FIMS: a New and Efficient Algorithm for the Computation of Minkowski Sum of Convex Polyhedra

Hichem Barki, Florence Denis, and Florent Dupont
LIRIS UMR CNRS 5205 – Université de Lyon – Université Claude Bernard Lyon 1

***Abstract*—The Minkowski sum computation and implementation in 2D and 3D domains is of a particular interest because it has a large number of applications in many domains such as: mathematical morphology, image processing and analysis, robotics, spatial planning, computer aided design and manufacturing, image processing … However, no exact, fast, and general algorithms are found in the literature. We present in this paper a new and efficient algorithm based on a simple idea, for the calculation of the Minkowski sum of convex and closed polyhedra. Our implementation is general in the sense that it does not assume any constraint on the positions or the sizes of the polyhedra; it produces exact results and is faster than algorithms based on the convex hull computation. Our method can be easily generalized to an arbitrary dimensional space. We are also working on its adaptation to convex polyhedra which are not necessarily closed and for non-convex polyhedra without passing through the decomposition and union steps.**

***Index Terms*—Minkowski Sum Algorithm, Mesh filtering, Morphological operations**

I. Introduction

THE Minkowski sum or addition of two sets *A* and *B* was defined by the German mathematician Hermann Minkowski (1864-1909) as a vector addition of elements of *A* with elements of *B*:

$$A \oplus B = \{a + b \,/\, a \in A, b \in B\} \qquad (1)$$

The Minkowski sum of two sets *A* and *B* is referred to as the dilation of set *A* by *B*, where *B* is named the structuring element. This operation constitutes the base of the mathematical morphology which is widely used in image analysis.

Since the 60s, the mathematical morphology has found many applications in the image

The authors are within the LIRIS Laboratory, UMR CNRS 5205, Université Claude Bernard Lyon 1, 43 Bd. du 11 novembre 1918, F-69622 Villeurbanne, France (e-mail: hichem.barki@liris.cnrs.fr; florence.denis@liris.cnrs.fr; florent.dupont@liris.cnrs.fr). This work was supported in part by the Cluster 2 of the Rhône-Alpes region within the LIMA Project.

processing and analysis fields, such as image filtering, segmentation, skeletonizing … Nevertheless, this theory remains poorly or even none exploited for three-dimensional polyhedra. The existing solutions are either partially implemented or not implemented at all.

Mathematical morphology interests us particularly in the context of mesh filtering and analysis, to reduce or eliminate the noise due to acquisition procedures, to perform opening and closing operations, to fill holes, to identify particular characteristics (hit-or-miss operations), to skeletonize or segment meshes, etc. Minkowski sum implementation in three-dimensional space also makes improvements to other applications such as: computer-aided design and manufacturing [2], collision free path computation for robot motion planning [3], penetration depth computation and dynamic simulation [4]…

The dilation in image processing field is the basic operation for every morphological treatment; the other morphological operations (erosion, opening, closing) are duals or particular combinations of dilations [6]. The dilation is equivalent to the Minkowski sum of two image regions. In this work, since we aim at efficiently implement the exact Minkowski sum (or dilation) for three-dimensional polyhedra, we have firstly begun by doing this for convex polyhedra. Our goal in the future is to be able to perform morphological analysis (as it is performed on images) on arbitrary three-dimensional meshes.

We propose in this paper a new and efficient algorithm that computes the Minkowski sum of convex polyhedra. We call it the *FIMS algorithm (Facet's based Incremental Minkowski Sum)* because it operates on facets – rather than other algorithms that operate on vertices (convex hull based algorithms). The FIMS algorithm is based on a simple idea and can be generalized to arbitrary dimensional spaces. Moreover, it outputs exact results. Our aim is to achieve a fast and efficient Minkowski sum computation without passing through the decomposition and union steps that are required by most algorithms.

The rest of this paper is organized as follows: in section 2 we review previous work on Minkowski sum computation for polyhedra, section 3 presents our solution to the problem, i.e. the FIMS algorithm, we present experimental results and discussions in section 4 and finally, we give some future directions for our work in section 5.

II. Previous work

Since its birth in the 60s [1][5], mathematical morphology progressed much, as well on the theoretical level as on the practical one. Nevertheless, this progress touches mainly the field of discrete images and discrete volumes (grids of voxels). Very few algorithms treat effectively the case of three-dimensional meshes.

It follows that mathematical morphology tools are poorly exploited in the field of three-dimensional meshes. The existing solutions are:

- Either non implemented (only theoretical approaches);
- Either too slow when applied to big size polyhedra;
- Either restricted to convex polyhedra or requiring some additional constraints.

We classify algorithms used for the Minkowski sum computation into three categories: those based on convex hulls, those based on dual space representations of polyhedra, and those based on Nef polyhedra.

A. *Convex hull based algorithms*

The convex hull based algorithms are the most discussed and the most implemented in the literature. For two polyhedra *A* and *B*, they comprise the following steps:

- Vector addition of all points of *A* and all points of *B* in order to build a point cloud *C*.

• Convex hull computation for the point cloud *C* and construction of the Minkowski sum polyhedron *S*.

However, this treatment applies only if *A* and *B* are convex polyhedra; otherwise, the steps are:

• Convex decomposition for each non-convex polyhedron.

• Computation of the pair-wise Minkowski sum between all possible pairs of convex pieces.

• Computation of the union of the pair-wise Minkowski sums.

The decomposition of a non-convex polyhedron into convex pieces is known to be NP-hard. More than twenty years ago, Chazelle [16] 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's facets. Nevertheless, no practical or robust implementation has been found in the literature for Chazelle's optimal algorithm.

The construction of the convex hull requires a great cost of computation since it aims at distinguishing interior points from those which will form the convex hull. Several algorithms were developed for the convex hull computation of a set of points; a summary of these algorithms can be found in [8]. Among them, let us cite the gift-wrapping algorithm [7], the incremental algorithm [9][10], the divide and conquer algorithm [11]....

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 convex polyhedra, where n is the number of polyhedra facets [12]. Moreover, there is no robust implementation for the union computation of convex polyhedra that handles all degeneracies [14].

The complexity of the Minkowski sum computation is $O(n^2)$ for convex polyhedra. However, it can have $O(n^6)$ worst-case complexity for non-convex ones [13][14], where n denotes the number of facets of the two polyhedra.

As what has been said, the convex hull based algorithms are the most discussed and the most implemented in literature. However, they generate a great cost of computation and thus are very slow. Their benefit is that they are used for both convex and non-convex polyhedra.

Varadhan and Manosha [15] used convex hull based algorithms to approximate the Minkowski sum of polyhedra. They have decomposed the polyhedra into convex pieces [16][17] and computed the pair-wise Minkowski sums. Instead of computing the exact union of these pair-wise Minkowski sums (which constitute the bottleneck of the convex hull based algorithms), they approximated it in an adaptively subdivided voxel grid. They guaranteed a two-sided Hausdorff distance bound on the approximation. Their approach was time efficient but it does not compute the exact Minkowski sum of polyhedra. Therefore, it is not suitable for feature extraction and analysis of meshes.

B. Dual space based algorithms

Ghosh [18] proposed to compute the Minkowski sum in a dual space. Each polyhedron is represented on a unit sphere called the slope diagram of the polyhedron (see Fig. 1). For a particular polyhedron, each facet is represented by the spherical point corresponding to its normal vector extremity embedded on the unit sphere after it has been normalized to a unit vector (Fig. 1.b). Then, each edge is represented by the spherical arc of the great circle joining the two spherical points representing the two facets incident to this edge (Fig. 1.c). Finally, a vertex is represented by a spherical area bounded by spherical arcs and spherical points corresponding to incident edges and incident facets to this vertex (Fig. 1.d), respectively. The Minkowski sum of two polyhedra is then computed by merging the two slope diagrams and finding the intersections between the various components of the two slope diagrams.

Theoretically, these algorithms are valid for an arbitrary dimension but their implementation is limited for two-dimensional operands [19]. Stereographic projection used to merge two slope diagrams is complicated and affects the accuracy of the algorithm.

The slope diagrams based algorithms are not well discussed and rarely implemented in the literature. However, they are more efficient and faster than those based on convex hulls because they compute the Minkowski sum in a two-dimensional domain. Although slope diagrams were defined for convex and non-convex polyhedra, they are implemented only for convex polyhedra. Moreover, they generate miscalculations and are difficult to implement.

Another variant of slope diagrams based algorithms has been proposed by Fogel and Halperin [22]. They used a dual space representation of convex polyhedra that they named Cubical Gaussian Maps; their implementation is efficient and outputs exact results. However, it is restricted to convex polyhedra.

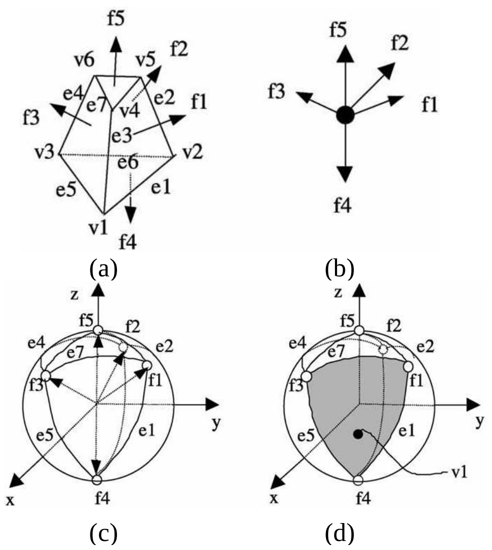

Fig. 1. A frustum and its slope diagram representation. (a) A frustum. (b) The normals to its facets. (c) The slope diagram embedded on the unit sphere. (d) The shaded spherical polygon is the dual of the vertex v_1 (courtesy of Wu, Shah, and Davidson [19]).

C. *Nef polyhedra based algorithms*

Nef polyhedra were introduced first in the mathematical work of Nef [23]. They consist of polyhedra represented by Boolean operations on half-spaces (union, intersection, and complement). These half-spaces are the result of partitioning the space into cells of various dimensions. Each cell is paired with a Boolean label that determines the membership of the cell to the polyhedron. Nef polyhedra have been slightly generalized to Selective Nef complexes [24][25] by considering a larger set of labels for the different cells.

Hachenberger [26] has used Selective Nef complexes to compute the Minkowski sum of polyhedra. He has implemented a new method for decomposing polyhedra into convex parts, based on Nef polyhedra implementation. Then, he has used convex hull based algorithms to compute the pair-wise Minkowski sums and finally, he has used the Nef implementation to compute the union of the pair-wise Minkowski sums. His approach is robust, achieves exactness, and handles all degeneracies by building upon the powerful Nef polyhedra implementation provided in CGAL [21]. However, his algorithm is not efficient because the union step requires a lot of computation time.

The main power of Nef polyhedra theory is that it is quite general, i.e., it can be used to represent non-manifold solids, unbounded solids, and objects having parts of different dimensionality.

III. FIMS ALGORITHM

This section describes our FIMS algorithm that makes it possible to easily implement the Minkowski sum computation for two arbitrarily shaped convex polyhedra. Before we go through further details, let us present the notation and definitions required for the understanding of the next sections.

A. Notation

In the rest of this paper, we consider two convex, closed and 2-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. As the Minkowski sum is equivalent to a morphological dilation, *A* is called polyhedron to be dilated and *B* is called the structuring element. The result of the Minkowski sum of *A* and *B* is denoted polyhedron *S*.

B. Overview and definitions

In FIMS algorithm, we are stating that the sum polyhedron S consists of three categories of facets, which are defined with respect to the polyhedron from which a particular facet of *S* comes.

Proposition

The Minkowski sum of two convex and closed polyhedra *A* and *B* is composed exactly of three types of facets:

- f_A facets that are copies of facets of *A*;
- f_B facets that are copies of facets of *B*;
- At most $e_A * e_B$ facets that result from the Minkowski sum of two non-parallel edges of *A* and *B*.

We will give definitions and names for these three types of facets. To illustrate these principles we also walk through a simple example as long as we define new concepts. Thus, let us consider two polyhedra *A* and *B*, where *A* is a cube and *B* is a tetrahedron. Fig. 2 shows the two polyhedra and their Minkowski sum (polyhedron *S*).

Definition 1

The *translated facets* of the sum polyhedron *S* are facets of *A* translated into another position. These facets are denoted "translated facets" because they are translated copies of all facets of *A* (see Fig. 2).

Definition 2

The *corner facets* of the sum polyhedron *S* are copies of all facets of structuring element *B*, but translated into another position (or by a certain position vector). These facets are denoted "corner facets" because they result from placing the structuring element *B* at each vertex (or corner) of the polyhedron *A* and taking only the facets of *B* that will contribute in the construction of the sum polyhedron *S* (see Fig. 2).

From definitions 1 and 2, and from the commutativity property of Minkowski sum ($A \oplus B = B \oplus A$), it follows that if we consider *B* as the polyhedron to be dilated and *A* as structuring element, the translated facets of $A \oplus B$ are corner facets of $B \oplus A$ and vice-versa.

Definition 3

The *edge facets* of the sum polyhedron *S* are facets resulting from the Minkowski sum of two non-parallel edges, one from the polyhedron *A* and the other from the polyhedron *B*. This is the reason why they are called "edge facets" (see Fig. 2). These are parallelogram-shaped facets.

From Fig. 2, it follows that the number of facets required for the construction of the sum polyhedron S is equal to the sum of the number of facets of the *A* and *B* plus the number of edge facets. The number of edge facets depends on the configuration (orientation, coordinates of the center or origin point) of the two operands of the Minkowski sum.

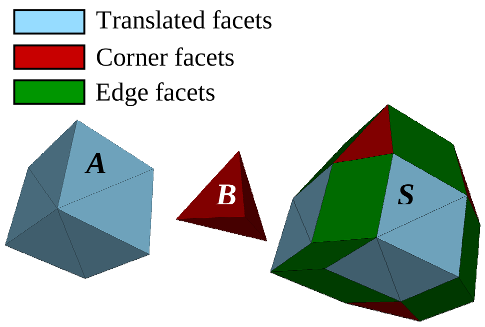

Fig. 2. Polyhedron *A* to be dilated, the structuring element *B*, and the sum polyhedron *S* composed of three categories of facets.

Our proposition is in conformity with the conclusion provided by Bekker et Roerdink [27]. In their work, the authors used a slope diagram based algorithm to compute the Minkowski sum of two polytopes. They showed that the sum polyhedron consists of three types of facets: facets coming from the first operand, facets coming from the second operand, and additional parallelogram facets (that we called edge facets in the FIMS algorithm). The FIMS algorithm constructs the sum polyhedron directly with simple geometric operations and thus eliminates the overhead of passing from three-dimensional domain to the dual space (slope diagram embedded on a sphere) and vice-versa. Therefore, it is more efficient than algorithms working on dual spaces. Moreover, the FIMS algorithm is intended to be generalized for non-convex polyhedra, which is not the case of dual space algorithms.

The FIMS approach aims at constructing the Minkowski sum of two polyhedra *A* and *B* by finding the three types of facets that make the sum polyhedron *S*.

The first concept that we will define is the concept of *"contributing vertex"*, this will lead us to the determination of all translated facets for the sum polyhedron *S*.

We will use a visibility criterion to determine the corner facets after placing *B* on each vertex of *A*.

Finally, the edge facets are found by using two criterions:

• The visibility criterion applied to the facets of *B* allows to find the facets that are visible and those that are invisible with respect to a sight direction defined by an edge of *A*. The frontier between invisible and visible facets of *B* consists of edges of *B* that are candidate for constructing an edge facet in collaboration with the edge of *A* that determined the visibility direction. This frontier is called horizon edges.

• From these candidate horizon edges, we retain only those that satisfy the second criterion of the orientation of the edges normal vectors. i.e., an edge belonging to the horizon edges set will contribute in an edge facet if its normal vector orientation lies between the two orientations for the two normal vectors to facets of *A* sharing the edge that determines the visibility direction for the first criterion.

C. Sum polyhedron construction

We will now show how we can find the different types of facets for the sum polyhedron *S*. For illustrating purposes, we consider the two operands:

• The polyhedron *A* is a sphere mesh with 1600 facets, 3160 edges and 1562 vertices (see Fig. 3.a).

• The structuring element *B* is a SnubDodecahedron with 92 facets, 150 edges and 60 vertices (see Fig. 3.b).

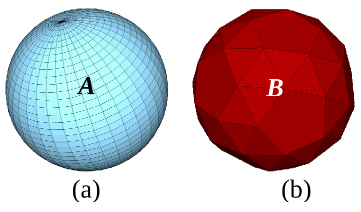

Fig. 3. Operands of the Minkowski sum.

1) Translated facets determination

The translated facets have the same shape as facets of polyhedron *A* but are translated into another position. The question which arises is, starting from *A* facets, what is the amount of translation we should apply to come to translated facets of *S* ?

Since we are working with boundary-representation of polyhedra, we are representing solids only by their external boundaries. Thus, the Minkowski sum of two boundary represented polyhedra A and B is the boundary representation of all points generated by the addition of all pairs of points from A and B.

This leads us to answer our question: the amount of translation that should be applied to each facet of *A* to have the corresponding translated facet of *S* is defined by the vertex of *B* (when placed on the considered facet of *A*) that generates the maximal translation of the considered facet of *A* according to its normal vector direction (which points outwards the polyhedron). In other words, this amount of translation is defined by the vertex of *B* which guarantees that the vertices of the translated facet will lie on the boundary of the sum polyhedron *S*.

Fig. 4 illustrates this principle, we consider a facet for which we will find the corresponding translated facet and we translate it by vectors starting from the origin of the structuring element *B* (the round point inside *B*) and ending at each vertex of *B*. The translated facet has the maximal translation (with respect to the facet's normal *n*) outwards the polygon *A*. The other translated facets will not be considered because they lie inside the sum polyhedron *S*.

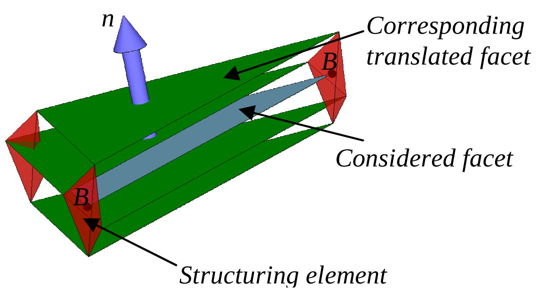

Fig. 4. The concept of translated facets.

In the rest of this paper, the vertex of *B* that defines the translation corresponding to each facet of *A* in order to have the corresponding translated facet will be denoted "the contributing vertex". This concept is the heart of the FIMS algorithm; it allows us to find the positions for all translated facets in the sum polyhedron *S*.

Definition 4

The contributing vertex $v_{k,B}$ corresponding to a facet $f_{i,A}$ of *A* is the vertex -among all vertices of *B*, that generates the maximal translation of the considered facet $f_{i,A}$ according to its normal direction $n_{i,A}$. Formally, the vertex $v_{k,B}$ satisfies:

$$\left\langle v_{k,B} - c, n_{i,A} \right\rangle = \max\left\langle v_{l,B} - c, n_{i,A} \right\rangle \forall l \neq k \qquad (2)$$

Where <.,.> denotes the scalar product of two vectors and *c* is the origin or the center of the structuring element *B*.

The translated facets for our two operands are shown in Fig. 5.

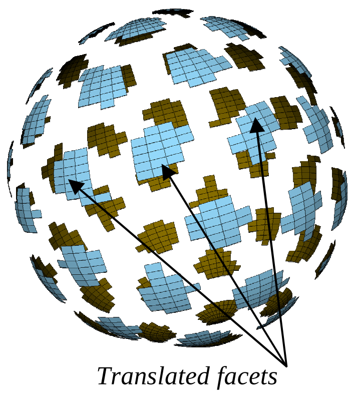

Fig. 5. (a) Translated facets of *S*.

2) Corner facets determination

Corner facets are those of *S* coming from structuring element *B*, they result from positioning *B* on each vertex of *A* and considering only *B* facets that lie on the boundary of the Minkowski sum polyhedron *S*.

Before placing *B* on each vertex $v_{k,A}$ $(k=1,\ldots,v_A)$ of *A* in order to determine the corresponding corner facets of *S*, we will first examine the geometry of translated facets that are copies of the incident facets to $v_{k,A}$. This geometry which is governed by the number of contributing vertices for the facets of *A* that are incident to the vertex $v_{k,A}$, reduces considerably the computation time by avoiding an exhaustive search for corner facets over all vertices of *A*. Thus, we distinguish three geometric configurations:

- The facets of *A* incident to vertex $v_{k,A}$ have **the same contributing vertex**: all corresponding translated facets will be incident to the same vertex of the sum polyhedron *S*. Therefore, there is no corner facet to be added for that particular vertex $v_{k,A}$
- The facets of *A* incident to vertex $v_{k,A}$ have **two different contributing vertices**: the corresponding translated facets will be incident to two vertices of the sum polyhedron *S*. These two vertices are connected by a chain of edges of *S*. Therefore, there is no corner facet to be

added for that particular vertex $v_{k,A}$

• The facets of A incident to vertex $v_{k,A}$ have **at least three different contributing vertices**: one or more corner facets must be added to the sum polyhedron S to fill the hole between the corresponding translated facets.

As it has been said previously, the determination of corner facets is based on a visibility criterion (see Fig. 6) applied to the facets of B according to several sight directions defined by all incident edges to vertex $v_{k,A}$ (a criterion similar to the one used for the convex hull computation in the incremental algorithm [9][10]).

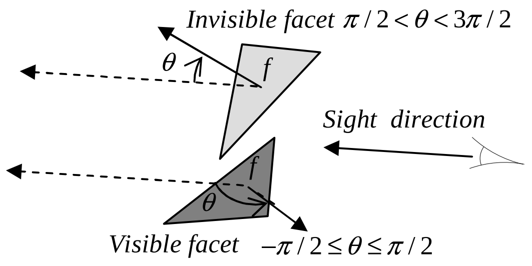
Invisible facet $\pi/2 < \theta < 3\pi/2$
θ
f
Sight direction
f
θ
Visible facet $-\pi/2 \leq \theta \leq \pi/2$

Fig. 6. Visibility criterion for two facets according to a sight direction.

The creation of corner facets for the sum polyhedron S follows the next steps:

1. For each vertex $v_{k,A}$ $(k=1,\ldots,v_A)$, if the incident facets have at least three different contributing vertices, do the next step, otherwise jump to the next vertex (there will be no corner facets to be added for that vertex).
2. Consider all sight directions that correspond to all edges incident and pointing towards the vertex $v_{k,A}$. For each sight direction $a_{j,A}$ calculate the visibility for all structuring element facets corresponding to sight direction $a_{j,A}$, this implies that a facet $f_{i,B}$ is invisible if :

$$\langle a_{j,A}, n_{i,B} \rangle \geq 0 \tag{3}$$

Where $n_{i,B}$ is the normal vector corresponding to the facet $f_{i,B}$. Otherwise the facet $f_{i,B}$ is visible and is not a corner facet.

3. Add the corner facets of S corresponding to that particular vertex $v_{k,A}$: the corner facets of S corresponding to that particular vertex $\mathrm{v}_{\mathrm{k,A}}$ of A are facets of B (after being translated such that its origin c coincides with $v_{k,A}$) that are invisible according to all sight directions related to vertex $v_{k,A}$ (all edges incident to $v_{k,A}$ and pointing towards it).
4. Return to step 1.

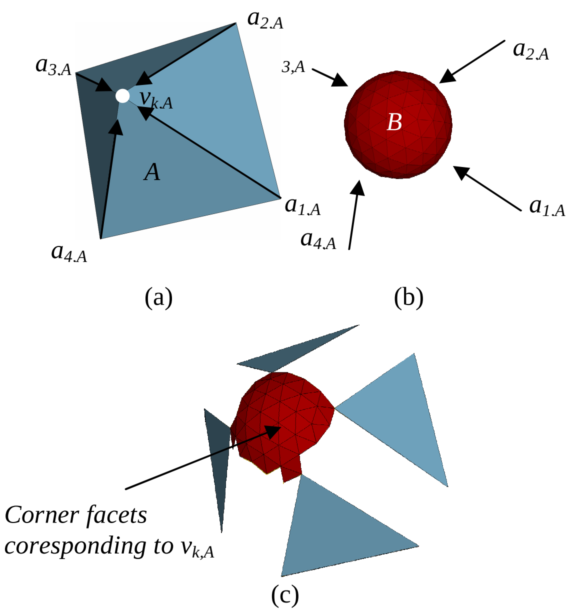

Fig. 7. Determination of corner facets of S corresponding to a particular vertex $v_{k,A}$ of A (hedra shaped) and a sphere as structuring element B.

To illustrate the principles of corner facets and corresponding sight directions for a particular vertex $v_{k,A}$, let us consider the polyhedra A and B shown in Fig. 7. Polyhedron A is hedra shaped, vertex $v_{k,A}$ and sight directions $a_{1,A}$, $a_{2,A}$, $a_{3,A}$, and $a_{4,A}$ are shown in Fig. 7.a. The structuring element B (a sphere mesh) together with sight directions is shown in Fig. 7.b. Finally, corner facets (invisible facets according to all sight directions) of *S* are shown in Fig. 7.c.

For the example we considered first, the corner facets added to *S* together with the previously created translated facets are shown in Fig. 8.

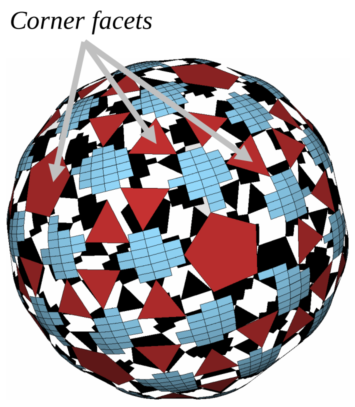

Fig. 8. Corner facets of *S*.

3) Edge facets determination

Similarly to what has been said for the corner facets creation step, in order to create edge facets of *S*, we will first examine the geometry of the two translated facets that are copies of the incident facets to each edge $a_{j,A}$ $(j=1,\ldots,e_A)$. This geometry is governed by the number of contributing vertices for the two facets of *A* that are incident to the edge $a_{j,A}$, and allows us to reduce considerably the computation time by avoiding an exhaustive search for edge facets over all edges of *A*. Thus, we distinguish two geometric configurations:

- The two facets of *A* incident to edge $a_{j,A}$ have **the same contributing vertex**: the two corresponding translated facets will be incident to the same edge of *S*. Therefore, there is no edge facet to be added for that particular edge $a_{j,A}$
- The two facets of *A* incident to edge $a_{j,A}$ have two **different contributing vertices**: the corresponding translated facets will not be incident to the same edge of *S*. Therefore, there is one or more edge facets to be added for that particular edge $a_{j,A}$

The creation of edge facets for the sum polyhedron *S* follows the next steps:

1. For each edge $a_{j,A}$ *(j=1,…,e*$_A$*)*, if the incident facets have two different contributing vertices, do the next step, otherwise jump to the next edge (there will be no edge facets to be added for that edge).

2. Calculate the visibility of all facets of *B* according to the sight direction defined by the edge $a_{j,A}$ (for details see the previous section). The edges $a_{j,B}$ that constitute the frontier between invisible and visible facets of *B* are called horizon edges. These edges are candidates for the creation of edge facets.

Calculate the normal $n_{j,B}$ to each edge $a_{j,B}$ as follows:

$$n_{j,B} = a_{j,A} \times a_{j,B} \qquad (4)$$

The normal vector $n_{j,B}$ will have its direction inversed if it points towards the interior of *B*. In other words, $n_{j,B}$ is the normal vector to a virtual edge facet (not yet added) created from the Minkowski sum of two edges $a_{j,A}$ and $a_{j,B}$

3. Validate or add a virtual edge facet which results the Minkowski sum of edges $a_{j,A}$ and $a_{j,B}$ to the sum polyhedron *S* if and only if:

$$\left\langle a_{1,A}, n_{j,B} \right\rangle \geq 0 \; and \; \left\langle a_{2,A}, n_{j,B} \right\rangle \geq 0 \qquad (5)$$

   Where $a_{1,A}$ and $a_{2,A}$ are two edges of *A* one from each facet incident to the edge $a_{j,A}$ and pointing to a vertex of $a_{j,B}$ (see Fig. 9).

4. Return to step 1.

Fig. 9 illustrates the validation process for edges that will contribute in edge facets creation. Fig. 9.a shows a 2D projection in the plane perpendicular to the two support planes for the two facets incident to edge $a_{j,A}$: there are six horizon edges $a_{j,B}$ $(j=1,\ldots,6)$ with their normal vectors $n_{j,B}$ $(j=1,\ldots,6)$ computed in (4) and two edges $a_{1,A}$ and $a_{2,A}$ pointing to the same vertex of $a_{j,A}$. The edges that satisfy the criterion (5) are those whose normal vectors orientations lie between the two orientations of the two normal vectors of the two facets incident to edge $a_{j,A}$ (edges $a_{3,B}$ and $a_{4,B}$ in Fig. 9.b). Fig. 10 shows the edge facets added to the sum polyhedron S.

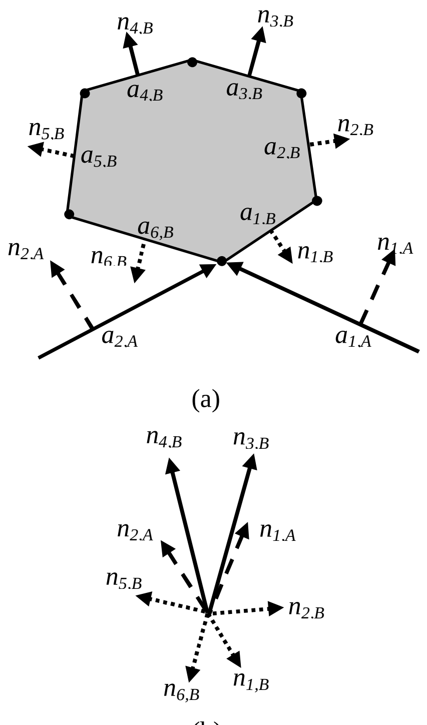

Fig. 9. Validation of horizon edges that will create edge facets of S.

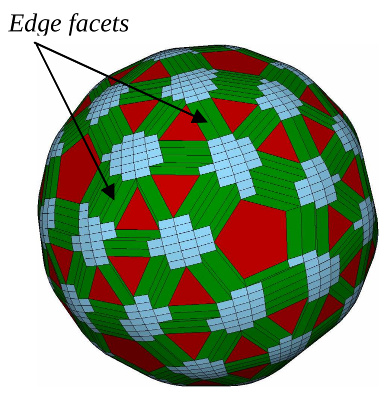

Fig. 10. Edge facets added to *S*.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we will discuss the FIMS algorithm. We will talk about implementation, running time, complexity, generalization to higher dimensions and extensibility to non-convex polyhedra. Some examples of polyhedra that are generated by FIMS algorithm are given too.

A. Implementation and performance

FIMS algorithm has been implemented on a 1 GB RAM, 3.0 GHZ Intel Pentium 4 personal computer. We have used C++ with CGAL (Computational Geometry Algorithm Library) [21] for its implementation and for the computation of the convex hull. Table 1 gives the running times compared to the convex hull approach for several operands *A* and *B*.

TABLE1. MINKOWSKI SUM COMPUTATION RUNNING TIME FOR SEVERAL CONVEX POLYHEDRA.

Operands (# of facettes)		Running time (sec.)	
A	B	**Proposed approach**	Convex hull
Hedra(8)	Sphere(320)	**0.125**	7.657
Hedra(8)	Sphere(1280)	**0.563**	36.844
Cube(6)	Sphere(320)	**0.109**	8.548
Cube(6)	Sphere(1280)	**0.594**	38.345
Sphere(80)	Sphere(320)	**0.578**	36.158
Sphere(320)	Sphere(1280)	**7.063**	687.099

Table 1 shows that FIMS algorithm is faster than the convex hull based algorithms in the CGAL environment. This is justified by the fact that only the vertices which will contribute to the Minkowski sum final result will be treated; thus, the cost of updating the convex hull (determination of visible facets for a vertex, removal of invisible facets, construction of temporary facets at each introduction of a new vertex…) is eliminated and the simplicity of the treatment is increased. Moreover, the FIMS algorithm preserves the facets degrees, which is not the case with the convex hull based approaches (see Fig. 11).

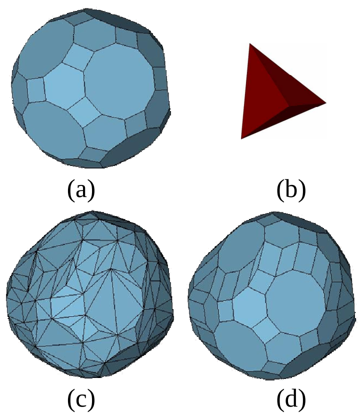

Fig. 11. (a) Polyhedron *A*. (b) Structuring element *B* (a tetrahedron). (c) Sum polyhedron S generated by the convex hull approach. (d) Sum polyhedron *S* generated by FIMS algorithm.

Polyhedra resulting from the Minkowski sum of several convex polyhedra with several structuring elements are given in Fig. 12. From these examples, it is clear that Minkowski sum can be used as a tool for morphing from one shape to another. For more examples, please visit: http://liris.cnrs.fr/hichem.barki/Research/Papers/FIMS_Convex

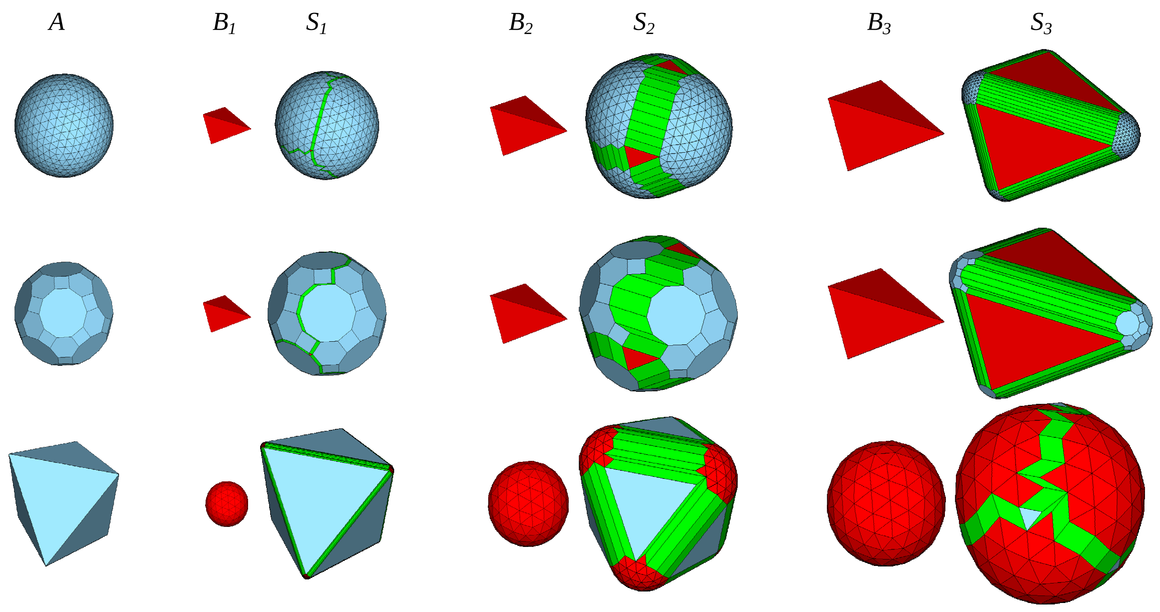

Fig. 12. Objects resulting from the Minkowski sum computation.

A. Complexity

We have seen previously that FIMS algorithm requires three steps. The first step iterates through all facets of A to find the contributing vertices; the second step iterates through all vertices of A to find the corner facets and the last step iterates through all edges of A to find the edge facets. We have computed the complexity of each step and we have used the Euler rule for manifold polyhedra, this has lead us to conclude that FIMS algorithm has a time complexity $O(f_A f_B)$. This result shows that FIMS algorithm is commutative, i.e., $O(f_A f_B)=O(f_B f_A)$, this result is also in accordance with the commutativity property of the Minkowski sum.

B. Generalization to higher dimensions

From the description of FIMS algorithm, it is clear that the construction of the sum polyhedron *S* requires the determination of three types of facets. For translated facets, we used only the scalar product to find the contributing vertex and vector additions to translate *A* facets. For corner facets, we used only vector additions to translate *B* and place it on each vertex of *A*, and the scalar product to find corner facets. For edge facets, we used scalar product to compute the visibility status of *B* facets, cross and scalar products to find horizon edges that will contribute to the construction of edge facets, and Minkowski sum of edges to construct edge facets. Since the vector addition, scalar product, cross product, and Minkowski addition of two edges (which is equivalent to vector addition of all points from the two edges) are concepts that are defined in arbitrary dimensional spaces; we can say that FIMS algorithm can be generalized for any arbitrary dimensional space. This is an important fact since FIMS algorithm can be easily used for other dimensions such as for the case of polygons in two-dimensional space or for higher dimensions.

C. Extensibility to non-convex polyhedra

Our FIMS algorithm is based on a simple idea; an important aspect we must consider is its possible extension to the computation of the Minkowski sum of non-closed and non-convex polyhedra. For non-closed polyhedra, our algorithm can treat this kind of operands by a special treatment of border edges of polyhedra; we are actually working on this case. For non-convex polyhedra, our aim is to compute the Minkowski sum without decomposing them into convex pieces and without passing through the union step.

Figure13 shows that FIMS algorithm is promising for adaptation to non-convex polyhedra, the set of three types of facets generated by FIMS algorithm for non-convex polyhedra is a superset of the facets of the Minkowski sum polyhedron, this superset includes all the Minkowski sum facets plus additional facets that are located inside the resulting polyhedron. The treatment and elimination of these additional facets constitute the direction towards which we are investigating.

V. CONCLUSION AND FUTURE WORK

We have presented a new, fast and general algorithm for the incremental construction of the Minkowski sum of convex polyhedra. The results show that the FIMS approach is faster than convex hull based algorithms. This is justified by the fact that only the vertices which will contribute to the Minkowski sum boundary will be treated. FIMS algorithm can be used for convex objects or to accelerate the calculation of the Minkowski sum for non-convex objects.

We are currently working on the generalization of FIMS algorithm for non-convex polyhedra without passing through the decomposition and union steps. This will be done by treating intersections between the three types of facets of the sum polyhedron we already defined. This will then open a new way to carry out morphological operations aiming to filter big size meshes, in addition to help improving other applications such as robotics, solid modelling, penetration depth estimation, …

VI. Acknowledgment

This work has been partly supported by the Cluster 2 of the Rhône-Alpes region within the LIMA Project.

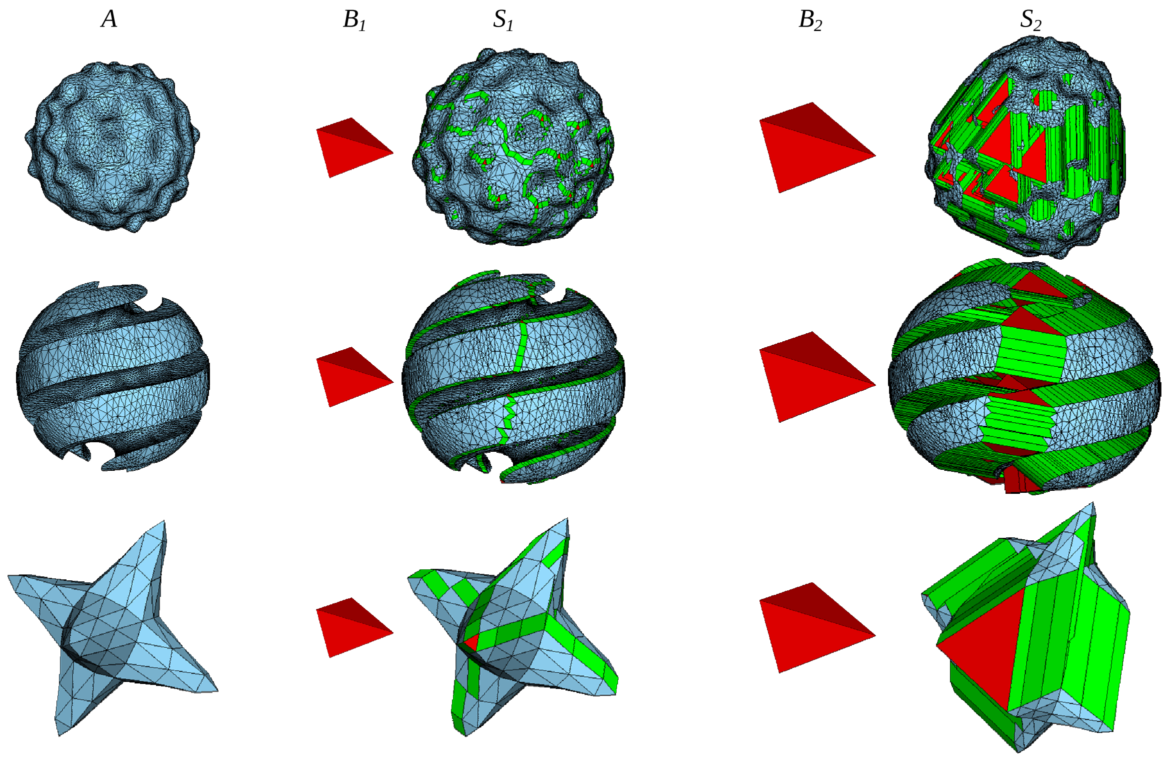

Fig. 13. The set of three types of facets that will form the Minkowski sum polyhedron of some non-convex polyhedra with a convex structuring element.

References

[1] J. Serra. Image analysis and mathematical morphology. Vol. 1, Academic Press, London, 1982.

[2] I.K. Lee, M.S. Kim, and G. Elber. Polynomial/rational approximation of Minkowski sum boundary curves. Graphical Models and Image Processing, Vol. 60, No. 2, 136–165, 1998.

[3] T. Lozano-Pérez. Spatial planning: a configuration space approach. IEEE Transaction on Computers, Vol. C-32, No. 2, 108–120, 1983.

[4] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Fast penetration depth computation using rasterization hardware and hierarchical refinement. Workshop on Algorithmic Foundations of Robotics, 2002.

[5] G. Matheron. Elements pour une théorie des milieux poreux. Masson, Paris, 1967.

[6] P. Soille. Morphological image analysis : principles and applications. 2nd edition, Springer-Verlag, 2003.

[7] D.R. Chand and S.S. Kapur. An algorithm for convex polytopes. Journal of ACM, Vol. 17, No. 1, 78–86, 1970.

[8] J. O'Rourke. Computational geometry in C, 2nd edition. Cambridge University Press, Cambridge, England, 1998.

[9] R. Seidel. Output-size sensitive algorithms for constructive problems in computational geometry. Report TR 86–784, Ithaca, NY: Department of Computer Science, Cornell University, 1986.

[10] M. Kallay. The complexity of incremental convex hull algorithm in Rd. Information Processing Letters, Vol. 17, No. 4, 197, 1984.

[11] F.P. Preparata and S.J. Hong. Convex hulls of finite sets of points in two and three dimensions. Communications of the ACM, Vol. 20, No. 2, 87–93, 1977.

[12] B. Aronov, M. Sharir, and B. Tagansky. The union of convex polyhedra in three dimensions, SIAM Journal on Computing. Vol. 26, No. 6. 1670–1688, 1997.

[13] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. Algorithmica, Vol. 9, No. 6, 518–533, 1993.

[14] D. Halperin. Robust geometric computing in motion. International Journal of Robotics Research. Vol. 21, No. 3, 219-232, 2002.

[15] G. Varadhan and D. Manocha. Accurate Minkowski sum approximation of polyhedral models. Graphical Models, Volume 68, Issue 4, 2006.

[16] B. Chazelle. Convex decompositions of polyhedra. In ACM Symposium on Theory of Computing, 70–79, 1981.

[17] B. Chazelle, D. Dobkin, N. Shouraboura, and A. Tal. Strategies for polyhedral surface decomposition: An experimental study. Computational Geometry, 297–305, 1995.

[18] P.K. Ghosh. A unified computational framework for Minkowski operations. Computers and Graphics, Vol. 17, No. 4, 357–378, 1993.

[19] Y. Wu, J.J. Shah, and J.K. Davidson. Improvements to algorithms for computing the Minkowski sum of 3-polytopes. Computer-Aided Design, Vol. 35, No. 13, 1181-1192, 2003.

[20] J.M. Lien. Point-Based Minkowski Sum Boundary. 15th Pacific Conference on Computer Graphics and Applications (PG'07), 261-270, 2007.

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

[22] E. Fogel and D. Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.05.017

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

[24] M. Granados, P. Hachenberger, S. Hert, L. Kettner, K. Mehlhorn, and M. Seel. Boolean Operations on 3D Selective Nef Complexes: Data Structure, Algorithms, and Implementation. In: Proc. of the 11th Annu. European Sympos. Algorithms (ESA'03), Budapest, Hungary. LNCS 2832, Springer, 654-666, 2003.

[25] P. Hachenberger and L. Kettner. Boolean Operations on 3D Selective Nef Complexes: Optimized Implementation and Experiments. In: Proc. of 2005 ACM Symposium on Solid and Physical Modeling (SPM), Cambridge, MA. 163-174, 2005.

[26] P. Hachenberger. Exact Minkowski Sums of Polyhedra and Exact and Efficient Decomposition of Polyhedra in Convex Pieces. Proc. 15th Annual European Symposium on Algorithms (ESA), LNCS, Springer Verlag, Vol. 4698, 669–680, 2007.

[27] H. Bekker and J.B.T.M. Roerdink. An efficient algorithm to calculate the Minkowski sum of convex 3D polyhedra. Proceedings of the International Conference on Computational Sciences-Part I, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2073, 619-628, 2001.