the facets of P.

Triangulation: A *n-triangulation* is an n-complex composed by n-simplices.

3 The gift-wrapping algorithm

3.1 General view

The program described in this article implements a recursive version of the *gift-wrapping* algorithm of Chand and Kapur [2]. The basic idea of this technique is to find an initial facet and then proceed from a facet to an adjacent one, until the entire convex hull border is computed. Figure 1 illustrates this idea for the 3-dimensional case.

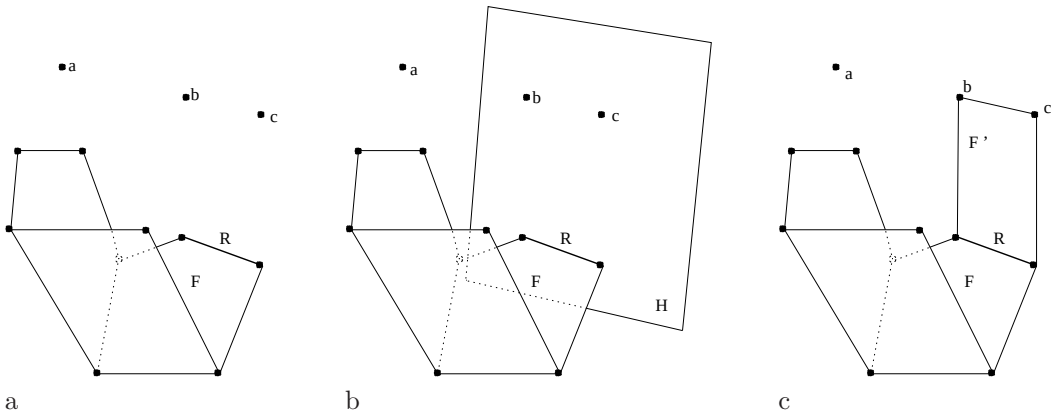

Figure 1: The gift-wrapping algorithm in three dimensions.

During the execution of the algorithm we accumulate a set C of already determined facets, which is an $(n-1)$-complex. We also maintain the $(n-2)$-quasi-complex Q of the elements of $\partial^\star C$.

The first step, finding the initial facet, requires finding a supporting hyperplane H_1 of S that contains n affinely independent points of S, and building recursively the convex hull

of $H_1 \cap S$, which is a facet F_1 of P. The set C is initialized with the polytope F_1, and the set Q with $\partial^\star F_1$. Each subsequent step consists in the following operations:

(1) Choose an $(n-2)$-polytope R of Q, that is facet of an $(n-1)$-polytope $F \in C$. See figure 1(a).

(2) Find geometrically a supporting hyperplane H of S that contains R and at least one point of S not in F. See figure 1(b) (the the points b and c are on H).

(3) Compute recursively the convex hull of $H \cap S$, which is a facet F' of P. See figure 1(c).

(4) Add F' to the complex C.

(5) Update the set Q, adding the facets of F' that are not in Q, and removing the facets of F' that are already in Q.

These five steps are repeated until the set Q (the border of complex C) becomes empty.

3.2 Finding the adjacent facet

The fundamental geometric operation of this algorithm consists in, given a facet F of the convex hull P and a facet R of F, finding the hyperplane H that contains the other facet F' of P that is incident to R.

Let G be the hyperplane of F and let p be a point of S not contained in G. We will denote by α_p the semi-hyperplane containing p whose border contains R. The convex angle between α_p and G will be denoted by $\theta(p)$. Figure 2 shows an example for the 3-dimensional case.

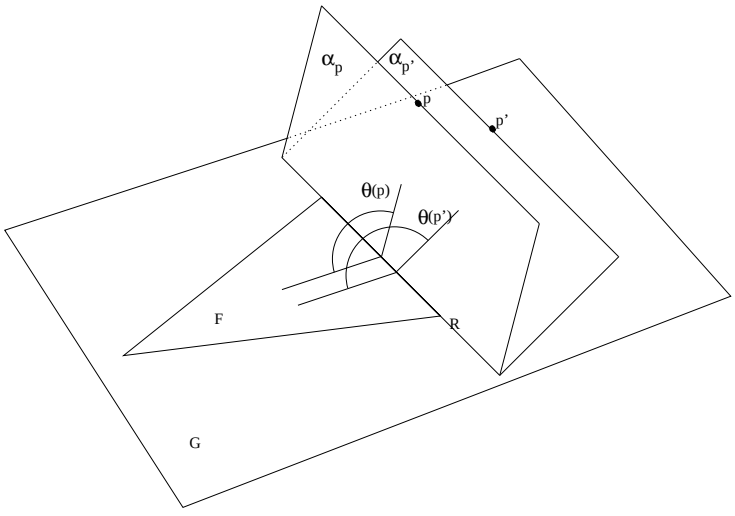

Figure 2: The angle $\theta(p)$ formed between the hyperplane G and the semi-hyperplane α_p

The supporting hyperplane H contains R and a point $p \in S$ with maximum $\theta(p)$. In our implementation the angle comparison is carried out as follows.

Let K be any hyperplane distinct from G that also contains the face R. The hyperplanes G and K are defined by affine (degree one) polynomials G and K on the coordinates of a generic point p such that $G(p) = 0$ iff $p \in G$ and $K(p) = 0$ iff $p \in K$. We can choose the signs of G and K so that $G(p) \geq 0$ for any $p \in S$, and $K(q) \geq 0$ for any $q \in F$.

Then it is easy to show that, for any p and $q \notin G$, $\theta(p) < \theta(q)$ if and only if $\frac{K(p)}{G(p)} > \frac{K(q)}{G(q)}$.

Hyperplane G can be found by taking $n-1$ affinely independent points $r_1, r_2, \ldots, r_{n-1}$ on R and a point t of F and evaluating the symbolic determinant

$$\begin{vmatrix} 1 & r_1 \\ & \vdots \\ 1 & r_{n-1} \\ 1 & t \\ 1 & X_1\ X_2\ \cdots\ X_n \end{vmatrix}$$

where the X_i are symbolic variables representing the coordinates of the generic point p. Hyperplane K can be obtained in a similar way, except that t is replaced by any point of S not on G.

In our algorithm, we denote by `AdjacentFacet(`S`,`G`,`K`)` the operation that takes the set of points S and hyperplanes G and K and produces the hyperplane H.

3.3 Finding the initial facet

We define now the operation `InitialFacet(`S`)` that determines a supporting hyperplane of the convex hull P of S containing at least n affinely independent points of S. The idea is to find successive supporting hyperplanes of P, each one containing a face of greater dimension than the previous, until we get a hyperplane containing a facet of P.

First we find a point p_0 of S with least x_0 value, where $x_0, x_1, \ldots, x_n$ are the points' coordinates. The first supporting hyperplane, called H_1, is the hyperplane normal to the $\vec{x}_0$ axis that contains p_0.

Let's now assume we have found the supporting hyperplane H_{i-1}. The supporting hyperplane H_i is determined as follows. If H_{i-1} contains i affinely independent points of S, then $H_i = H_{i-1}$. Otherwise, we find a point q of S not in H_{i-1} and create a hyperplane K containing q and $H_{i-1} \cap S$, parallel to the axes $\vec{x}_i, \vec{x}_{i+1}, \ldots, \vec{x}_n$. Then, $H_i =$ `AdjacentFacet(`S`,`H_{i-1}`,`K`)`.

Finally, H_n is a supporting hyperplane of P containing n affinely independent points of S.

4 Barycentric subdivision

Convex polytopes can have a very complicated combinatorial structure. For $n \geq 2$, there is no limit on the number of facets of a n-polytope. For this reason, it is convenient to

subdivide the polytopes in less complex objects in order to represent them. We employ a special subdivision, the *barycentric subdivision*.

Let $C = \{P_1, \ldots, P_m\}$ be a n-complex. A *flag* of C is a tuple $(F_0, F_1, \ldots, F_n)$ of faces of C such that each F_i has dimension i, and F_i is facet of F_{i+1}, for $0 \leq i < n$. The barycentric subdivision of C, denoted by ΔC, is a n-triangulation such that for each flag $(F_0, \ldots, F_n)$ of C there is a n-simplex of ΔC, whose vertices are the barycenters of faces $F_0, \ldots, F_n$.

If the vertex v of ΔC is the barycenter of a i-dimensional face of C, we say it has *type* i, or it is a *i-type* vertex. If d is a k-face of ΔC with vertices of types $i_1, i_2, \ldots, i_k$, we say that d has type $\{i_1, i_2, \ldots, i_k\}$. The simplices A and $B \in \Delta C$ are *i-adjacent* if they share a facet of type $\{0, \ldots, n\} \setminus \{i\}$.

The i-adjacency relations of the barycentric subdivision carries enough information to reconstruct the topology (incidence relations) of the original polytope complex. Each k-face F of a complex C can be obtained by the union of the k-faces of ΔC of type $\{0, \ldots, k\}$ incident on a same k-type vertex, which is the barycenter of F.

Although the barycentric subdivision creates many additional vertices, our program does not have to compute their coordinates. In fact, the only points used in geometric calculations are the ones given as input.

5 Gems

To represent the topology of polytope complexes, we use a novel representation for n-dimensional triangulations, the *graph encoded map* (*gem*) *data structure*. This structure is used for the final convex hull and for the partial hulls that exists during the algorithm's execution.

The gem representation, which was introduced as a mathematical device by S. Lins in 1982 [12], extends Brisson's *cell-tuple* representation [1] and Lienhardt's *generalized maps* [10] to triangulations that are not barycentric subdivisions, to manifolds with borders, and to non-manifold (but triangulable) topological spaces. However, in this paper we use the gem structure to encode the topology of a polytope complex C, or (equivalently) the adjacency relations of the simplices of ΔC. General gem structures will be discussed in separate papers.

5.1 Definition

Formally, an *n-dimensional gem* or n-gem is a tuple $(V, \phi_0, \ldots, \phi_n)$ where V is a finite set of gem *nodes*, and each ϕ_i is an involution of V.

A gem $(V, \phi_0, \ldots, \phi_n)$ is said to *represent* an n-complex $C = \{P_1, \ldots, P_m\}$ if there is a bijection δ between the set V and the n-simplices of ΔC, such that, for each i between 0 and n, $\phi_i(v) = w$ if and only if **(1)** $\delta(v)$ and $\delta(w)$ are distinct i-adjacent n-simplices or **(2)** $v = w$ and $\delta(v)$ is the only n-simplex of ΔC incident to its facet of type $\{0, \ldots, n\} \setminus \{i\}$.

Note that, in a gem that represents an n-complex C, the functions $\phi_0, \ldots, \phi_{n-1}$ are strict involutions of V, that is, for each node $v \in V$, $\phi_i(v) \neq v$ for $0 \leq i \leq n-1$. The function ϕ_n, however, can have fixed points, corresponding to the elements of $\partial^\star C$.

5.2 Gems as colored graphs

We can also interpret a gem $(V, \phi_0, \ldots, \phi_n)$ as a non-directed graph with colored edges, where V is the set of graph nodes and there is a i-colored edge between the nodes v and $w \in V$ if and only if $\phi_i(v) = w$. See figure 3. This interpretation provides implicitly the concepts of *path*, *cycle*, *connectedness*, etc.

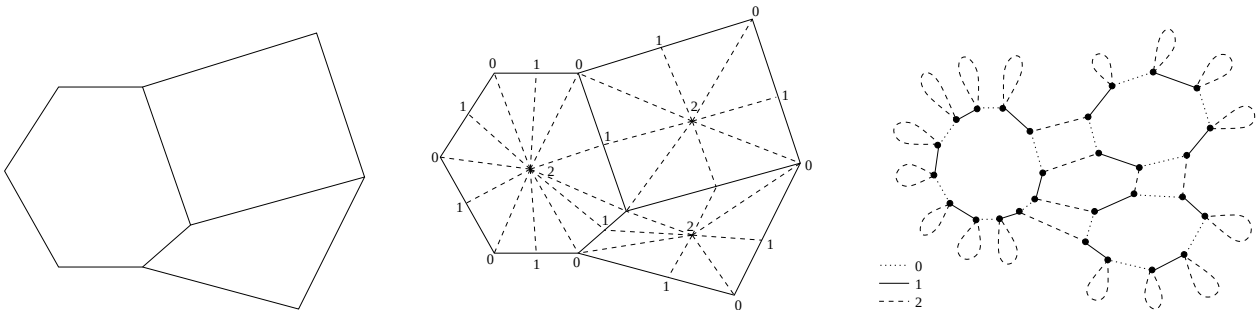

Figure 3: (a) A 2-complex C with three polygons (b) The barycentric subdivision of C (c) The colored graph of the gem of C

5.3 Isomorphism of gems

Let $G = (V, \phi_0, \ldots, \phi_n)$ and $G' = (V', \phi'_0, \ldots, \phi'_n)$ be two gems. An *isomorphism* between G and G' is a bijection f between V and V' such that, for each $v \in V$ and $i \in \{0, \ldots, n\}$, $\phi_i(v) = w$ if and only if $\phi'_i(f(v)) = f(w)$.

An isomorphism f between the connected gems $G = (V, \phi_0, \ldots, \phi_n)$ and $G' = (V', \phi'_0, \ldots, \phi'_n)$ is determined by any pair of nodes a and $a' = f(a)$, in the following way: if $x \in V$ and $x = \phi_{i_1}(\phi_{i_2}(\ldots \phi_{i_s}(a) \ldots))$, for $i_1, i_2, \ldots, i_s \in \{0, \ldots, n\}$, then $f(x) = \phi'_{i_1}(\phi'_{i_2}(\ldots \phi'_{i_s}(a') \ldots))$.

5.4 The gem of a complex

When gems are viewed as abstract graphs, it is convenient to consider two gems to be the same if they differ only by a color-preserving isomorphism. In that view, its obvious that each complex C has a unique representative gem, called *the gem of* C and denoted by $\mathcal{G}C$. By extension, if P is a single polytope, we define *the gem of* P, denoted by $\mathcal{G}P$, as being the gem of its border, $\mathcal{G}\partial^\star P$.

6 The gem data structure

In an arbitrary n-triangulation, there are $n!$ ways in which a specific facet of an n-simplex can be glued to a specific facet of another n-simplex. Therefore, in data structures for general n-triangulations, one must use $\lceil \log_2(n!) \rceil$ bits for each adjacent pair, in order to encode this information [4, 8]; or, alternatively, this information must be recomputed at each step when the structure is traversed [9]. In a barycentric subdivision, however, the

constraints on vertex types remove the need for that information, since two simplices can share a specific facet in only one way. The gem data structure makes use of this constraint to greatly simplify the repertoire of elementary topological operators.

The gem data structure consists of records corresponding to the gem nodes. Each record r contains $n+1$ pointers to other records, representing the functions $\phi_0, \ldots, \phi_n$. We will write $\phi_i(r)$ to mean the record referenced by pointer i of record r. Each record r may also contain application-specific non-topological information $data(r)$.

The only constraint on this structure is that, if the pointer i of record r points to record r', then the pointer i of r' points to r.

The gem data structure of a complex will be referred by a pair (r, k), where r is a record and k is the dimension of the complex's elements. The gem represented by this pair is the connected component of the graph inducted by the edges of color between 0 and k. Then, if $k < n$, the pointers $k+1, \ldots, n$ of each record will be ignored.

7 Basic operations

Any gem data structure can be built using only two elementary operations.

`MakeNode():` creates a new record v and makes $\phi_i(v) = v$ for $0 \leq i \leq n$. This operation can be interpreted as adding to the triangulation a new n-simplex not adjacent to any other n-simplex.

`SpliceNodes(`a`,`b`,`k`):` where a and b are records and $k \in \{0, \ldots, n\}$. The precondition for `SpliceNodes` is that $\{\phi_k(a)\} \bigcup \{\phi_k(b)\} = \{a, b\}$. This operation exchanges the values of $\phi_k(a)$ and $\phi_k(b)$. Thus, if the nodes are not adjacent ($\phi_k(a) = a$ and $\phi_k(b) = b$), they become adjacent ($\phi_k(a) = b$ and $\phi_k(b) = a$), and vice-versa. Note that, this operation is its own inverse, and it is a no-op if $a = b$.

8 Attaching polytopes

In our application, the gem data structure is used to represent the barycentric subdivision of an n-complex C. We need an operation that attaches two polytopes P and Q of this complex by identifying a facet of P with a facet of Q, which is actually the same $(n-1)$-polytope.

`SplicePolytopes(`k`,`a`,`b`):` a and b are records such that (a, k) e (b, k) are gems of distinct k-polytopes P and Q; and a and b determine an isomorphism f between the gems $(a, k-1)$ and $(b, k-1)$.

This operation applies `SpliceNodes(`x`,`$f(x)$`,`k`)` for each record x of $(a, k-2)$, which causes two possible effects, depending on the original state of the gem:

1) If, for each record v of $(a, k-2)$, $\phi_k(v) = v$ and $\phi_k(f(v)) = f(v)$, then $\phi_k(v) := f(v)$ and $\phi_k(f(v)) := v$.

2) If, for each record v of $(a, k-2)$, $\phi_k(v) = f(v)$ and $\phi_k(f(v)) = v$, then $\phi_k(v) := v$ and $\phi_k(f(v)) := f(v)$.

Like `SpliceNodes`, `SplicePolytopes` is its own inverse.

9 The vertex tuple of a node

Let P_1 be a n-polytope and let G_1 be the gem of P_1. The *vertex tuple* of the node a of G_1, denoted by $T(a)$, is a tuple $(v_0, v_1, \dots v_n)$, where $v_0 = data(a)$, $v_1 = data(\phi_0(a))$,..., $v_n = data(\phi_0(\phi_1(\cdots \phi_{n-1}(a)\cdots)))$. We will also denote by $T_k(a)$ the *partial vertex tuple* consisting of the first $k+1$ vertices of $T(a)$.

Vertex tuples provide simple solutions for three issues of the algorithm:

1) If G_1 and G_2 are gems of polytopes P_1 and P_2 of a quasi-complex, how to decide whether $P_1 = P_2$?

2) If G_1 and G_2 are gems of the same polytope, how to identify a pair of nodes $a \in G_1$ and $b \in G_2$ that determines an isomorphism that preserves geometric information between G_1 and G_2?

3) If G is the gem of a polytope P, how to obtain $n+1$ affinely independent vertices of P?

The answers to these questions are provided by theorems 1, 2, and 3 below. We will need the following concept:

Flag of a node: According to the definition in section 4, there is a one-to-one correspondence between the n-simplices of ΔC and the flags of C. It follows that each node v of $\mathcal{G}C$ has a unique corresponding flag of C, which will be denoted by $Flag(v)$.

Lemma 1 *If $T(a) = (v_0, \dots, v_n)$ and $Flag(a) = (F_0, \dots, F_{n-1})$ then, for $0 \le i \le n-1$, the vertices $v_0, \dots, v_i$ are faces of F_i, but the vertex v_{i+1} is not.*

Proof :
Let δ be the bijection between the nodes of G_1 and the simplices of $\Delta\partial^\star P_1$. First we give the inductive proof that the vertices $v_0, \dots, v_i$ are faces of F_i.

base: v_0 is the 0-type vertex of the simplex $\delta(a)$. Hence the face F_0 is v_0 itself.

hypothesis: $v_0, \dots, v_{i-1}$ are faces of F_{i-1}.

step: By the definition of *Flag*, F_{i-1} is a face of F_i. So the vertices $v_0, \dots, v_{i-1}$ are faces of F_i. Let x be the node $\phi_0(\cdots \phi_{i-1}(a)\cdots)$ and let $Flag(x) = (F'_0, \dots, F'_{n-1})$. The simplices $\delta(a)$ and $\delta(x)$ share a face of type $\{i, \dots, n-1\}$, thus they share the i-type vertex, which is the barycenter of F_i. v_i is the 0-type vertex of $\delta(x)$, thus $F'_0 = v_i$ and $F'_i = F_i$. Therefore, v_i is a face of F_i.

Now we give the inductive proof that v_{i+1} is not a face of F_i.

base: v_{i+1} is not a face of F_0.

Let x be the node $\phi_1(\cdots\phi_i(a)\cdots)$. The simplices $\delta(a)$ and $\delta(x)$ share the 0-type vertex v_0. Considering that ϕ_0 is a strict involution, $\phi_0(x) \neq x$. The simplices $\delta(\phi_0(x))$ and $\delta(x)$ share a facet of type $\{1,\ldots,n-1\}$, so $\delta(\phi_0(x))$ and $\delta(x)$ have distinct 0-type vertices. Since the 0-type vertex of $\delta(\phi_0(x))$ is v_{i+1}, v_0 and v_{i+1} are distinct.

hypothesis: the vertex v_{i+1} is not a face of F_{i-1}.

step: Let x be the node $\phi_i(a)$. ϕ_i is a strict involution, so $x \neq a$. $\delta(x)$ and $\delta(a)$ share a facet of type $\{0,\ldots,n-1\}\setminus\{i\}$, thus, their i-type vertices are distinct. Let's say that the i-type vertex of $\delta(x)$ is the barycenter of the i-face $F_i' \neq F_i$. Since $\delta(x)$ and $\delta(a)$ share a $(i-1)$-type vertex, which is the barycenter of F_{i-1}, F_i and F_i' share the facet F_{i-1}.

Let then y be the node $\phi_0(\cdots\phi_{i-1}(x)\cdots)$. v_{i+1} is the 0-type vertex of $\delta(y)$. $\delta(x)$ and $\delta(y)$ share the i-type vertex, so v_{i+1} is face of F_i'. Since F_i and F_i' are facets of a $(i+1)$-polytope, their only intersection is F_{i-1}. By the induction hypothesis, v_{i+1} is not a face of F_{i-1}, thus v_{i+1} is not a face of F_i.

□

Lemma 2 *$Flag(a)$ can be determined from $T(a)$.*

Proof: Let $T(a) = (v_0,\ldots,v_n)$ and $Flag(a) = (F_0,\ldots,F_{n-1})$. The face F_0 is the vertex v_0. Let's assume we have already determined F_{i-1}. The polytope F_i is the i-face of P incident to F_{i-1} and v_i. Since the intersection of two i-faces of P is at most one face, and v_i is not a face of F_{i-1}, there is no other i-face of P incident to both F_{i-1} and v_i. □

Lemma 3 *Let C be a n-quasi-complex containing the polytopes P_1 and P_2 and let G_1 and G_2 be the gems of P_1 and P_2. If a_1 and a_2 are nodes of G_1 and G_2, and $T(a) = T(b)$, then $Flag(a) = Flag(b)$.*

Proof: This statement follows from the lemma 2. □

Natural isomorphism: If G_1 and G_2 are gems of the same n-complex C there is a single isomorphism f_n between G_1 and G_2 such that, for each node x of G_1, $data(x) = data(f_n(x))$. We say that f_n is the *natural isomorphism* between G_1 and G_2. Notice that, for each node x of G_1, x and $f_n(x)$ correspond to the same flag of C.

Theorem 1 *Let C be a n-quasi-complex containing the polytopes P_1 and P_2 and let G_1 and G_2 be the gems of P_1 and P_2. If a_1 and a_b are nodes of G_1 and G_2, then $T(a_1) = T(a_2)$ if and only if a_1 and a_2 determine a natural isomorphism between G_1 and G_2, which implies $P_1 = P_2$.*

Proof:

If $P_1 = P_2$ and the natural isomorphism between G_1 and G_2 maps a_1 in a_2, it is trivial that $T(a_1) = T(a_2)$.

If $T(a_1) = T(a_2)$, by the lemma 3, $Flag(a) = Flag(b)$. If $Flag(a) = Flag(b) = (F_0, \ldots, F_{n-1})$ then P_1 and P_2 share the facet F_{n-1}. If $T(a) = T(b) = (v_0, \ldots, v_n)$, P_1 and P_2 share the vertex v_n, which is not a face of F_{n-1}. The intersection of two distinct polytopes of a quasi-complex is at most one face, thus P_1 and P_2 are the same polytope. Since $Flag(a_1) = Flag(a_2)$, a_1 and a_2 determine the natural isomorphism between G_1 and G_2. $\square$

Theorem 2 *Let G_1 be the gem of a n-polytope P_1. For each node a of G_1, the vertices of $T_k(a) = (v_0, \ldots, v_k)$ are affinely independent, for $0 \leq k \leq n$.*

Proof: For $0 \leq i < k$, the vertices $v_0, \ldots, v_i$ belong to the same i-face F_i, whereas v_{i+1} does not belong. Since P_1 is convex, v_{i+1} does not belong to the affine hull of F_i. Therefore, if $v_0, \ldots, v_i$ is an affinely independent set of vertices, the set $v_0, \ldots, v_{i+1}$ also is. $\square$

Vertex tuple ordering: Let G be the gem of the border of a polytope P and let S be the set of vertices of P. Let's fix an arbitrary order on S. If a and b are nodes of G such that $T(a) = (v_0, \ldots, v_n)$ and $T(b) = (v'_0, \ldots, v'_n)$, we say that $T(a) < T(b)$ if, for some i between 0 and n, $v_0 = v'_0$, $v_1 = v'_1$,..., $v_{i-1} = v'_{i-1}$ e $v_i < v'_i$. This provides an implicit ordering on the set of G nodes.

Head node: The *head node* of a gem G, denoted by $Head(G)$, is the node with minimum $T(a)$.

Theorem 3 *Let C be a n-complex containing the polytopes P_1 and P_2, and let G_1 and G_2 be the gems of P_1 and P_2. $T(Head(G_1)) = T(Head(G_2))$ if and only if $P_1 = P_2$.*

Proof: If $T(Head(G_1)) = T(Head(G_2))$, by theorem 1, $P_1 = P_2$.

If $P_1 = P_2$, there is a natural isomorphism between G_1 and G_2, hence, the sets of vertex tuples of G_1's nodes and G_2's nodes are identical. Thus, $T(Head(G_1)) = T(Head(G_2))$. $\square$

Our algorithm obtains the head node of the convex hull's gem by identifying recursively the head nodes of the the facets' gems of and obtaining the minimum.

10 Algorithm outline

Our algorithm is presented below in the recursive routine `BuildGem(`S`,`n`)`, where S is the set of input points and n is the dimension of the space.

We assume that S contains at least $n+1$ affinely independent points. Otherwise S must be contained in some k-dimensional subspace V of $\mathbb{R}^n$, with $k < n$; in that case, one can obtain the convex hull by choosing any projection of V onto $\mathbb{R}^k$, and calling `BuildGem(`S'`,`k`)` with the projected set S'.

The `BuildGem` procedure constructs a gem data structure that represents the convex hull P of S, and returns its head node. It also returns a list cointaining the head nodes of the gems of P's facets.

In this routine, C is the $(n-1)$-complex of already determined facets, and Q is the $(n-2)$-quasi-complex $\partial^\star C$. The polytopes of C and Q are represented by their head nodes. We denote by `Hyperplane`$(p_0, p_1, \ldots, p_n; q)$ an operation that provides the coefficients of a hyperplane G containing the affinely independent points $p_0, p_1, \ldots, p_n$ and such that $G(q) > 0$.

`BuildGem(`S`,`n`)`

```
 1. H ← InitialFacet(S)
 2. S' ← (S ∩ H) ↓ R^{n-1}
 3. (f,Q) ← BuildGem(S',n-1)
 4. C ← {f}
 5. HeadNode ← f
 6. While(Q ≠ ∅)
 7.   Get a node r of Q
 8.   (v_0,...,v_{n-1}) ← T_{n-1}(r)
 9.   p ← a point of S not coplanar with T_{n-1}(r)
10.   G ← Hyperplane(v_0,...,v_{n-2},v_{n-1};p)
11.   K ← Hyperplane(v_0,...,v_{n-2},p;v_{n-1})
12.   H ← AdjacentFacet(S,G,K)
13.   S' ← (S ∩ H) ↓ R^{n-1}
14.   (f,Q') ← BuildGem(S',n-1)
15.   Add f to C
16.   For each node r' of Q'
17.     If there is a r'' in Q such that T_{n-2}(r') = T_{n-2}(r''), then
18.       SplicePolytopes(n-2,r',r'')
19.       Remove r'' from Q
20.     Else
21.       Add r' to Q
22.   If (T_{n-1}(f) < T_{n-1}(HeadNode))
23.     HeadNode ← f
24. Return (HeadNode,C)
```

It is also necessary to implement the trivial case of the recursion, which deals with points in $\mathbb{R}^1$. We make now additional comments about some steps of the algorithm.

Steps 2 and 13: Before the recursive call of `BuildGem`$_{n-1}$, the points of S contained on the supporting hyperplane H are projected onto the space $\mathbb{R}^{n-1}$, e.g. by deleting any coordinate whose axis is not parallel to H.

Steps 10 and 11: By the theorem 2, the vertices of $T_{n-1}(r)$ are affinely independent. By the lemma 1, the points $v_0, \ldots, v_{n-1}$ are vertices of an $(n-1)$-polytope F of C, and

the points $v_0, \ldots, v_{n-2}$ are vertices of a facet R of F. So, the hyperplane G contains the $(n-1)$-polytope F, and the intersection of K and G contains the $(n-2)$-polytope R.

Step 17: By the theorems 1 and 3, $T_{n-2}(r') = T_{n-2}(r'')$ if and only if r' and r'' determine the natural isomorphism between $(r', n-2)$ and $(r'', n-2)$. In this case, the ridges of the convex hull represented by $(r', n-2)$ and $(r'', n-2)$ are coincident.

Step 18: Here we attach the $(n-1)$-polytopes represented by $(r', n-1)$ and $(r'', n-1)$ through their coincident facet. This is done by the operation `SplicePolytopes(`$n-2$`,`r'`,`r''`)` using the natural isomorphism between $(r', n-2)$ and $(r'', n-2)$ determined by r' and r''.

Steps 22 and 23: These steps are responsible for finding the head node of the convex hull's gem.

References

[1] Erik Brisson. Representing geometric structures in d dimensions: Topology and order. *Proc. 5th Annual ACM Symp. on Computational Geometry*, pages 218–227, June 1989.

[2] Donald R. Chand and Sham S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17(1):78–86, 1970.

[3] Y.-K. Choi, X. Li, F. Rong, W. Wang, and S. Cameron. Computing the minimum directional distance between two convex polyhedra.

[4] David P. Dobkin and Michel J. Laszlo. Primitives for the manipulations of three-dimensional subdivisions. In *Proc. 3rd ACM Symp. on Comp. Geometry*, pages 86–99. ACM Press, June 1987.

[5] Herbert Edelsbrunner and Ernst P. Mucke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. In *Proc. 4th ACM Symposium on Computational Geometry*, pages 118–133, 1988.

[6] Ioannis Emiris and John Canny. A practical method for the sparse resultant. In *ISSAC '93: Proceedings of the 1993 International Symposium on Symbolic and Algebraic Computation*, pages 183–192, New York, NY, USA, 1993. ACM Press.

[7] Ioannis Z. Emiris and John F. Canny. An efficient approach to removing geometric degeneracies. In *Proc. 8th ACM Symp. on Computational Geometry*, pages 74–82, 1992.

[8] Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulations of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[9] D.-T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *Int. J. Computer Information Science*, 9:219–242, 1980.

[10] Pascal Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. *Proc. 5th Annual ACM Symp. on Computational Geometry*, pages 228–236, June 1989.

[11] M. Lin and D. Manocha. Collision and proximity queries, 2003.

[12] Sóstenes Lins. Graph-encoded maps. *Journal of Combinatory Theory (B)*, 32:171–181, 1982.

[13] Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and ordering large radiosity computations. *Computer Graphics*, 28(Annual Conference Series):443–450, 1994.

[14] C.-K. Yap. Symbolic treatment of geometric degeneracies. *J. Symb. Comput.*, 10(3–4):349–370, 1990.