Contributing vertices-based Minkowski sum computation of convex polyhedra

Hichem Barki, Florence Denis, Florent Dupont

Contributing vertices-based Minkowski sum computation of convex polyhedra

Hichem Barki [a,*] Florence Denis [a] Florent Dupont [a]

[a] *Université de Lyon, CNRS*

Université Lyon 1, LIRIS, UMR5205

43 Bd. du 11 novembre 1918, F-69622 Villeurbanne, France

Abstract

Minkowski sum is an important operation. It is used in many domains such as: computer-aided design, robotics, spatial planning, mathematical morphology, and image processing. We propose a novel algorithm, named the Contributing Vertices-based Minkowski Sum (CVMS) algorithm for the computation of the Minkowski sum of convex polyhedra. The CVMS algorithm allows to easily obtain all the facets of the Minkowski sum polyhedron only by examining the contributing vertices—a concept we introduce in this work, for each input facet. We exploit the concept of contributing vertices to propose the Enhanced and Simplified Slope Diagram-based Minkowski Sum (ESSDMS) algorithm, a slope diagram-based Minkowski sum algorithm sharing some common points with the approach proposed by Wu et al. [1]. The ESSDMS algorithm does not embed input polyhedra on the unit sphere and does not need to perform stereographic projections. Moreover, the use of contributing vertices brings up more simplifications and improves the overall performance. The implementations for the mentioned algorithms are straightforward, use exact number types, produce exact results, and are based on CGAL, the Computational Geometry Algorithms Library. More examples and results of the CVMS algorithm for several convex

polyhedra can be found at `http://liris.cnrs.fr/hichem.barki/mksum/CVMS-convex`

Key words: Minkowski sum, contributing vertices, slope diagram, convex hull, computer-aided design

1 Introduction

Minkowski sum implementation is of a particular interest and used in a variety of domains such as computer-aided design and manufacturing [2], computer animation and morphing [3], morphological image analysis [4,5], similarity measures for convex polyhedra [6], penetration depth computation and dynamic simulation [7], robot motion planning [8], and solid modeling.

The Minkowski sum or addition of two sets $\mathcal{A}$ and $\mathcal{B}$ in a vector space was defined by the German mathematician Herman Minkowski (1864–1909) as a position vector addition of elements of $\mathcal{A}$ and elements of $\mathcal{B}$:

$$\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\} \tag{1}$$

Where + denotes the vector addition of two position vectors a and b coming from the two sets $\mathcal{A}$ and $\mathcal{B}$. The Minkowski sum can also be obtained by the following definition:

$$\mathcal{A} \oplus \mathcal{B} = \bigcup_{a \in \mathcal{A}} \mathcal{B}_a \tag{2}$$

* Corresponding author. Tel.: +(33) 04 72 43 26 51; Fax: +(33) 04 72 43 15 36.
Email addresses: `hichem.barki@liris.cnrs.fr` (Hichem Barki),

`florence.denis@liris.cnrs.fr` (Florence Denis),

`florent.dupont@liris.cnrs.fr` (Florent Dupont).

Where $\cup$ denotes the set union operation and $\mathcal{B}_a$ denotes the set $\mathcal{B}$ translated by a vector a. The second definition states that the Minkowski sum of two sets $\mathcal{A}$ and $\mathcal{B}$ is obtained by sweeping all points of $\mathcal{A}$ by $\mathcal{B}$ and taking the union of all resulting points. The sweep aims at positioning or translating $\mathcal{B}$ such that its origin (the common initial point of all its position vectors) coincides with each point of $\mathcal{A}$ and to take the union of the resulting sets, as depicted in Fig. 1.

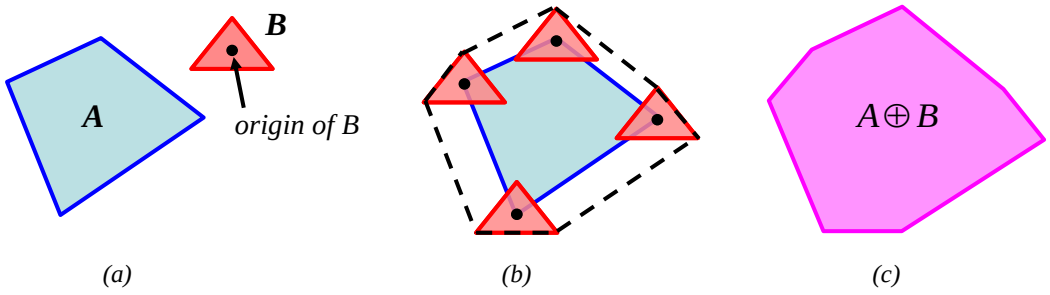


Fig. 1. Minkowski sum as a sweep of two sets. (a) Two polygons $\mathcal{A}$ and $\mathcal{B}$. (b) $\mathcal{B}$ is positioned on each point of $\mathcal{A}$. (c) The Minkowski sum $\mathcal{A} \oplus \mathcal{B}$ is the union of the resulting translations of $\mathcal{B}$ on all points of $\mathcal{A}$.

Our goal is to compute the Minkowski sum polyhedron $S = A \oplus B$ where A and B are two convex polyhedra in $\mathbb{R}^3$. The polyhedra A and B are the respective boundary representations of the sets $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^3$ ($A = \partial\mathcal{A}$ and $B = \partial\mathcal{B}$). It is clear that $\mathcal{A}$ and $\mathcal{B}$ are completely defined by their boundaries A and B. Moreover, the Minkowski sum $\mathcal{A} \oplus \mathcal{B}$ is also completely defined by the polyhedron $S = A \oplus B$ (its corresponding boundary representation).

A convex polyhedron P in $\mathbb{R}^3$ can be seen as the intersection of a finite number of closed half-spaces $P = \bigcap_{i=1,\ldots,n} H_i^-$. Where H_i^- is a closed half-space bounded by the plane H_i having a normal vector u_i , the half-space H_i^- is called the negative side of the plane H_i (see Fig. 2 for an example in 2D). The boundary of a polyhedron can be partitioned into two-dimensional faces

(facets), one-dimensional faces (edges), and zero-dimensional facets (vertices). The supporting plane of a particular facet of a polyhedron A is the plane where the facet lies. As an example, for the convex polygon P depicted in Fig. 2.b, the supporting line (the supporting plane becomes a supporting line in 2D) of the one-dimensional face (edge) e_3 is H_3. Further details on support function representation of convex polyhedra and supporting planes can be found in [9].

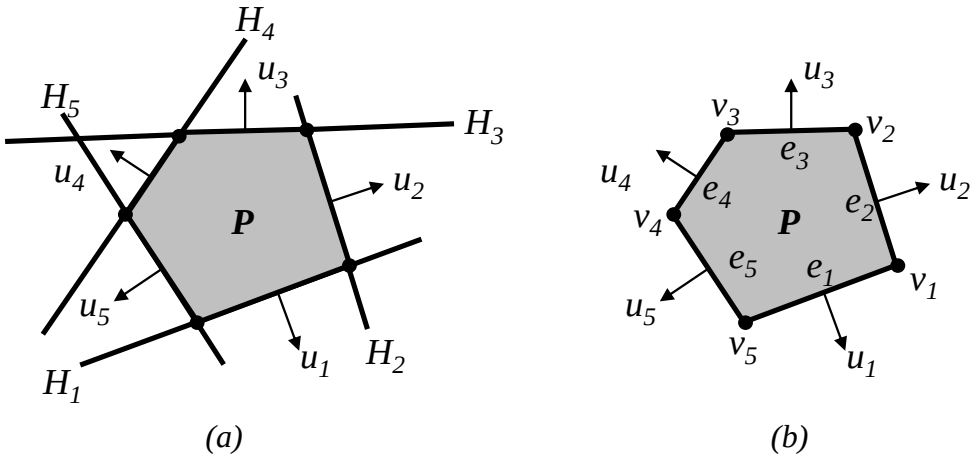


Fig. 2. The intersection of closed half-planes. (a) A finite number of closed half–planes $H_1^-, \ldots, H_5^-$ defined by $H_1, \ldots, H_5$. (b) The boundary of P is partitioned into one-dimensional faces (edges) $e_1, \ldots, e_5$ and zero-dimensional faces (vertices) $v_1, \ldots, v_5$.

The rest of this paper is based on the definitions we gave above and is organized as follows:

- In section 2, we review existent techniques for the computation of Minkowski sum of polyhedra.
- In section 3, we give an overview of our work and introduce key concepts we use throughout all the paper, such as the concept of contributing vertices, the relation between contributing vertices and the sweep process, the translated facets, the corner facets, and the edge facets.

- In section 4, we present in detail our new algorithm, based on the concept of contributing vertices, for the computation of the Minkowski sum of convex polyhedra. We call it Contributing Vertices-based Minkowski Sum (CVMS) algorithm.
- In section 5 we explain how the contributing vertices concept can help improve slope diagram-based methods and present our Enhanced and Simplified Slope Diagram-based Minkowski Sum (ESSDMS) algorithm.
- Finally, we present results, performance study, and comparisons between our two algorithms and other approaches found in the literature.

2 Previous work

The large number of applications needing Minkowski sum has inspired researchers to propose and develop various algorithms. The most common approach used for polyhedra is based on convex hulls. For two convex polyhedra A and B, it performs vector addition of all points of A and B and computes the convex hull of the resulting point set giving us the Minkowski sum polyhedron $A \oplus B$. Convex hulls are computed by several algorithms such as: the Quickhull algorithm [10], the incremental algorithm [11,12], the gift-wrapping algorithm [13], the divide and conquer algorithm [14], and Graham's algorithm [15]. A summary of these algorithms can be found in [16].

For two non-convex polyhedra A and B, the computation of the Minkowski sum polyhedron $A \oplus B$ is done by decomposing each non-convex polyhedron into convex pieces, computing the the pairwise Minkowski sums of all possible pairs of convex pieces from A and B, and performing the union of the pairwise Minkowski sums. The computation of the Minkowski sum of a pair of convex

pieces (or polyhedra) is an important step in this approach.

For readers interested in the Minkowski sum of non-convex polyhedra, we give some references about convex decomposition and union steps. Convex decomposition aims at decomposing a non-convex polyhedron into convex pieces. The smaller the number of pieces, the better the decomposition. The optimal decomposition of a non-convex polyhedron into convex pieces is known to be NP-hard. More than twenty years ago, Chazelle [17] proposed an optimal decomposition algorithm which generates $O(r^2)$ convex pieces in $O(nr^3)$ time, where r denotes the number of reflex edges and n denotes the number of polyhedron facets. Nevertheless, no practical or robust implementation has been found in the literature for Chazelle's optimal algorithm. An experimental study of convex decomposition strategies can be found in [18].

The union step is the most time consuming step when computing the Minkowski sum of non-convex polyhedra. It can have $O(n^6)$ time complexity for non-convex polyhedra, where n is the number of polyhedra facets [19]. Moreover, there is no robust implementation for the union computation of convex polyhedra that handles all degeneracies [20].

While the combinatorial complexity of the convex hull approach is $O(mn)$ for convex polyhedra with m and n features, it can have $O(m^3n^3)$ worst-case combinatorial complexity for non convex polyhedra [20,21].

Recently, Hachenberger implemented a data structure in CGAL [22] based on Nef polyhedra theory [23]. He used this Nef polyhedra implementation [24,25] for the decomposition of polyhedra into convex parts and the union computation of pairwise Minkowski sums [26]. Unfortunately, this approach is time consuming because it is based on the decomposition of polyhedra into

convex pieces.

Due to non-robustness and expense of the union operation, researchers tried to bypass it by using other representations or computing only approximations. Varadhan and Manocha [27] decomposed the polyhedra into convex pieces, computed the pairwise Minkowski sums of convex pieces (by means of convex hull algorithms), approximated the union of the pairwise Minkowski sums using an adaptively subdivided voxel grid, computed signed distance on the grid points and used isosurface extraction from the distance field [28]. They guarantee that their approximation has the same topology as the exact Minkowski sum and provide a two-sided Hausdorff distance bound on the approximation. Recently, Lien [29] used a point-based representation instead of the widely used mesh-based representation to compute the Minkowski sum of polyhedra. He uniformly sampled two point sets from the boundaries of two polyhedra, constructed a point set by adding all points from the two point sets already generated, and used three filters (a collision detection, normal, and octree filter) to discard interior points that are not on the boundary of the sum polyhedron. He demonstrated that the point-based representation provides the same functionality as the mesh-based representation by using it in several applications such as motion planing.

Guibas et al. [30,31] presented a kinetic framework in two dimensions. An accomplishment of this framework was to define the operation of convolution on planar tracings in 2D. Basch et al. [32] extended the convolution operation to polyhedral tracings in 3D and used it to generate a superset of the Minkowski sum. They extracted the exact boundary of the Minkowski sum by using arrangement computations.

Ghosh [33] proposed a unified computational framework to compute the Minkowski sum of polygons and polyhedra (in 2D and 3D domains). The polyhedra are represented in a dual space called the slope diagram. Computing the intersections between the two merged or overlaid slope diagrams gives the sum polyhedron. Although Ghosh stated that slope diagrams can be used for both convex and non-convex polyhedra, the existent implementations of slope diagram-based Minkowski sum computation are only devoted to convex polyhedra.

Some researchers proposed other variants of slope diagram representation. Bekker and Roerdink [34] used a slightly different slope diagram representation that works with edges instead of facets, so their 2D representation of the slope diagram reduces the problem's dimension. Wu et al. [1] presented some improvements to the slope diagram by merging the two diagrams without an explicit embedding on the unit sphere. They also avoided the use of stereographic projections (needed in the original slope diagram algorithm proposed by Ghosh).

An interesting algorithm is that proposed and implemented by Fogel and Halperin [35]. The authors represented the convex polyhedra in a dual space they called the Cubical Gaussian Map (CGM) and implemented their algorithm on the base of the arrangement package of CGAL. The cubical Gaussian map is a Gaussian map embedded on a unit cube instead of the unit sphere. The Minkowski sum of two convex polyhedra A and B is obtained by computing $CGM(A)$ and $CGM(B)$, the respective cubical Gaussian maps of A and B, overlaying or merging $CGM(A)$ and $CGM(B)$, and building the Minkowski sum polyhedron $S = A \oplus B$ from the resulting overlay. The overlay process has a time complexity of $O(f_S log(f_A + f_B))$, where f_A, f_B, and f_S are

the respective facets number of the polyhedra A, B, and S.

3 Overview of our work

In this work, we are interested in computing the Minkowski sum of convex polyhedra. We present a novel algorithm based on the concept of contributing vertices associated to every facet of the two polyhedra we are working with, hence its name : **Contributing Vertices-based Minkowski Sum (CVMS) algorithm**.

3.1 Introduction to the CVMS algorithm

The CVMS algorithm is inspired from the definition given in equation 2. This equation states that the Minkowski sum set $\mathcal{A} \oplus \mathcal{B}$ is obtained by sweeping all points of $\mathcal{A}$ by $\mathcal{B}$ and taking the union of all resulting points. Since we are working with polyhedra, it is clear that it is sufficient to only sweep the boundary of A by B and discard the interior points of A and B in order to have the Minkowski sum polyhedron $S = A \oplus B$. Thus, we are only interested in the boundary of the resulting union.

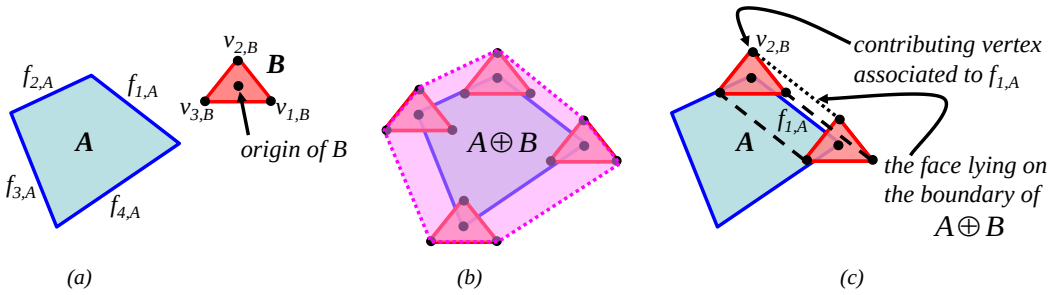
The boundary of A is composed of facets, vertices, and edges. So, in order to obtain the Minkowski sum polyhedron S, we must sweep all facets of A by B, sweep all vertices of A by B (or place B on all vertices of A), and sweep all edges of A by B. Observe that because the Minkowski sum is commutative, sweeping the vertices of A by B (or positioning B on vertices of A) is equivalent to sweeping the facets of B by A. Therefore, in the CVMS algorithm, we need to sweep all facets of A by B, sweep all facets of B by A, sweep all edges of

A by B, and retain only facets that lie on the boundary of the union of these sweeps (facets lying on the boundary of the Minkowski sum polyhedron S).

3.2 From the sweep to the definition of contributing vertices

To give an idea of what is a contributing vertex and how it relates to the sweep operation, we consider two convex polygons A and B depicted in Fig. 3.a, their Minkowski sum $A \oplus B$ is also shown in Fig. 3.b. We take a 2D example for simplicity purposes and without loss of generality. Let's consider the face (edge) $f_{1,A}$ of A and sweep it by B. Sweeping $f_{1,A}$ by B resumes to parallel-translating B such that its origin passes through all points of $f_{1,A}$. During this sweep, all vertices of B (i.e. $v_{1,B}$, $v_{2,B}$, and $v_{3,B}$) are also moving in lines parallel to the supporting line of $f_{1,A}$. The displacement of these vertices produces faces (edges) having supporting lines parallel to that of $f_{1,A}$ (faces that are drawn as dashed and dotted lines in Fig. 3.c). But only the facet generated by the vertex $v_{2,B}$ (the dotted line in Fig. 3.c) which is at maximal distance away from the supporting line of $f_{1,A}$ is considered because it lies on the boundary of $A \oplus B$. The other produced faces (drawn as dashed lines in Fig. 3.c) are simply discarded because they do not lie on the boundary of $A \oplus B$. This particular vertex $v_{2,B}$ which generated this facet of the Minkowski sum polygon $A \oplus B$ is called the **"contributing vertex"** associated to the face $f_{1,A}$.

For two convex polyhedra A and B, the same logic applies. So, sweeping a particular facet $f_{i,A}$ of A by B is done by parallel-translating B such that its origin (the common initial point of all its position vectors) passes through all points of $f_{i,A}$. This sweep implies that all vertices of B are also parallel-



(a) (b) (c)

Fig. 3. (a) Two convex polygons A and B. (b) The Minkowski sum $A \oplus B$ depicted as a sweep. (c) The contributing vertex $v_{2,B}$ associated to the face $f_{1,A}$. The dotted line is the face that lies on the boundary of $A \oplus B$. The two other faces (dashed lines) are discarded.

translated in the same manner, they are moving in planes parallel to the supporting plane of $f_{i,A}$ and generating facets within supporting planes parallel to that of $f_{i,A}$. But only the facet generated by the vertex $v_{k,B}$ of B which is at maximal distance away from the supporting plane of $f_{i,A}$ is taken into account because it lies on the boundary of the sum polyhedron. The other facets generated by the displacement of all other vertices of B are discarded since they lie in the interior of the sum polyhedron. This particular vertex $v_{k,B}$ which generated this facet of the sum polyhedron $A \oplus B$ is called the **"contributing vertex"** associated to the facet $f_{i,A}$.

3.3 Computation of the Minkowski sum polyhedron from contributing vertices

To construct the Minkowski sum polyhedron, we must perform three sweep steps mentioned above. The first step is to sweep all facets of A by B. From what precedes, it is clear that the computation of the contributing vertices associated to the facets of A is a mean to obtain the facets of $S = A \oplus B$ that result from sweeping the facets of A by B, these facets of S are called **"translated facets"**.

The second step is to sweep all the facets of B by A. Therefore, computing the contributing vertices associated to the facets of B is a mean to obtain the facets of $S = A \oplus B$ that result from sweeping the facets of B by A. these facets are denoted **"corner facets"** of S.

The last step is to sweep the edges of A by B. For a particular edge $e_{j,A}$ of A, this operation implies the construction of several facets that are the Minkowski sum of the edge $e_{j,A}$ and all the edges of B. But not all these facets will lie necessarily on the boundary of S. Therefore, some of these facets are discarded. To determine which of these facets are retained, we use two criteria: a visibility criterion and normal orientation criterion as explained in section 4.3. The retained facets are denoted **"edge facets"**. Note that only edges of A or B incident to facets having distinct contributing vertices will be considered in this step. A proof of this property is given in section 4.

These three steps of sweep enable us to construct the Minkowski sum polyhedron S. We show in section 4 how to compute the contributing vertices associated to the facets of A and B. This resumes to the computation of distances from vertices to supporting planes. We also show that the sum polyhedron S is composed of three (already mentioned) types of facets: "translated facets" obtained from the facets of A and their associated contributing vertices, "corner facets" obtained from the facets of B and their associated contributing vertices (commutativity property), and "edge facets" obtained from the Minkowski sum of two non-parallel edges one from A and the other from B, having incident facets with no common contributing vertices. An outline of the CVMS algorithm is presented in algorithm 1. Further details can be found found in section 4.

Algorithm 1 An outline of the The CVMS algorithm for convex polyhedra

Require: two convex polyhedra A and B

Ensure: the Minkowski sum polyhedron $S = A \oplus B$

for each facet $f_{i,A}$ of A **do**
  compute its associated contributing vertices (w.r.t. B)
  deduce the translated facet corresponding to it
end for
for each facet $f_{i,B}$ of B **do**
  compute its associated contributing vertices (w.r.t. A)
  deduce the corner facet corresponding to it
end for
for each edge $e_{j,A}$ of A incident to facets having distinct contributing vertices **do**
  for each edge $e_{k,B}$ of B incident to facets having distinct contributing vertices **do**
    if $e_{j,A}$ and $e_{k,B}$ satisfy both visibility and normal orientation criteria **then**
      construct the edge facet $e_{j,A} \oplus e_{k,B}$
    end if
  end for
end for

3.4 Contributing vertices applied to slope diagram-based algorithms

We also show how the contributing vertices concept benefits slope diagram algorithms. So, we present the **Enhanced and Simplified Slope Diagram-based Minkowski sum (ESSDMS) algorithm**, which is somewhat similar to the improved slope diagram algorithm (improved 3D-MSSD algorithm) pre-

sented in the work of Wu et al. [1]. This similarity is due to the fact that we do not embed input polyhedra on the unit sphere and do not use stereographic projections. The merging of slope diagrams and the handling of the intersections are made possible by simple geometric operations on the outer normal orientation information extracted from the polyhedra. The contributing vertices concept allows to further simplify the algorithm and to gain considerable performance by eliminating point in polygon queries and reducing edge arcs intersection queries. The computation of the contributing vertices does not introduce additional performance overhead since they are deduced directly from the intersection configuration. Details on our ESSDMS algorithm can be found in section 5.

4 The CVMS algorithm for convex polyhedra

In the rest of this paper, we will not distinguish between a vector v and a point v. We will talk about a vertex v and simply refer, in the equations, to the vector pointing from the coordinates origin o to the point v as v. The distinction between a point and the corresponding vector is clear from the context.

First, let's give some notation and definitions required for the understanding of the following notions.

4.1 Notation

We consider two convex, closed and two-manifold polyhedra A and B. A is composed of f_A facets, e_A edges, and v_A vertices. Similarly, B is composed

of f_B facets, e_B edges, and v_B vertices. The Minkowski sum of A and B is denoted S.

4.2 Prerequisites and definitions

First, let's define the concept of contributing vertices since it is needed in the rest of definitions.

Definition 1: The **contributing vertex** $v_{k,B}$ of a particular facet $f_{i,A}$ with an outer normal $n_{i,A}$ is the vertex of B which is at maximal distance away from the supporting plane of $f_{i,A}$. Formally, the contributing vertex $v_{k,B}$ of a particular facet $f_{i,A}$ satisfies:

$$\langle v_{k,B}, n_{i,A} \rangle = \max_{\mathbb{R}} \langle v_{l,B}, n_{i,A} \rangle \ \forall l = 1, 2, ..., v_B \tag{3}$$

Where $\langle ., . \rangle$ denotes the scalar product. An illustration of the contributing vertex concept in 2D is presented in Fig. 4.

The contributing vertex $v_{k,A}$ of a particular facet $f_{i,B}$ with an outer normal $n_{i,B}$ is defined in the same manner by putting A in place of B and B in place of A in equation 3.

For some facets of A and B, we can find many contributing vertices due to the fact that these contributing vertices are at the same (maximal) distance away from the supporting planes of the considered facets. This case is treated normally and does not cause any degenerate behavior.

Since the boundary of polyhedra is composed of facets, edges, and vertices, sweeping the boundary of A by B is equivalent to sweeping the facets of A by B, sweeping the edges of A by B, and sweeping the vertices of A by B (or positioning B on vertices of A). Remember that it is sufficient to only sweep

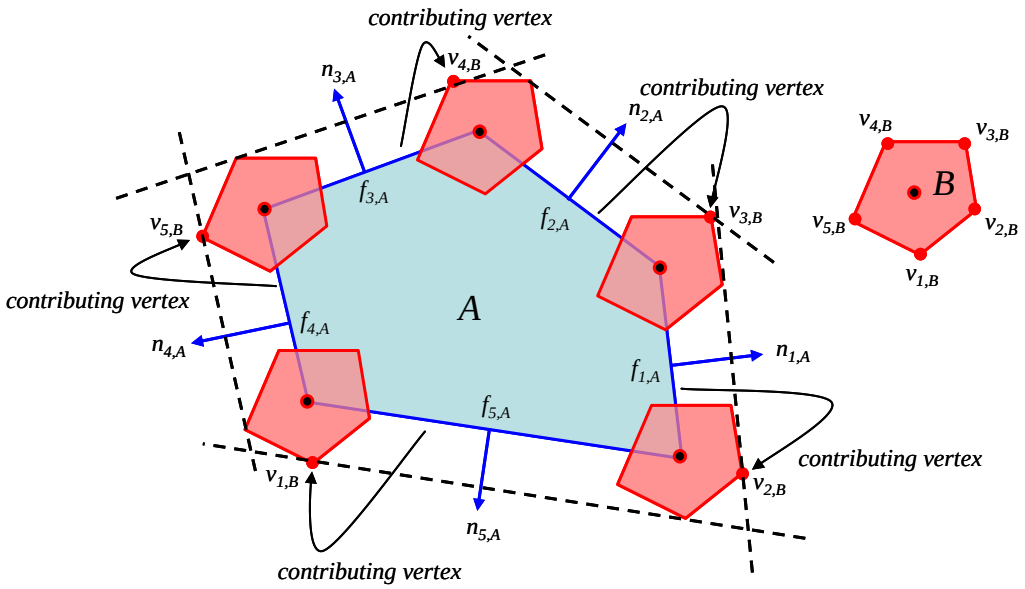


Fig. 4. Illustration of the contributing vertices concept for the facets of a convex polygon A to be added to the convex polygon B.

the boundary of A by B and discard the interior points of A and B.

(1) Sweeping all the facets of A by B: when a particular facet $f_{i,A}$ is swept by B (see Fig. 5.a), this results in the facet of the sum polyhedron S having a supporting plane parallel to that of $f_{i,A}$ since it is generated by the contributing vertex associated to $f_{i,A}$ (i.e. the vertex $v_{k,B}$ which is at maximal distance away from the supporting plane of $f_{i,A}$). So, the sweeping of all f_A facets of A results in the f_A translated facets with supporting planes parallel to those of the facets of A (see Fig. 6.d).

(2) Positioning B on all vertices of A: this means positioning B so that its origin coincides with each vertex $v_{k,A}$. It is clear that only some facets of B will lie on the boundary of the sum polyhedron S when B is positioned on a particular vertex of A. The remaining facets of B are discarded because they lie inside S. To find the facets of B that are part of S, we exploit the commutativity property of the Minkowski sum operation, i.e.

$A \oplus B = B \oplus A$. If we sweep all facets of B by A, we observe that the generated facets are the same as the facets we want to find by positioning B on each vertex $v_{k,A}$. So, positioning B on all v_A vertices results in the f_B corner facets with supporting planes parallel to those of the facets of B (see Fig. 6.e).

(3) Sweeping all edges of A by B: when a particular edge $e_{j,A}$ is swept by B, i.e. the origin of B is translated so it passes through all points of $e_{j,A}$ (see Fig. 5.b). This produces facets of S that are the result of the Minkowski sum of the edge $e_{j,A}$ and all edges of B. But not all these generated facets lie on the boundary of the sum polyhedron S. So, S is composed of at most $e_A e_B$ edge facets (see Fig. 6.f). The determination of edge facets is explained in section 4.3.

Proposition 1: The Minkowski sum polyhedron S of two convex, closed, and two-manifold polyhedra A and B is composed exactly of three types of facets:

(1) f_A facets with supporting planes parallel to those of the facets of A, these facets are named the **"translated facets"** of S;

(2) f_B facets with supporting planes parallel to those of the facets of B, these facets are named the **"corner facets"** of S;

(3) At most $e_A e_B$ facets that result from the Minkowski sum of some pairs of non-parallel edges of A and B, these facets are named the **"edge facets"** of S.

Proof: Let's sweep all f_A facets of A by B. For a particular facet $f_{i,A}$, $i = 1, \ldots, f_A$ we keep only the facet that is the result of translating $f_{i,A}$ according to its contributing vertex or vertices. So, there is a one-to-one mapping between the facets of A and the facets of S with supporting planes parallel

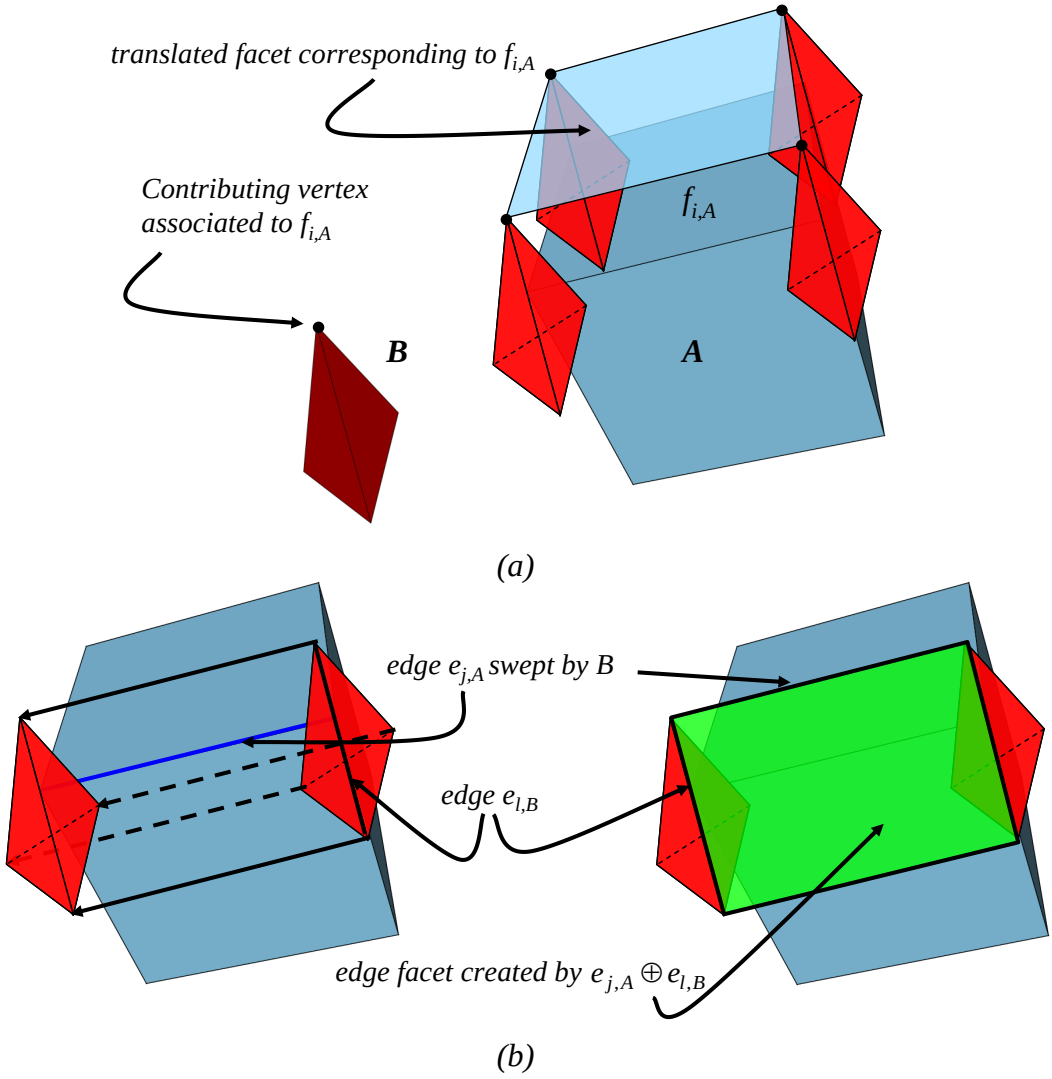


Fig. 5. (a) The creation of a translated facet corresponding to a particular facet $f_{i,A}$ and its associated contributing vertex (the corner facets are determined in the same manner). (b) Sweeping an edge $e_{j,A}$ by B and creation of an edge facet $e_{j,A} \oplus e_{l,B}$.

to those of the facets of A and that we called translated facets. Therefore, the sum polyhedron S contains f_A translated facets. In a similar manner, if we sweep all f_B facets of B by A, we will obtain exactly f_B corner facets, i.e. facets of S with supporting planes parallel to those of the facets of B. Finally, it remains to sweep the edges of A by B. The result of this sweep are facets that are the Minkowski sum of the edges of A and the edges of

B. But since we are working with polyhedra (boundary representation), we will not keep all these $e_A e_B$ facets, only those lying on the boundary of the Minkowski sum polyhedron S are retained. Therefore, the Minkowski sum polyhedron is composed of at most $e_A e_B$ edge facets.

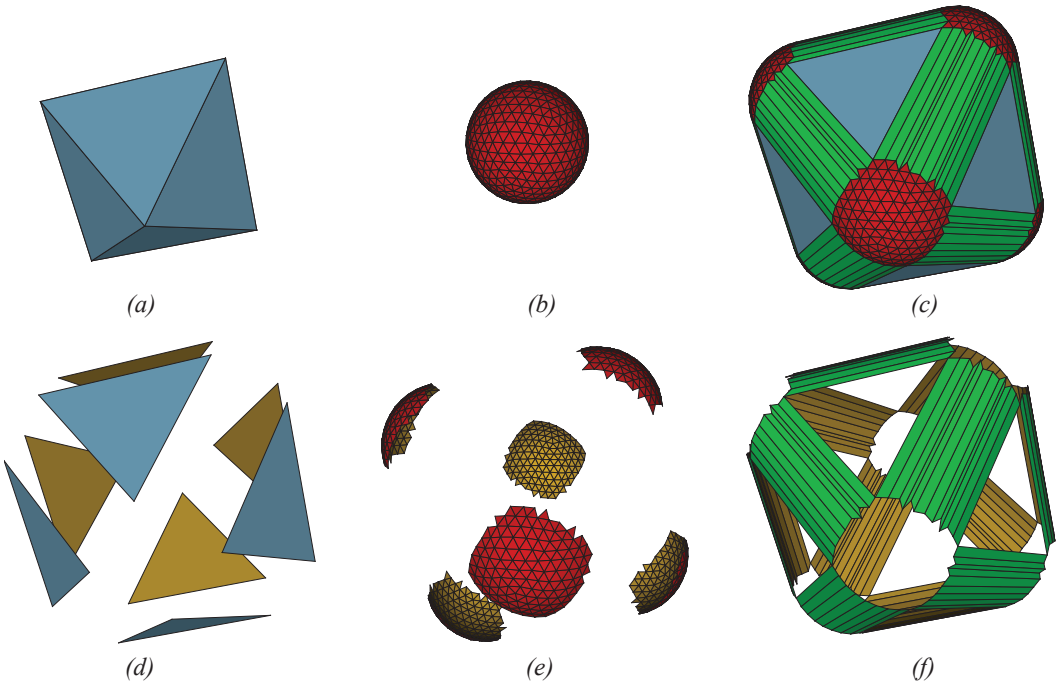


Fig. 6. (a) Polyhedron A. (b) Polyhedron B. (c) Sum polyhedron S (d) Translated facets. (e) Corner facets. (f) Edge facets.

From the definitions of translated and corner facets, it is clear that the translated facets of $A \oplus B$ are corner facets of $B \oplus A$ and vice-versa. Furthermore, since the Minkowski sum of two edges, one coming from A and the other from B is commutative: $e_A \oplus e_B = e_B \oplus e_A$, it follows that the result of the CVMS algorithm is commutative as well as the Minkowski sum operation.

As said previously, some facets of A and B can be associated to many contributing vertices due to the fact that these contributing vertices are at the same (maximal) distance away from the supporting planes of the considered facets. If a facet of A has two contributing vertices, the corresponding trans-

lated facet is the result of a planar Minkowski sum of that facet and the edge of B incident to the two contributing vertices belonging to a supporting line parallel to the supporting plane of the considered facet of A. If a facet of A has more than two contributing vertices, the corresponding translated facet is the result of a planar Minkowski sum of that facet and the facet of B incident to the contributing vertices and belonging to a supporting plane parallel to that of the considered facet of A. The same rules apply to facets of B having two or more contributing vertices.

Here are two important properties used by the CVMS algorithm:

Property 1: If two adjacent facets $f_{i,A}$ and $f_{j,A}$ have at least one common contributing vertex, the edge $e_{k,A}$ shared by these adjacent facets will never contribute in any edge facets construction and can be simply ignored. This property is also applicable to any two adjacent facets $f_{m,B}$ and $f_{n,B}$ having at least a common contributing vertex.

Proof: Let's consider any edge $e_{k,A}$ of A incident to two facets $f_{i,A}$ and $f_{j,A}$ having at least one common contributing vertex. The facets $f_{i,A}$ and $f_{j,A}$ share the vertices $v_{m,A}$ and $v_{n,A}$ incident to the edge $e_{k,A}$. Since $f_{i,A}$ and $f_{j,A}$ have a common contributing vertex that we denote $v_{p,B}$, constructing the translated facets $f_{i,S}$ and $f_{j,S}$ corresponding to $f_{i,A}$ and $f_{j,A}$ respectively involves the translation of $f_{i,A}$ and $f_{j,A}$ by a vector starting from coordinates origin o and ending at $v_{p,B}$. This translation implies that the two vertices $v_{m,A}$ and $v_{n,A}$ incident to the edge $e_{k,A}$ and shared by the facets $f_{i,A}$ and $f_{j,A}$ are also translated to the positions $v_{m,A} + v_{p,B}$ and $v_{n,A} + v_{p,B}$. Therefore, the edge $e_{k,A}$ will be incident to $f_{i,S}$ and $f_{j,S}$ and will never contribute in any edge facet construction.

In contrast, if another edge $e_{k,A}$ is incident to facets $f_{i,A}$ and $f_{j,A}$ having

distinct common contributing vertices $v_{p,B}$ and $v_{q,B}$ respectively, it is clear that the construction of the translated facets $f_{i,S}$ and $f_{j,S}$ corresponding to $f_{i,A}$ and $f_{j,A}$ involves $v_{p,B}$ and $v_{q,B}$ respectively. Therefore, $f_{i,A}$ is translated by a vector starting from coordinates origin o and ending at $v_{p,B}$ and $f_{j,A}$ is translated by a vector starting from o and ending at $v_{q,B}$. Thus, the vertices $v_{m,A}$ and $v_{n,A}$ incident to the edge $e_{k,A}$ are translated to the positions $v_{m,A}+v_{p,B}$ and $v_{n,A}+v_{p,B}$ when computing $f_{i,S}$ and to the positions $v_{m,A}+v_{q,B}$ and $v_{n,A}+v_{q,B}$ when computing $f_{j,S}$. Therefore, the translated facets $f_{i,S}$ and $f_{j,S}$ are not incident to the same edge of polyhedron S (they are disjoint) and the edge $e_{k,A}$ creates one or more edge facets.

The proof of property 1 can be done in a similar manner for any edge $e_{k,B}$ of B.

In Fig. 7, a simple example illustrates property 1. The edge $e_{1,A}$ in Fig. 7.a is incident to the facets $f_{1,A}$ and $f_{2,A}$. The vertices $v_{1,A}$ and $v_{2,A}$ are those vertices incident to $e_{1,A}$. The facets $f_{1,A}$ and $f_{2,A}$ have a common contributing vertex $v_{1,B}$ (see Fig. 7.b). Therefore, the translated facets $f_{1,S}$ and $f_{2,S}$ corresponding to $f_{1,A}$ and $f_{2,A}$ are incident to the same edge of S delimited by two vertices with positions $v_{1,A}+v_{1,B}$ and $v_{2,A}+v_{1,B}$ (see Fig. 7.c). In contrast, the edges $e_{2,A}$ and $e_{3,A}$ in Fig. 7.a are incident to facets having distinct contributing vertices, so they contribute to the edge facets creation (Fig. 7.c).

Property 2: A facet of each polyhedron A or B contributes only once in the Minkowski sum polyhedron S; this has been proved in earlier works [3,33,34]. This property allows to reduce progressively the number of facets considered in the subsequent steps of the algorithm. For example, if a facet contributed to the creation of a translation facet of S, it can be omitted

from the corner facets determination step.

The two mentioned properties enable a significant gain of performance by reducing the number of facets and edges to be considered when computing translated, corner, and edge facets.

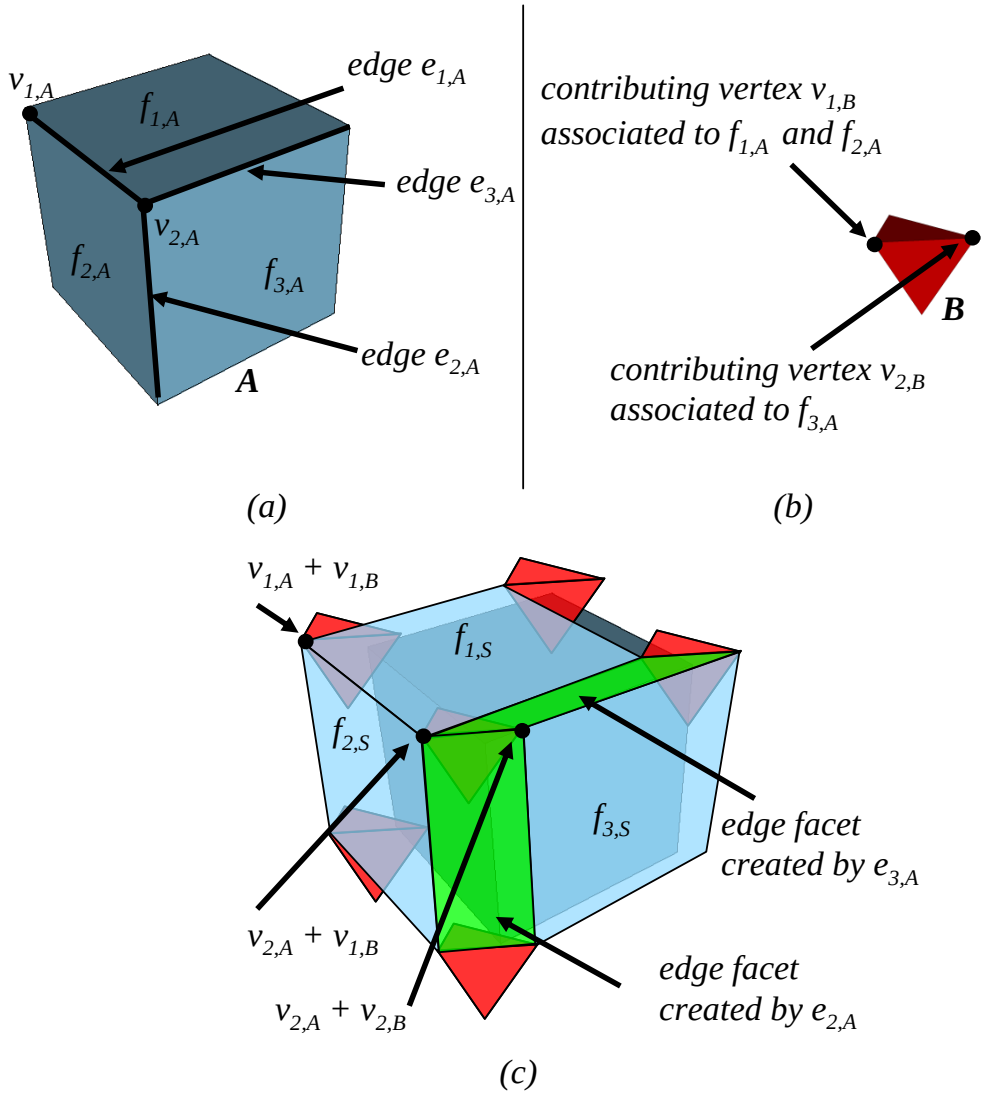


Fig. 7. (a) Some edges $e_{1,A}$, $e_{2,A}$, and $e_{3,A}$ of A. (b) Contributing vertices associated to $f_{1,A}$, $f_{2,A}$, and $f_{3,A}$. (c) Edge facets created by $e_{2,A}$, and $e_{3,A}$. The edge $e_{1,A}$ will not produce edge facets since incident facets $f_{1,A}$ and $f_{2,A}$ have the same contributing vertex.

4.3 Steps of the CVMS algorithm

The CVMS algorithm begins by determining the translated and corner facets of S by computing the contributing vertices associated to all facets of A and B respectively. Then, it determines edge facets. Property 1 is used in the translated and corner facets determination steps and property 2 is used in the edge facets determination step. Given two convex and closed two-manifold polyhedra A and B, the CVMS algorithm performs as follows:

(1) **Translated facets determination:** for each facet $f_{i,A}$:

- Find its associated contributing vertices $v_{k,B}$ (see equation 3).
- If there are more than two contributing vertices, the facet $f_{j,B}$ incident to the contributing vertices associated to $f_{i,A}$ lies on a supporting plane parallel to that of $f_{i,A}$. The corresponding translated facet is the result of the planar Minkowski sum of $f_{i,A}$ and $f_{j,B}$ ($f_{i,A} \oplus f_{j,B}$). The translated facet has the same outer normal orientation as $f_{i,A}$ and $f_{j,B}$. The facet $f_{j,B}$ will be ignored in the corner facets determination step since it contributes once.
- If there are exactly two contributing vertices, the edge $e_{j,B}$ incident to the two contributing vertices associated to $f_{i,A}$ lies on a supporting line parallel to the supporting plane of $f_{i,A}$. The resulting translated facet is computed by a planar Minkowski sum of $f_{i,A}$ and $e_{j,B}$ ($f_{i,A} \oplus e_{j,B}$) and has the same outer normal orientation as the facet $f_{i,A}$.
- If there is only one contributing vertex $v_{k,B}$, the corresponding translated facet results from translating the facet $f_{i,A}$ by a position vector pointing from coordinates origin o to $v_{k,B}$ and has the same outer normal orientation as the facet $f_{i,A}$.

(2) **Corner facets determination:** for each facet $f_{i,B}$ that have not yet contributed in the previous step:

- Find its associated contributing vertices $v_{k,A}$. Note that there is no case where a facet $f_{i,B}$ has more than two contributing vertices since such a facet has contributed once in the translated facets determination step.
- If there are exactly two contributing vertices, the edge $e_{j,A}$ incident to the two contributing vertices associated to $f_{i,B}$ lies on a supporting line parallel to the supporting plane of $f_{i,B}$. The resulting corner facet is computed by a planar Minkowski sum of $f_{i,B}$ and $e_{j,A}$ ($f_{i,B} \oplus e_{j,A}$) and has the same outer normal orientation as the facet $f_{i,B}$.
- If there is only one contributing vertex $v_{k,A}$, the corresponding corner facet results from translating the facet $f_{i,B}$ by a position vector pointing from coordinates origin o to $v_{k,A}$ and has the same outer normal orientation as the facet $f_{i,B}$.

(3) **Edge facets determination:** we use two criteria for this purpose: the visibility of facets of B with respect to a visibility direction (defined by an edge $e_{i,A}$; see Fig. 8.a) allows to find horizon edges of B. Horizon edges are edges that separate invisible facets from other facets B with respect to the considered visibility direction. By other facets of B, we mean facets that are either visible or having supporting planes parallel to the visibility direction $e_{i,A}$ (see Fig. 8.b). All these horizon edges are candidates to construct an edge facet of S, but only those verifying the second criterion of normal orientation will do (see Fig. 8.c). Thus, for each edge $e_{i,A}$ incident to facets having distinct contributing vertices:

- Determine the visibility of all facets of B. A facet $f_{j,B}$ with an outer normal

$n_{j,B}$ is invisible with respect to the visibility direction $e_{i,A}$ if and only if:

$$\langle e_{i,A}, n_{j,B} \rangle > 0 \tag{4}$$

The facets of B that do not satisfy the above inequality are either visible or lie on supporting planes parallel to the visibility direction. The horizon edges are frontier edges between invisible facets and other facets of B with respect to the visibility direction $e_{i,A}$ (see Fig. 8.b).

- For each horizon edge $e_{k,B}$ incident to facets having distinct contributing vertices:

  Add an edge facet $e_{i,A} \oplus e_{k,B}$ to the sum polyhedron S if its outer normal orientation $n_{i,k} = e_{i,A} \times e_{k,B}$ lies between the two outer normal orientations $n_{1,A}$ and $n_{2,A}$ of the facets $f_{1,A}$ and $f_{2,A}$ incident to the edge $e_{i,A}$ ($n_{2,A}$ follows $n_{1,A}$ in a counter-clockwise order; see Fig. 8.c). Therefore, an edge facet $e_{i,A} \oplus e_{k,B}$ is added to the sum polyhedron S if and only if:

$$\langle n_{1,A} \times n_{i,k}, e_{i,A} \rangle < 0 \; and \; \langle n_{i,k} \times n_{2,A}, e_{i,A} \rangle < 0 \tag{5}$$

At this point, we have constructed the sum polyhedron S. We note that we have not talked about the origin of the polyhedron B, but it is easy to consider an origin point c that is different from the coordinates origin o by translating the sum polyhedron S by a vector beginning from c and ending at o.

5 Slope diagram-based Minkowski sum

In order to show how the concept of contributing vertices is a benefit to the construction of slope diagrams, we propose a novel algorithm named the En-

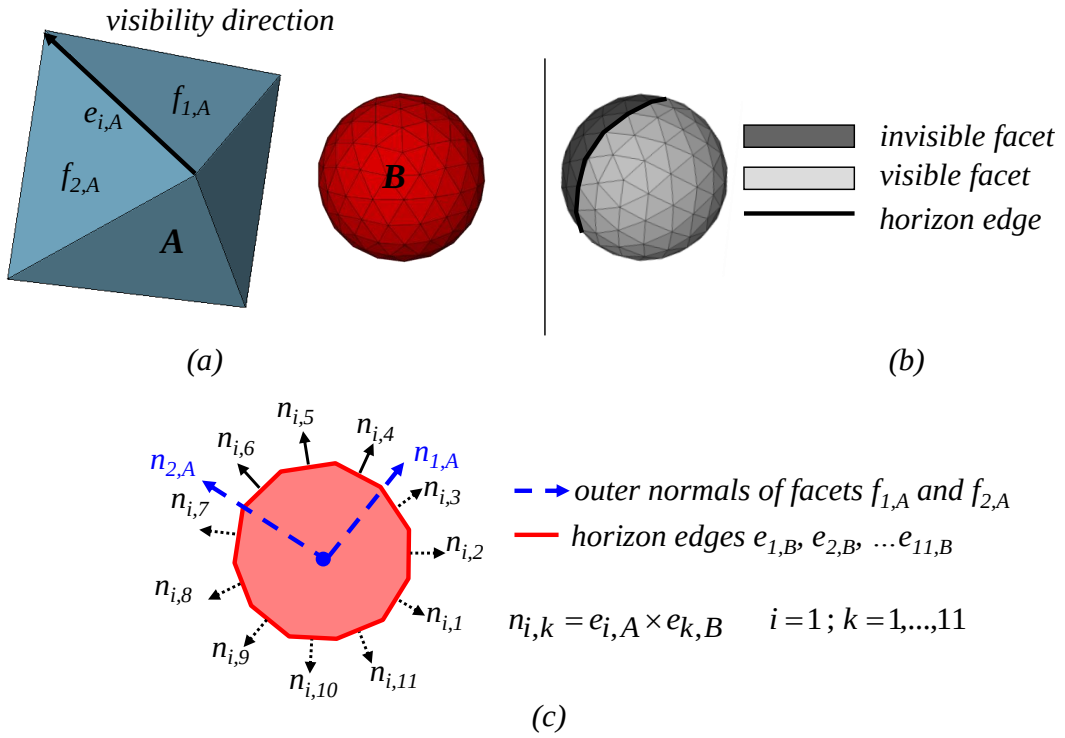


Fig. 8. (a) An edge $e_{i,A}$ as a visibility direction . (b) Visibility computation of all facets of B w.r.t. $e_{i,A}$ and horizon edges determination. (c) Only horizon edges $e_{4,B}$, $e_{5,B}$, and $e_{6,B}$ will produce edge facets $e_{i,A} \oplus e_{4,B}$, $e_{i,A} \oplus e_{5,B}$, and $e_{i,A} \oplus e_{6,B}$.

hanced and Simplified Slope Diagrams-based Minkowski Sum (ESSDMS) algorithm. We first describe the principles behind the original slope diagram. After that, we describe the ESSDMS algorithm and indicate the role played by contributing vertices in the improvement of performance (by eliminating unnecessary processing) and the simplification of some steps.

5.1 Original slope diagram-based Minkowski sum algorithm

Slope diagrams were proposed for the first time by Ghosh [33]. The principle is to represent two convex and closed two-manifold polyhedra A and B in a dual space such that the orientation of the normals of the facets, edges, and vertices of the two operands determine the Minkowski sum polyhedron S. The

dual space used by Ghosh is a unit sphere. For a convex polyhedron A, the slope diagram representation $SD(A)$ is obtained as follows:

- Each facet $f_{i,A}$ with an outer unit normal $n_{i,A}$ is represented by a spherical point corresponding to the end point of the outer unit normal $n_{i,A}$. This point is referred to as a *facet point*.
- Each edge $e_{j,A}$ is represented on the unit sphere by the arc of the great circle joining the two facet points representing the two facets of A incident to the edge $e_{j,A}$. The spherical arc representing the edge $e_{j,A}$ is called an *edge arc*. By "the arc of the great circle" we mean the shorter arc among the two ones joining the two facet points. The reason for choosing the shorter arc is that for convex polyhedra, the angle between two neighboring facets is always less than 180 degrees.
- Each vertex $v_{k,A}$ is represented by a region on the unit sphere (a spherical polygon) bounded by the facet points corresponding to the facets of A incident to the vertex $v_{k,A}$ and by the edge arcs corresponding to the edges of A incident to $v_{k,A}$. We call this spherical polygon a *vertex polygon* or a *vertex region*.

The slope diagram of polyhedron B is obtained by following the same steps used for polyhedron A. Fig. 9 shows a slope diagram representation of a convex and closed polyhedron.

The Minkowski sum polyhedron $A \oplus B$ is obtained by merging the two slope diagrams $SD(A)$ and $SD(B)$. The merging process aims at deducing the sum facets following the intersections or coincidences of the two slope diagrams components (facet points, edge arcs, and vertex regions). Four intersection cases are detected:

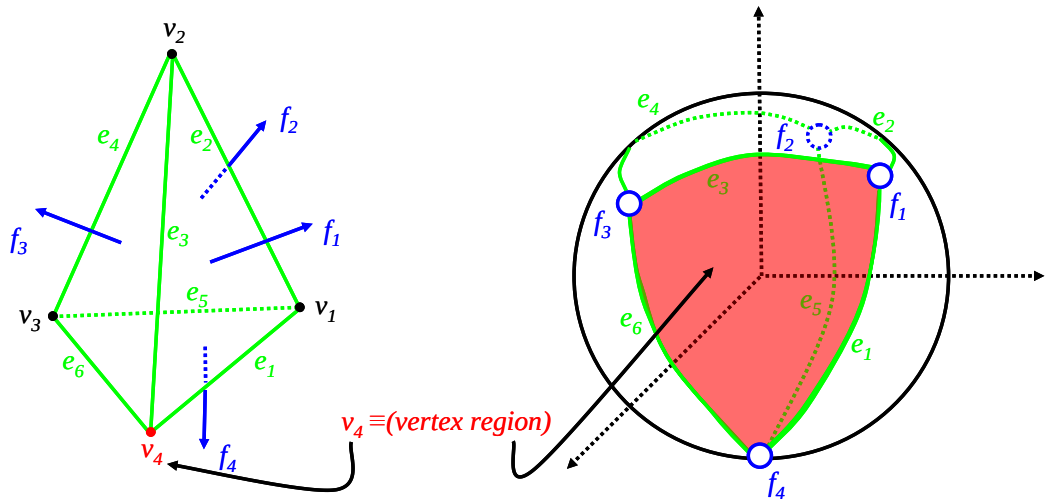


Fig. 9. A convex polyhedron and its slope diagram representation (revised from [33]).

- **Intersection of two facet points:** this occurs when a facet point of $SD(A)$ is coincident with a facet point of $SD(B)$. The corresponding Minkowski sum facet is the result of a planar Minkowski sum of the two facets of A and B represented by the two coincident facet points.
- **Intersection of a facet point with an edge arc:** this occurs when a facet point of one of the slope diagrams lies on an edge arc of the other slope diagram. The corresponding Minkowski sum facet is the result of a planar Minkowski sum of the facet represented by the facet point and the edge represented by the edge arc.
- **Intersection of two edge arcs:** this occurs when two non-parallel edge arcs one from $SD(A)$ and the second from $SD(B)$ have a common intersection point. The corresponding Minkowski sum facet is the result of a planar Minkowski sum of the two non-parallel edges represented by these two edge arcs. Two edges arcs are non-parallel if they do not lie on the same great circle of the unit sphere.
- **Intersection of a facet point with a vertex region:** this occurs when

a facet point of one of the two slope diagrams lies inside a vertex region of the second slope diagram. The corresponding Minkowski sum facet is a translated version of the facet represented by the facet point (the sum facet has the same shape as this facet), the translation vector is defined by the coordinates of the vertex represented by the vertex region.

In order to merge the two slope diagrams embedded on the unit sphere, Ghosh transformed them into a 2D planar form by means of stereographic projection. The stereographic projection is a projection of a sphere from one of the points N or N' onto the plane σ tangent to the sphere in the diametrically opposite point N' or N (see Fig. 10).

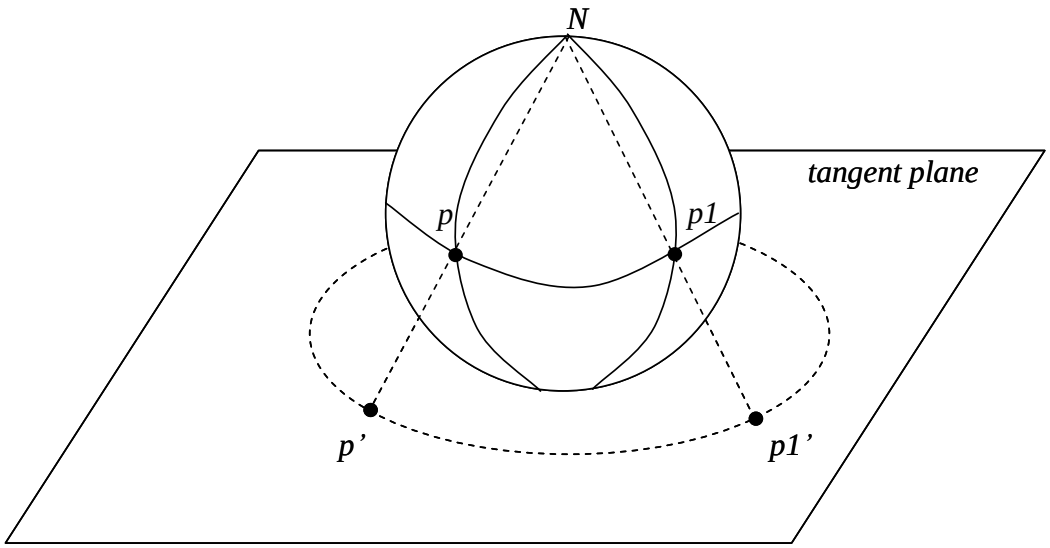


Fig. 10. Stereographic projection used to merge two slope diagrams (revised from [33]).

Although Ghosh formulated slope diagram for the computation of the Minkowski sum of convex and non-convex polyhedra, the existent implementations are only restricted to convex polyhedra. Furthermore, the weakness of the slope diagrams based Minkowski sum algorithms come from the embedding on the unit sphere and from the stereographic projection used to merge two slope

diagrams. These two steps are relatively complex, require a non-negligible computation time, and lead to precision problems when working with built-in number types such as float or double.

To overcome these two problems, we propose the ESSDMS algorithm which is somewhat similar to the improved slope diagram algorithm presented by Wu et al. [1] and detailed in the next section.

5.2 The ESSDMS algorithm

The ESSDMS algorithm we propose here comes with simplifications and enhancements to the algorithm of Wu's et al.. It uses the concept of contributing vertices to gain additional performance and simplify some steps of the algorithm. Its implementation is based on exact number types instead of built-in number types to avoid using position thresholds. So the ESSDMS algorithm is similar to that of Wu et al. in the following points:

- **No embedding of the slope diagram representation:** the slope diagram principle is exploited without explicitly embedding objects in the unit sphere. When information about facet points, edge arcs, or vertex regions is needed, it is directly extracted from the polyhedra we are working with.
- **No stereographic projection:** because there is no explicit embedding or representation of slope diagrams, there is nothing to project. The intersections between slope diagram components are detected by the use of geometric operations obtained directly from polyhedra.

In contrast, the ESSDMS algorithm differs from that of Wu et al. in the following points:

- **Use of the contributing vertices concept**: this enhancement reduces the number of edges used in the determination of the intersections between edge arcs by considering only those having incident facets associated to common contributing vertices. Moreover, it allows to avoid the point-in-polygon queries needed when testing if a facet point lies inside a vertex region (intersection of a facet point and a vertex region). If we perform all other intersection queries (intersection of two facet points, intersection of a facet point and an edge arc, and the intersection of two edge arcs), it is clear that the remaining facets correspond to facet points lying necessarily inside vertex regions. So the corresponding facets in the Minkowski sum polyhedron are translated copies of these facets. By analogy with the CVMS algorithm, we can conclude that each of these facets has only one contributing vertex. The creation of the sum facets corresponding to the remaining facets is done by using their contributing vertices. The computation of the contributing vertices does not introduce additional performance overhead (compared to the cost of detecting intersections) since they are deduced directly from the intersection configuration.
- **No round-off errors:** this is an implementation issue, by using exact number types (such those provided by GMP (GNU Multi Precision) library [36], we ensure that computations are done exactly and intersections are computed correctly. This avoids the use of position tolerance values or thresholds when testing coincidences of geometric features.

Given two convex, closed and two-manifold polyhedra A and B, the four intersection cases are handled as the following:

- **Intersection of two facet points:** two facet points are coincident if the two outer normals a and b of the corresponding facets $f_{i,A}$ and $f_{j,B}$ of A

and B respectively have the same orientation (see Fig. 11.a); this is the case when a and b satisfy:

$$a \times b = 0 \; and \; \langle a, b \rangle > 0 \tag{6}$$

The first part of equation 6 (the cross product) checks if a and b are on the same supporting line (since they have the same origin point, they have the same supporting line if they are parallel). The second part of equation 6 (the dot product) checks if a and b have the same orientation since they can be on the same supporting line but with opposite orientations.
The corresponding Minkowski sum facet f_S for this case is the result of a planar Minkowski sum of $f_{i,A}$ and $f_{j,B}$:

$$f_S = f_{i,A} \oplus f_{j,B} \tag{7}$$

Since the facets $f_{i,A}$ and $f_{j,B}$ have parallel supporting planes, the contributing vertices associated to the facet $f_{i,A}$ are the vertices of $f_{j,B}$ and the contributing vertices associated to the facet $f_{j,B}$ are the vertices of $f_{i,A}$.

- **Intersection of a facet point with an edge arc:** a facet point f corresponding to a facet of one of the two slope diagrams lies on an edge arc st of the other slope diagram (s and t are facet points corresponding to the facets incident to the edge represented by the edge arc st) if it lies on the great circle passing through facet points s and t and if f lies on the shorter arc joining s and t (see Fig. 11.b). These two conditions are equivalent to:

$$f \times n = 0 \; where \; n = s \times t \tag{8}$$

$$\langle s \times f, n \rangle > 0 \; and \; \langle f \times t, n \rangle > 0 \tag{9}$$

Equation 8 checks if the facet point f lies in the great circle passing through s and t, this is the case when f is perpendicular to the normal n of the great

circle. Equation 9 checks if the facet point f lies on the shorter arc of the great circle joining s and t.

The corresponding Minkowski sum facet f_S for this configuration is the result of a planar Minkowski sum of the facet represented by the facet point f and the edge represented by the edge arc st. For example, if the facet point f corresponds to a facet $f_{i,A}$ of A and the edge arc on which f lies corresponds to an edge $e_{j,B}$ of B, the Minkowski sum facet f_S of S is given by:

$$f_S = f_{i,A} \oplus e_{j,B} \tag{10}$$

In this case, the contributing vertices associated to the facet $f_{i,A}$ represented by the spherical point f are the two vertices incident to the edge $e_{j,B}$ represented by the edge arc st.

By commutativity, if the facet point f corresponds to a facet $f_{i,B}$ and the edge arc on which f lies corresponds to an edge $e_{j,A}$, the Minkowski sum facet f_S is given by equation 11 and the contributing vertices associated to the facet $f_{i,B}$ are the two vertices incident to the edge $e_{j,A}$.

$$f_S = f_{i,B} \oplus e_{j,A} \tag{11}$$

- **Intersection of two edge arcs:** before trying to determine if two non-parallel edge arcs st and $s't'$ representing two edges $e_{i,A}$ and $e_{j,B}$ of A and B have a common intersection point, we must check if the contributing vertices associated to the facets incident to $e_{i,A}$ have distinct contributing vertices. If this is not the case, we conclude that the corresponding edge arc st will not intersect any edge arc $s't'$. Therefore, it will no longer be considered in the subsequent edge arcs intersection queries. In a similar manner, we must also check the contributing vertices associated to the facets incident to $e_{j,B}$.

If there is at least one common contributing vertex, the edge arc $s't'$ will also be discarded from the rest of the iterations.

If edge arcs st and $s't'$ pass the previous step, we continue by checking if one of the two spherical points resulting from the intersection of the two great circles supporting st and $s't'$ lies on st and $s't'$ at the same time (see Fig. 11.c). Suppose n is the normal of the circle containing st, i.e. $n = s \times t$ and n' the normal vector of the great circle containing $s't'$, i.e. $n' = s' \times t'$. The two spherical points x and x' resulting from the intersection of the two great circles are obtained by the cross products $x = n \times n'$ and $x' = n' \times n$ (see Fig. 11.c). To check that x (or x') lies on the edge arcs st and $s't'$, we can consider it as a facet point and use equations 8 and 9 derived for the intersection of a facet point and an edge arc.

The corresponding Minkowski sum facet f_S for this configuration is the result of a planar Minkowski sum of two edges $e_{i,A}$ and $e_{j,B}$ represented by the two edge arcs st and $s't'$:

$$f_S = e_{i,A} \oplus e_{j,B} \tag{12}$$

- **Intersection of a facet point with a vertex region:** from Fig. 11.d, it seems that checking if a facet point f of one of the two slope diagrams lies within the interior of a vertex region (gray-shaded in Fig. 11.d) of the other slope diagram can be done by algorithms used to test if a point lies within a polygon or not (such as ray-crossing for example). In our algorithm, we will bypass the point-in-polygon queries by using some properties of the CVMS algorithm. The CVMS algorithm states that for convex polyhedra, a facet contributes only once in the Minkowski sum polyhedron construction. Thus, after enumerating the first three kinds of intersections (two facet points, a facet point and an edge arc, and two edge arcs), it is clear that the remaining

facets of A and B will necessarily lie within vertex regions (avoiding point in polygon queries). The vertex v corresponding to the vertex region in which lies a particular facet point f coincides with the contributing vertex associated to that facet. So, all we need to do is to find the contributing vertex corresponding to the facet represented by the facet point f (see equation 3).

The corresponding Minkowski sum facet f_S is obtained by translating the facet represented by the facet point f by a position vector pointing from the coordinates origin and ending at the corresponding contributing vertex.

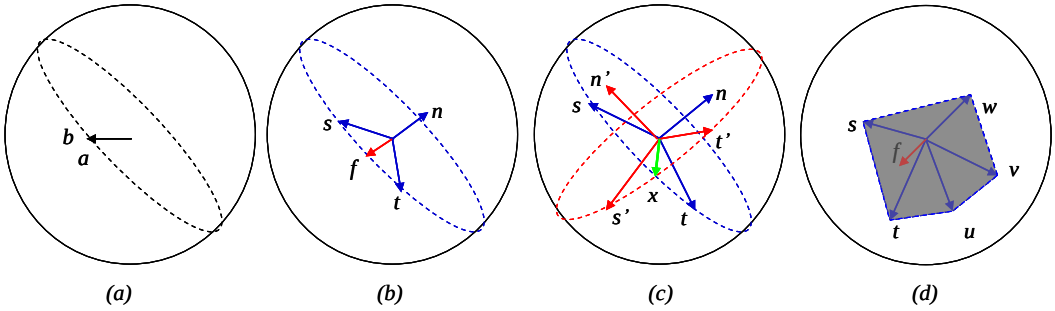


Fig. 11. Intersections in slope diagrams. (a) Intersection of two facet points. (b) Intersection of a facet point and an edge arc. (c) Intersection of two edge arcs. (d) Intersection of a facet point and a vertex region.

6 Implementation and performance

In this section, we describe the implementations of the CVMS algorithm, the ESSDMS algorithm, and the Convex Hull-based Minkowski Sum (CHMS) algorithm. After that, we compare the performance of these approaches with other methods found in the literature and present examples of Minkowski sum polyhedra we computed.

The CVMS algorithm has been implemented using C++ and CGAL. We used the same environment for the implementation of the ESSDMS and the CHMS algorithms. The convex polyhedra are handled by the CGAL `Polyhedron_3` data structure.

The CHMS algorithm is implemented by using the CGAL function `convex_hull_3` for the computation of convex hulls in 3D. This function is an implementation of the Quickhull algorithm [10].

For the computation of the planar Minkowski sums involved in the CVMS and the ESSDMS algorithms, we used the function `minkowski_sum_2` provided in the 2D Minkowski sum package of CGAL. This function implements the convolution operation [30].

As stated previously, we guarantee the exactness of the obtained results by using exact number types. For our implementation, we used the exact number types provided by the GNU Multi Precision (GMP) library [36] under CGAL. By using exact number types, we penalize run-time performance. In fact, exact number types are very slow compared to the built-in floating point number types. To overcome this problem, we used the lazy kernel adapter [37] which speeds up exact computations.

We also re-implemented the improved 3D-MSSD algorithm of Wu et al. using the same components of CGAL. Our implementation of the improved 3D-MSSD algorithm guarantees exact results because it uses exact number types, so it does not rely on position thresholds as done in the work of Wu at al..

In the case of the CVMS algorithm, finding the contributing vertices for the f_A facets of A takes $O(f_A v_B)$ time. Similarly, finding the contributing vertices for the f_B facets of B takes at most $O(f_B v_A)$ time. The determination of translated facets takes $O(f_A)$ time since it is done directly by examining the number of contributing vertices associated to each facet. In the same manner, constructing corner facets takes at most $O(f_B)$ time. For edge facets, it takes at most $f(e_A e_B)$ time to find them. By adding times of all steps, we conclude that the CVMS algorithm has a worst-case time complexity of $O(f_A v_B + f_B v_A + f_A + f_B + e_A e_B)$.

For the ESSDMS algorithm, the computation of the intersections of two facet points requires $O(f_A f_B)$ time, the computation of the intersections of facet points and edge arcs requires $O(f_A e_B + f_B e_A)$ time, and the computation of the intersection of two edge arcs requires at most $O(e_A e_B)$ time. Since the computation of the intersection of a facet point and a vertex region reduces to finding the contributing vertices corresponding to the remaining facet points (that have not yet contributed into the sum), the complexity of finding such intersections is at most $O(f_A v_B + f_B v_A)$. By summing all these times, we conclude that the ESSDMS algorithm has a worst-case time complexity of $O(f_A f_B + f_A e_B + f_B e_A + e_A e_B + f_A v_B + f_B v_A)$.

For the Convex Hull-based Minkowski Sum (CHMS) algorithm we implemented, we performed vector addition of all points of A and all points of B in order to build a point cloud C. Then, we computed the convex hull of the point cloud C using the Quickhull algorithm. The first step takes $O(v_A v_B)$

time and the second has a worst-case complexity of $O((f_A + f_B)^2)$ [10]. Thus, the CHMS algorithm has a worst-case time complexity of $O((f_A+f_B)^2+v_Av_B)$.

6.3 Performance benchmark

For our benchmark, we computed the Minkowski sum of several polyhedra using the CVMS algorithm, the ESSDMS algorithm, the CHMS algorithm, and the improved 3D-MSSD algorithm we re-implemented.

We also compared the above mentioned algorithms with three other algorithms. The first one is based on Nef polyhedra embedded on the sphere and implemented by Hachenberger [24], the second one is a method based on linear programming implemented by Weibel [38] (following the work of Fukuda [39]), and the third one is the Cubical Gaussian Map-based (CGM-based) algorithm of Fogel and Halperin [35].

We shall note that the method based on Nef polyhedra is more powerful than necessary for the Minkowski sum computation since it aims at overlaying two arbitrary Nef polyhedra embedded on the sphere (with lower dimensional features, unbounded or bounded boundaries, etc.). A second remark concerns the implementation of Weibel which is intended for the computation of polytopes (convex polyhedra in $\mathbb{R}^n$) in an arbitrary dimension, so it is not specifically optimized for convex polyhedra in $\mathbb{R}^3$.

The experiments where done on a 2 GB RAM, 2.2 GHZ Intel Core 2 Duo personal computer. We used a cube and a sphere with varying facets number as input polyhedra. A comparison between running times is given in table 1:

Table 1

Running times of the Minkowski sum computation of convex polyhedra performed by several algorithms.

Operands (facets number)		Running time (sec.)							
A	B	CHMS	ESSDMS	**CVMS**	MSSD	NGM	Fuk.	CGM* (overlay)	CGM** (whole)
Cube(6)	Tetra(4)	0.016	0.015	**0.015**	0.031	0.093	0.096	0.015	0.030
Cube(6)	Hedra(8)	0.015	0.015	**0.015**	0.078	0.093	0.100	0.015	0.030
Cube(6)	Truncated cube(14)	0.046	0.015	**0.015**	0.110	0.110	0.232	0.015	0.030
Cube(6)	Sphere1(20)	0.046	0.046	**0.015**	0.156	0.203	0.300	0.015	0.046
Cube(6)	Sphere2(80)	0.125	0.093	**0.031**	0.359	0.421	0.786	0.047	0.124
Cube(6)	Sphere3(320)	0.546	0.250	**0.078**	1.156	1.438	3.269	0.078	0.296
Cube(6)	Sphere4(1280)	2.438	0.735	**0.266**	4.344	5.563	15.224	0.203	1.249
Cube(6)	Sphere5(5120)	11.407	2.719	**1.375**	17.938	22.625	69.956	0.562	7.406
Cube(6)	Sphere6(20480)	77.422	16.828	**12.234**	79.750	94.157	146.870	1.921	219.592

CHMS - Convex Hull-based Minkowski Sum algorithm, ESSDMS - Enhanced and Simplified Slope Diagram-based Minkowski Sum algorithm, CVMS - Contributing Vertices-based Minkowski Sum algorithm, MSSD - improved 3D-MSSD algorithm of Wu et al., NGM - Nef polyhedra-based algorithm of Hachenberger et al., Fuk. - Weibel's implementation of the algorithm of Fukuda. CGM - Cubical Gaussian Map-based (CGM-based) algorithm of Fogel and Halperin.

* only the overlay step of the CGM-based algorithm.

** the whole CGM-based algorithm: the construction of the cubical Gaussian maps plus their overlay.

Fig. 12 shows the ratio of the running time for the algorithms mentioned in table 1 to the sum of the number of facets of several polyhedra (the sum of the number of facets of A and B).

The CGM-based algorithm of Fogel and Halperin proceeds in two steps: first it computes the cubical Gaussian maps of the polyhedra A and B, then it overlays the computed CGMs in order to obtain the result. In table 1, the ninth column reports the running time of the overlay step and the tenth column

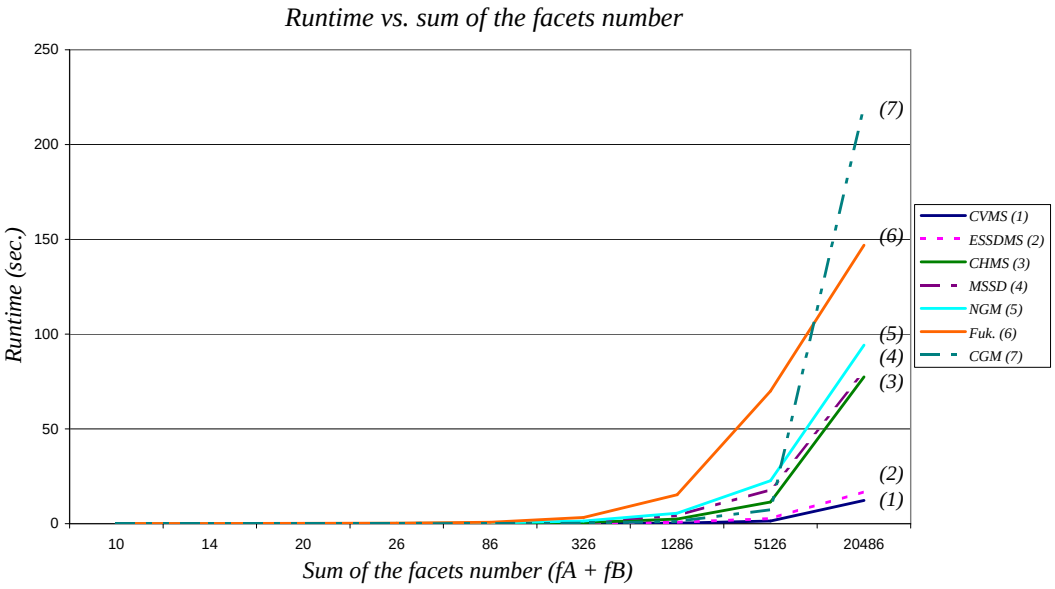


Fig. 12. Evolution of the running time of the algorithms mentioned in table 1 with respect to the sum of the number of facets of the operands. This figure shows that when increasing the number of facets, the CVMS and the ESSDMS algorithms are largely preferred in comparison to other algorithms.

reports the running time of the whole CGM-based algorithm (the construction of the cubical Gaussian maps plus their overlay). In order to have a fair comparison of our CVMS algorithm, the CGM-based algorithm of Fogel and Halperin, and the other algorithms, we must consider the running times reported in the tenth (the last) column as the running time of the CGM-based algorithm since the overlay process alone is not sufficient for the computation of the Minkowski sum without the construction of the cubical Gaussian maps.

The running time reported in table 1 show that our CVMS algorithm performs better than all other algorithms. Specifically, the CVMS algorithm is more efficient than the ESSDMS algorithm. If we eliminate the common terms $O(f_A v_B + f_B v_A + e_A e_B)$ in the time complexities of the CVMS and the ESSDMS algorithms, it is clear that $O(f_A + f_B)$ (the remaining term in the CVMS time complexity) is significantly inferior to $O(f_A f_B + f_A e_B + f_B e_A)$ (the remaining

term in the ESSDMS time complexity). This proves that the CVMS algorithm performs better than the ESSDMS algorithm and consolidates experimental results. Table 1 shows also that our CVMS and ESSDMS algorithms are much more efficient than the CGM-based algorithm of Fogel and Halperin; the computation of the cubical Gaussian maps in their algorithm is the most time consuming step when compared to the overlay one (especially for polyhedra with high complexity).

The algorithm implemented by Weibel is the slowest among all these algorithms since it is not optimized in $\mathbb{R}^3$. Another observation that is verified in table 1 is that the overlay process of the Nef polyhedra-based algorithm is not specifically devoted to the Minkowski sum computation of convex polyhedra.

An interesting fact of table 1 is that a good convex hull-based algorithm can perform better than other algorithms. In the present case, the CHMS algorithm performs better than the improved 3D-MSSD algorithm of Wu et al..

6.4 Examples

Fig. 13 shows some examples of Minkowski sum polyhedra computed by the CVMS algorithm. In Fig. 14, we included more results to show that the CVMS algorithm works with complex or big size polyhedra (having thousands or tens of thousands of facets; see Fig. 14.a and b). The CVMS algorithm handles well degenerate cases that require special treatment with some other methods. An example of a degenerate case is that of two facets having parallel supporting planes (having the same outer unit normal) one from polyhedron A and the

other from polyhedron B. An extreme degenerate case occurs when $A = B$ (two convex polyhedra with identical sets of facet outer normals). These degenerate cases do not require any special handling by the CVMS algorithm. Fig 14.c shows the result of such an extreme case correctly computed by the CVMS algorithm. More results of the CVMS algorithm can be found at: `http://liris.cnrs.fr/hichem.barki/mksum/CVMS-convex`

7 Conclusion and future work

In this work, we have presented the CVMS algorithm for the computation of the Minkowski sum of convex polyhedra. The CVMS algorithm is based on the concept of contributing vertices. By finding the contributing vertices for polyhedra facets, it is straightforward to deduce translated and corner facets of the Minkowski sum polyhedron. Furthermore, the concept of contributing vertices allows additional gain of performance when constructing edge facets by examining only relevant edges from input polyhedra.

We also took benefit from the concept of contributing vertices in the ESSDMS algorithm. This results in more simplicity by eliminating point in polygon queries and performance gain by reducing the number of edge arcs intersection requests.

The proposed algorithms are easy to implement and they guarantee the exactness of the result by using exact number types. The complexity study and the experimental results show that the CVMS algorithm is faster than the ESSDMS algorithm and that the CVMS algorithm is more efficient than the other algorithms proposed in the literature and cited in this paper. The CVMS

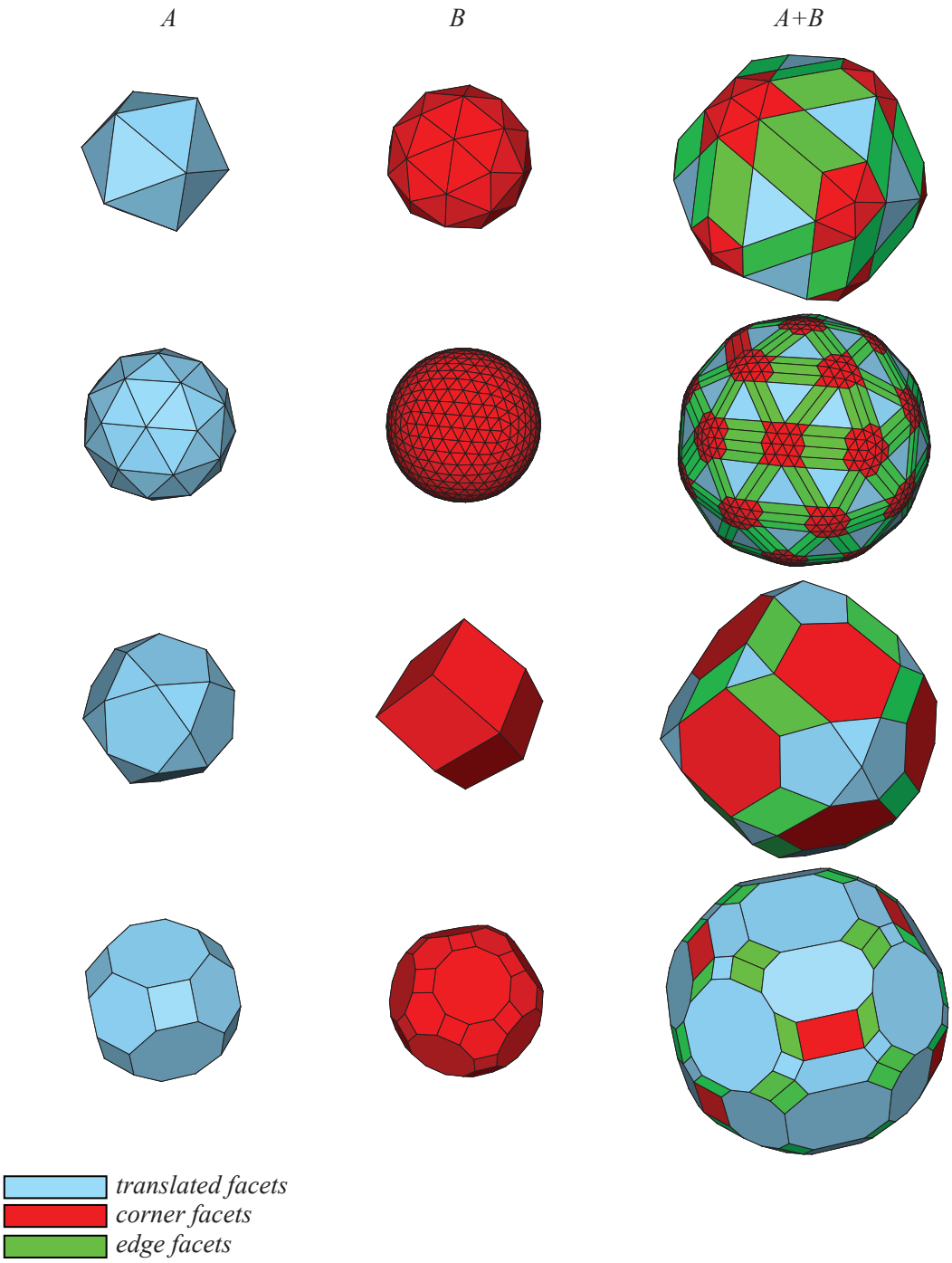


Fig. 13. Minkowski sum examples for convex polyhedra generated by the CVMS algorithm.

algorithm can be used as an efficient replacement for the convex hull-based algorithms in order to speed-up the computation of the pairwise Minkowski sums for non-convex polyhedra (along with convex-decomposition and union

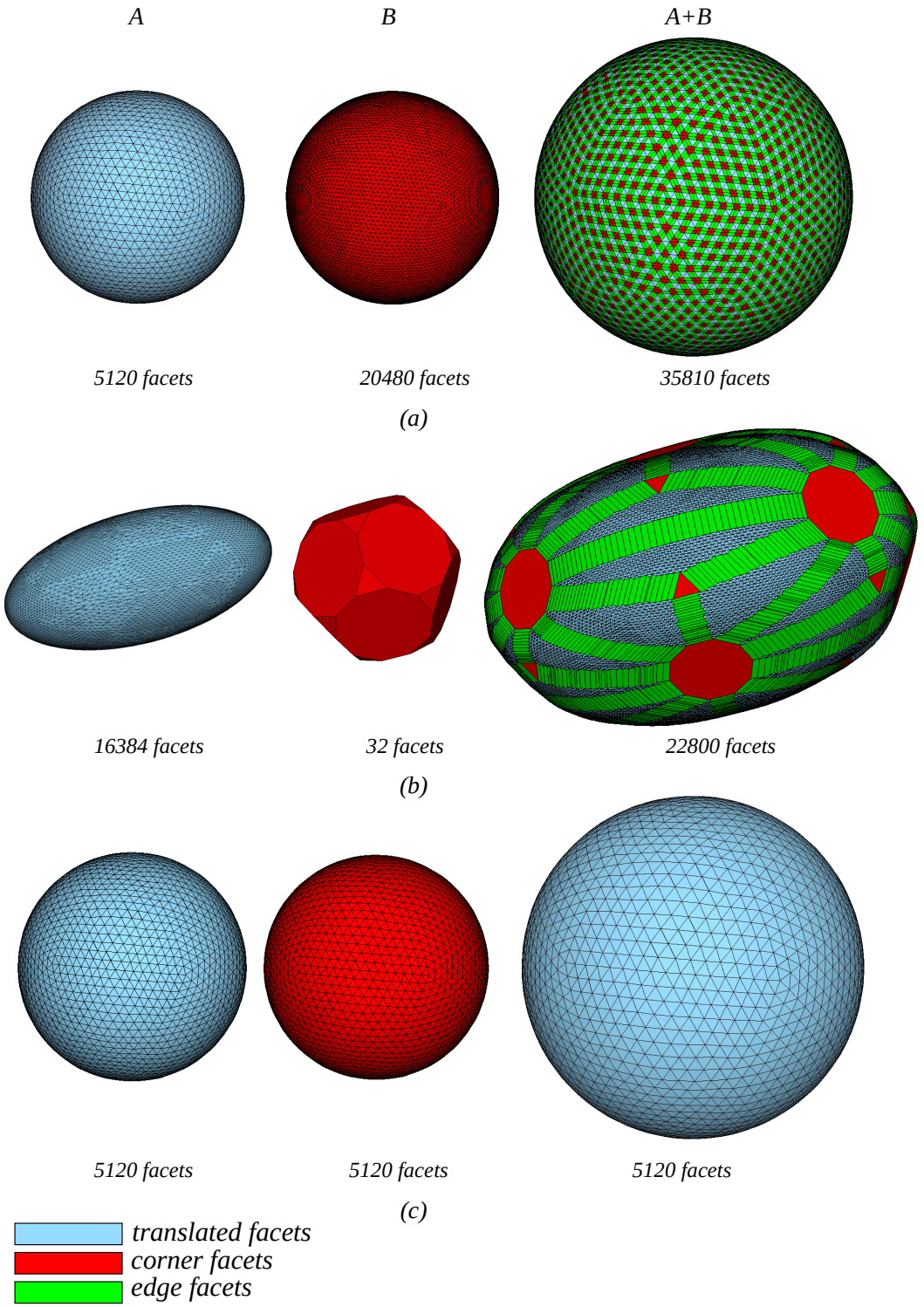


Fig. 14. More results of the CVMS algorithm for: (a) two spheres having thousands and tens of thousands of facets. (b) an ellipsoid-like polyhedron and a truncated dodecahedron. (c) for an extreme degenerate case (two identical spheres with thousands of facets).

steps).

As a part of our future work, we are generalizing the CVMS algorithm to non-convex polyhedra. The main idea is that the boundary of each non-convex polyhedron can be seen as the union of disjoint convex surface patches (sets of facets containing no reflex edges). So by treating polyhedra in a local manner (working on each convex patch) and by correctly handling reflex edges, it is possible to generate a superset of the Minkowski sum facets. To extract the boundary of the Minkowski sum, we plan to use arrangements and surface envelope computations for which CGAL provides powerful packages.

8 Acknowledgment

This work has been partly supported by the Cluster 2 of the Rhône-Alpes region within the LIMA Project. We are grateful to the help of P. Hachenberger and E. Fogel who provided us the necessary material and assistance for the use of their respective Nef polyhedra-based and cubical Gaussian map-based algorithms.

References

[1] Y. Wu, , J. Shah, J. Davidson, Improvements to algorithms for computing the Minkowski sum of 3-polytopes, Comput. Aided Des. 35 (13) (2003) 1181–1192.

[2] I.-K. Lee, M.-S. Kim, G. Elber, Polynomial/rational approximation of Minkowski sum boundary curves, Graph. Models Image Process. 60 (2) (1998) 136–165.

[3] A. Kaul, J. Rossignac, Solid-interpolating deformations: construction and animation of PIPs, Comput. Graph. 16 (1) (1992) 107–115.

[4] J. Serra, Image Analysis and Mathematical Morphology, Vol. 1, Academic Press, London, 1982.

[5] J. Serra, Image Analysis and Mathematical Morphology, Vol. 2, Academic Press, New York, 1988.

[6] A. Tuzikov, J. Roerdink, H. Heijmans, Similarity measures for convex polyhedra based on Minkowski addition, Pattern Recognition 33 (2000) 979–995.

[7] Y. Kim, M. Otaduy, M. Lin, D. Manocha, Fast penetration depth estimation using rasterization hardware and hierarchical refinement, in: SCG '03: Proc. of the Nineteenth Annual Symposium on Computational Geometry, ACM, New York, NY, USA, 2003, pp. 386–387.

[8] T. Lozano-Pérez, Spatial planning: a configuration space approach, IEEE Trans. Comput. 32 (2) (1983) 108–120.

[9] P. Ghosh, K. Kumar, Support function representation of convex bodies, its application in geometric computing, and some related representations, Comput. Vis. Image Underst. 72 (3) (1998) 379–403.

[10] C. B. Barber, D. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Softw. 22 (4) (1996) 469–483.

[11] R. Seidel, Output-size sensitive algorithms for constructive problems in computational geometry, Ph.D. thesis, Faculty of the Graduate School, Ithaca, NY, USA (1987).

[12] M. Kallay, The complexity of incremental convex hull algorithms in $\mathbb{R}^d$, Inf. Process. Lett. 19 (4) (1984) 197.

[13] D. Chand, S. Kapur, An algorithm for convex polytopes, J. ACM 17 (1) (1970) 78–86.

[14] F. Preparata, S. Hong, Convex hulls of finite sets of points in two and three dimensions, Commun. ACM 20 (2) (1977) 87–93.

[15] R. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Inf. Process. Lett. 1 (4) (1972) 132–133.

[16] J. O'Rourke, Computational Geometry in C, 2nd edition, Cambridge University Press, New York, NY, USA, 1998.

[17] B. Chazelle, Convex decompositions of polyhedra, in: STOC '81: Proc. of the Thirteenth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, 1981, pp. 70–79.

[18] B. Chazelle, D. Dobkin, N. Shouraboura, A. Tal, Strategies for polyhedral surface decomposition: an experimental study, Comput. Geom. Theory Appl. 7 (5-6) (1997) 327–342.

[19] B. Aronov, M. Sharir, B. Tagansky, The union of convex polyhedra in three dimensions, SIAM J. Comput. 26 (6) (1997) 1670–1688.

[20] D. Halperin, Robust geometric computing in motion, Int. J. Robotics Res. 21 (3) (2002) 219–232.

[21] D. Dobkin, J. Hershberger, D. Kirkpatrick, S. Suri, Computing the intersection-depth of polyhedra, Algorithmica 9 (1993) 518–533.

[22] The CGAL project homepage. http://www.cgal.org.

[23] W. Nef, Beiträge zur theorie der polyeder (1978).

[24] P. Hachenberger, L. Kettner, K. Mehlhorn, Boolean operations on 3d selective nef complexes: data structure, algorithms, optimized implementation and experiments, Comput. Geom. Theory Appl. 38 (1-2) (2007) 64–99.

[25] P. Hachenberger, L. Kettner, Boolean operations on 3d selective nef complexes: optimized implementation and experiments, in: SPM '05: Proc. of the 2005 ACM symposium on Solid and Physical Modeling, ACM, New York, NY, USA, 2005, pp. 163–174.

[26] P. Hachenberger, Exact Minkowski sums of polyhedra and exact and efficient decomposition of polyhedra in convex pieces, in: Proc. 15th Annual European Symposium on Algorithms, 2007, pp. 669–680.

[27] G. Varadhan, D. Manocha, Accurate Minkowski sum approximation of polyhedral models, Graph. Models 68 (4) (2006) 343–355.

[28] G. Varadhan, S. Krishnan, T. Sriram, D. Manocha, Topology preserving surface extraction using adaptive subdivision, in: SGP '04: Proc. of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, ACM, New York, NY, USA, 2004, pp. 235–244.

[29] J.-M. Lien, Point-based Minkowski sum boundary, in: PG '07: Proc. of the 15th Pacific Conference on Computer Graphics and Applications (PG'07), IEEE Computer Society, Washington, DC, USA, 2007, pp. 261–270.

[30] L. J. Guibas, L. Ramshaw, L. Stolfi, A kinetic framework for computational geometry, in: Proc. of 24th annual IEEE Symposium on the Foundation of Computer Science, 1983, pp. 100–111.

[31] L. Guibas, R. Seidel, Computing convolutions by reciprocal search, Discrete Comput. Geom. 2 (1987) 175–193.

[32] J. Basch, L. Guibas, G. Ramkumar, L. Ramshaw, Polyhedral tracings and their convolution, in: Proc. of 2nd Workshop on the Algorithmic Foundations of Robotics, 1996, pp. 171–184.

[33] P. Ghosh, A unified computational framework for Minkowski operations, Comput. Graph. 17 (4) (1993) 357–378.

[34] H. Bekker, J. Roerdink, An efficient algorithm to calculate the Minkowski sum of convex 3d polyhedra, in: ICCS '01: Proc. of the International Conference on Computational Sciences-Part I, Springer-Verlag, London, UK, 2001, pp. 619–628.

[35] E. Fogel, D. Halperin, Exact and efficient construction of Minkowski sums of convex polyhedra with applications, Comput. Aided Des. 39 (11) (2007) 929–940.

[36] The GNU MP bignum library. http://gmplib.org/.

[37] A. Fabri, S. Pion, A generic lazy evaluation scheme for exact geometric computations, in: Proc. 2nd Library-Centric Software Design, 2006.

[38] C. Weibel, Minkowski sums, http://roso.epfl.ch/cw/poly/public.php.

[39] K. Fukuda, From the zonotope construction to the minkowski addition of convex polytopes, Journal of Symbolic Computation 38 (4) (2004) 1261–1272.