Minkowski Operations for Satellite Antenna Layout

Jean-Daniel Boissonnat, Eelco de Lange, Monique Teillaud

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Minkowski Operations For Satellite Antenna Layout

Jean-Daniel Boissonnat , Eelco de Lange , Monique Teillaud

N˚ 3070

Décembre 1996

THÈME 2

ISSN 0249-6399

Minkowski Operations For Satellite Antenna Layout

Jean-Daniel Boissonnat[*] , Eelco de Lange[**] , Monique Teillaud[***]

Thème 2 — Génie logiciel
et calcul symbolique
Projet Prisme

Rapport de recherche n ° 3070 — Décembre 1996 — 20 pages

Abstract: Satellite layout is a very hard task because available space for the equipments is small and the physical constraints on the layout are strong. In this article, we show how some physical layout constraints can be modeled geometrically and how Minkowski operations can be used to place equipments, and in particular antennas. Since antennas are supposed to be placed onto a satellite wall, search space is two-dimensional. We discuss an algorithm that efficiently calculates only the (planar) part we need of the admissible space and describe how we implemented this.

Key-words: satellite layout, Minkowski operations, degeneracies, numerical stability

(Résumé : tsvp)

This research was partially supported by the ESPRIT IV LTR Project No. 21957 (CGAL).

[*] Jean-Daniel.Boissonnat@sophia.inria.fr

[**] INRIA and Matra Marconi Space, Dept. AOII, 31 Avenue des Cosmonautes, Z.I. du Palays, 31402 Toulouse CEDEX 4, FRANCE, Eelco.de_Lange@sophia.inria.fr

[***] Monique.Teillaud@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA ANTIPOLIS Cedex (France)
Téléphone : 04 93 65 77 77 – Télécopie : 04 93 65 77 65

Opérations de Minkowski pour l'aménagement de satellites

Résumé : L'aménagement de satellites est une tâche difficile car l'espace disponible pour les équipements est réduit et les contraintes physiques sur l'aménagement sont fortes. Dans cet article, nous montrons comment certaines contraintes physiques d'aménagement peuvent être modélisées géométriquement et comment des opérations de Minkowski peuvent être utilisées pour placer les équipements, en particulier les antennes. Puisque les antennes doivent être placées sur une paroi du satellite, l'espace de recherche est bidimensionnel. Nous proposons un algorithme qui calcule efficacement uniquement la partie indispensable de l'espace admissible et nous décrivons l'implantation de cet algorithme.

Mots-clé : aménagement de satellites, opérations de Minkowski, dégénérescences, stabilité numérique

1 Introduction

The layout of a satellite is a very hard problem in practice. Because a satellite is tried to be constructed as small as possible, the layout of the equipments is extremely constrained. One problem that is reputedly difficult to solve is the layout of the antennas and observation instruments on the outer walls of a satellite.

For instance, many feasibility studies are done to verify whether the specified set of antennas and instruments can be fitted on the outer walls. All along the development phase, specifications often change so one wants to know as soon as possible if a possibly existing layout is still realizable (or if not, how it should be modified) when the size or type of an equipment is changed or when an equipment is added. Typically, manual tools supplying placeability information are the most useful for these types of problems but automatic placement facilities are useful also.

An antenna, in general, consists of a parabolic reflector, a source that either emits or receives a signal, and their supporting structures. It also has a field of vision (from now on written as FOV), that can be shaped more or less as a cone. A FOV must not be obscured (not even partly) by an obstacle, which means that this cone is not intersected. An antenna may be mobile, which means that it can rotate along one or two predefined axes. There also exist deployable antennas which are folded in at the moment of launch and which are deployed once the satellite has entered into orbit.

We assume that we start the layout from scratch, adding the equipments one by one. Given an antenna, we want to place it *onto* a specified exterior wall of the satellite. Only when this isn't possible, we try to place it further from the wall but we try to stay as close as possible, to minimize the size and weight of the structure needed to support it. In practice, first the placements onto the wall are considered, then the placements at a certain distance of the wall, and so on. As well as the distance from the wall, we can also discretize the rotation of the antenna when we want to take it into account. So, basically, at every stage we only consider translations of the antenna in a plane, and the search space is contained in $\mathbb{R}^2$.

First, we will define the antenna layout problem and model it geometrically by defining an antenna's admissible space that is contained in a plane. We show how we can use planar sections of Minkowski differences between polyhedra to calculate this admissible space. Then, we present our algorithm that directly calculates this section and discuss the handling of degeneracies and numerical precision problems. Finally, we show some experimental results, and we compare them with the experimental results obtained by an algorithm that calculates a Minkowski difference in $\mathbb{R}^3$ and then finds the interesting planar section.

2 Definitions and Properties

The objects we handle are contained in $\mathbb{R}^3$. The parts of the equipments and the environment are modeled as convex polyhedra. Polyhedra are written as capitals, while planes, polygonal and polyhedral regions are written using calligraphic letters.

Layout problems are known to be NP-hard: placing polygons in a rectangle of minimal length is and NP-hard problem [DM95]. In practice, fortunately, we are satisfied with a reasonable solution. By placing the objects one by one using a heuristic, one should be able to find good solutions in a reasonable time. However, while it is rather easy to find a good heuristic for placements of large numbers of relatively simple objects, the antenna layout problem consists of a small number of highly constrained objects and an efficient heuristic seems to be much harder to find.

We call $\mathcal{O}$ the set of *obstacles*, consisting of the satellite structure plus all equipments that have been placed on it at a certain moment. $\mathcal{P}$ is the equipment we are currently placing, and the set of all allowed translations that place $\mathcal{P}$ so that it does not collide with $\mathcal{O}$ (the *admissible space*) is written as $\mathcal{A}(\mathcal{O}, \mathcal{P})$. The unions of all physical parts and of all FOVs of an equipment or obstacle $\mathcal{S}$ are denoted as $\mathcal{S}_P$ and $\mathcal{S}_F$ respectively.

$P^C = \{p \in \mathbb{R}^3 \mid p \notin P\}$ is the complement of P. The symmetric of P is $\ominus P = \{-p \mid p \in P\}$.

The Minkowski sum between two polyhedra A and B is notated as $A \oplus B$. It is defined as $A \oplus B = \{a + b \mid a \in A, b \in B\}$. Note that the Minkowski sum is distributive and commutative over the union, so $(A_1 \cup A_2) \oplus B = B \oplus (A_1 \cup A_2) = (A_1 \oplus B) \cup (A_2 \oplus B)$.

3 Geometric Modelization

We shall describe how this problem can be modeled geometrically and we define the Gaussian diagram that allows us to calculate the admissible space efficiently.

3.1 Admissible Space

Given two convex polyhedra O and P, translating P by a vector $\vec{t}$ will make it intersect O when there exist two points $o \in O$ and $p \in P$ such that $o = p + \vec{t}$, so, $\vec{t} = o - p$. Consequently, the set of all translations for P such that it intersects O is defined as $\{o - p \mid o \in O, p \in P\}$, which is exactly the Minkowski difference $O \ominus P$. Since O and P are convex, $O \ominus P$ also is convex. The boundary of $O \ominus P$ represents

all translations that bring P into contact with O without intersecting its interior, while its complement represents all "free" translations for P.

Consequently, if we want to place P without colliding with O, we must pick a translation $\vec{t} \in (O \ominus P)^C$. Since we want to allow contacts, we consider the boundary of the Minkowski difference to be part of the admissible space. If we want to place $\mathcal{P} = \{P \in \mathcal{P}\}$ among a set $\mathcal{O} = \{O \in \mathcal{O}\}$ of obstacles, we need to calculate $(\mathcal{O} \ominus \mathcal{P})^C$ where $\mathcal{O} \ominus \mathcal{P}$ is defined as $(\cup_{O \in \mathcal{O}} O) \ominus (\cup_{P \in \mathcal{P}} P)$ (see figure 1). Using the the distributivity of the Minkowski sum, this can be rewritten as $\cup_{O \in \mathcal{O}, P \in \mathcal{P}} O \ominus P$.

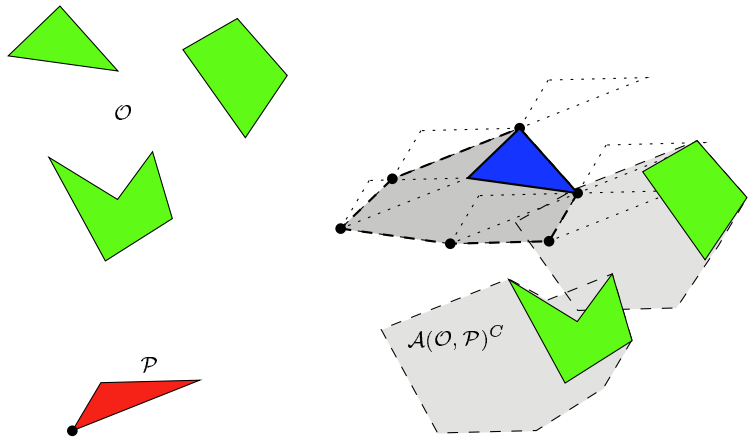

Figure 1: The admissible space for polygonal region $\mathcal{P} = \{P\}$ with respect to the polygonal region $\mathcal{O}$ is the complement of the polygonal region drawn in light gray.

This handles the physical parts of the equipment. FOVs can almost be treated as physical parts, because they may not intersect with "real" parts; the only difference is that two FOVs may freely intersect. Taking only in account parts that interfere, we can notate this as follows:

$$\mathcal{A}(\mathcal{O},\mathcal{P}) = (\ \ (\mathcal{O} \ominus \mathcal{P}_P) \cup (\mathcal{O}_P \ominus \mathcal{P}_F)\ \ )^C \tag{1}$$

We can rewrite this as the complementary of the union of $O \ominus P$ for all pairs (O, P) such that O is any element of $\mathcal{O}$ and $P \in \mathcal{P}_P$ or $O \in \mathcal{O}_P$ and $P \in \mathcal{P}_F$.

All the equipments and obstacles are decomposed as unions of convex polyhedra, so that in the sequel, P and O will always represent convex polyhedra.

We want to place the equipment onto a wall. Let us assume without loss of generality that this wall is horizontal and define a coordinate system (O, X, Y, Z)

where the Z-axis is perpendicular to the wall and the X- and Y- axes as well as O are contained in it. Since we want to position the equipment $\mathcal{P}$ at a given height z, we are only interested in the translations $\vec{t} \in \mathcal{A}(\mathcal{O}, \mathcal{P})$ that position $\mathcal{P}$ at this height. All these translations are contained in a horizontal plane $\mathcal{Z}$ defined by $Z = z$ in the admissible space so we are actually looking for the intersection of $\mathcal{A}(\mathcal{O}, \mathcal{P})$ with $\mathcal{Z}$.

3.2 Gaussian diagrams of Convex Polyhedra

We shall define a mapping of a convex polyhedron P onto the unit sphere of $\mathbb{R}^3$ denoted S^2 that is called the normal or Gaussian diagram (see [LP83, GS87, AB89, Avn89, CEGS93]), and which can be used to calculate a Minkowski sum and bound its complexity.

For each facet $f \in P$, let n_f denote its normal vector. The Gaussian diagram P' on the unit sphere S^2 consists of the following elements (we intentionally use another terminology to avoid confusion between polyhedral elements and elements in the Gaussian diagram):

- Its *nodes* are the vectors n_f.
- Its *arcs* are great circular arcs, each being the locus of the normal vectors of all planes supporting P at one of its edges.
- Its *cells* are regions, each being the locus of the normal vectors of all planes supporting P at one of its vertices.

So, there is a one-to-one correspondence between all elements of P and those of P' (see figure 2) if P has no coplanar facets.

When a polyhedron has coplanar facets, the nodes associated with such coplanar adjacent facets coincide and the arcs and cells between such nodes reduce to points. Instead of trying to handle this type of degeneracy in the algorithm, we can work around it by eliminating coplanar adjacent facets in a preprocessing step. This can be achieved by merging possible coplanar facets in the input polyhedra.

So, from now on, we will assume that a convex polyhedron has no coplanar facets.

3.3 Gaussian Diagrams and Minkowski Sums

Let A and B be two convex polyhedra in general position and M' be the superposition of their Gaussian diagrams. It is known that M' is the Gaussian diagram of $M =$

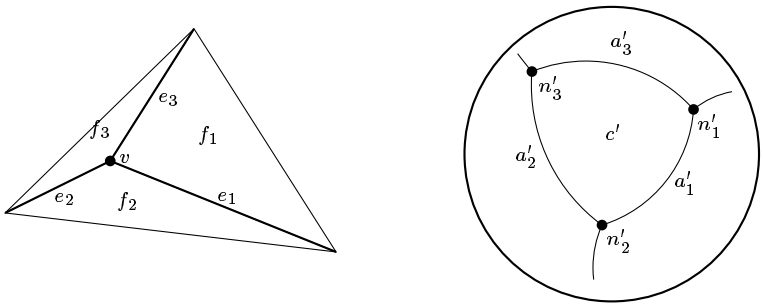

Figure 2: A convex polyhedron and its Gaussian diagram.

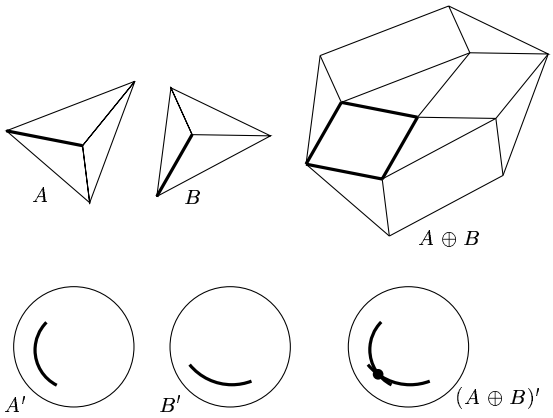

Figure 3: The intersection of two arcs of the Gaussian diagrams corresponds with an aa-facet of the Minkowski sum.

$A \oplus B$ [LP83, GS87]. Any facet of M corresponds either to a node coming from A' or B' or to an intersection of two arcs of A' and B' (see figure 3).

If we assume that A and B are in general position, we distinguish three types of nodes of M' associated with facets of M, all uniquely defined by an element of A' and an element of B':

- Node $n_{A'} \in A'$ is contained in cell $c_{B'} \in B'$. We call this a *nc*-node.
- Symmetrically, a node $n_{B'} \in B'$ that is contained in a cell $c_{A'} \in A'$ is called a *cn*-node.
- An intersection of an arc $a_{A'} \in A'$ with an arc $a_{B'} \in B'$ is an *aa*-node.

When allowing degeneracies, we can encounter three other cases:

- Node $n_{A'} \in A'$ is contained in arc $a_{B'} \in B'$. We call this a *na*-node.
- Symmetrically, a node $n_{B'} \in B'$ that is contained in arc $a_{A'} \in A'$ defines a *an*-node.
- When two nodes $n_{A'} \in A'$ and $n_{B'} \in B'$ coincide, they define a *nn*-node.
- Arcs $a_{A'} \in A'$ and $a_{B'} \in B'$ overlap.

Knowing the elements of A and B that are associated with the two elements defining $n_{M'} \in M'$, we can construct the associated facet $f_M \in M$: it is the Minkowski sum of the two elements of A and B.

3.4 Worst-case complexity

We will bound the worst-case size of the intersection of a Minkowski sum with a plane.

It is known that, given two convex polyhedra A and B of sizes m and n respectively, the size of their Minkowski sum $A \oplus B$ is $\Theta(mn)$ in the worst case. This result is easily verified by considering the superposition of the Gaussian diagrams of the two polyhedra: arcs of A' and B' can intersect $\Theta(mn)$ times, which means that as many *aa*-facets exist in the Minkowski sum. A new theorem, more relevant for our application, shows that, in the worst case, a section of $A \oplus B$ has the same worst-case size as $A \oplus B$ itself.

Theorem 3.1 *Given two convex polyhedra A and B, the intersection of $A \oplus B$ with any plane has a complexity of $\Theta(mn)$ in the worst case.*

Proof It is clear that the upper bound is quadratic, so we only have to show an example which is actually quadratic. We construct a polyhedron A by intersecting two open polyhedral vertical coaxial cones such that a pattern of m zigzag edges is formed at the place where their boundaries intersect. B is formed by taking the convex hull of a very small horizontal segment and of n points on a small arc of a vertical circle centered at the midpoint of the segment. Then $A \oplus B$ is quadratic (see the Gaussian diagrams on figure 4(b)).

Let us denote u (resp v) the upper (resp lower) vertex of B, and take the midpoint of the horizontal axis of B as the origin on B. If any edge $[p, q]$ of the zigzag pattern on A is sufficiently high with respect to B, then point $p + v$ of $A \oplus B$ will be above point $q + u$. So all the points of $[p, q] \oplus w$ defined as the sum $p + w$ of the upper endpoint p of $[p, q]$ and a vertex w of B (except the two vertices of the axe of B) are above all the points defined as the sum $q + w$ of the lower endpoint q of and a vertex v of B. Which implies that an horizontal plane $\mathcal{Z}$ passing between $p + v$ and $q + u$ will intersect all the $O(mn)$ edges of the form $[p, q] \oplus w$ of $A \oplus B$.

□

4 A Direct Method to Compute a Minkowski Slice

While the boundary of the polyhedron $O \ominus P$ represents all translations that bring P into contact with O without intersecting it, the boundary of the polygon slice $(O \ominus P) \cap \mathcal{Z}$ is the subset of all such translations that position P at a given distance from the wall. It corresponds with the displacement by translations that P makes around O when it keeps the same height and stays in contact with O.

The sequence of edges of $(O \ominus P) \cap \mathcal{Z}$ is given by the sequence of facets of $O \ominus P$ intersected by $\mathcal{Z}$. As $O \ominus P$ is convex (P and O are convex) if we project the normal vectors to this sequence of faces onto the horizontal plane, we notice that the sequence of polar angles of the projected vectors is monotonic. In fact, $(O \ominus P) \cap \mathcal{Z}$ is represented on $(O \ominus P)'$ as a path in the superposition of the Gaussian diagrams of O and $\ominus P$. This path is monotonic in θ, if θ denotes the polar angle of the projected points of the sphere onto the horizontal plane.

This expresses only the fact that, since P and O are convex, when P is moving by translation at a fixed height while keeping a contact with an obstacle O, the contact point follows a path on O describing a continuous sequence of adjacent facets, edges and vertices touched by P. If we project the normal vectors to this sequence of faces of O onto the horizontal plane, the sequence of polar angles of the projected vectors is monotonic. The same holds on P.

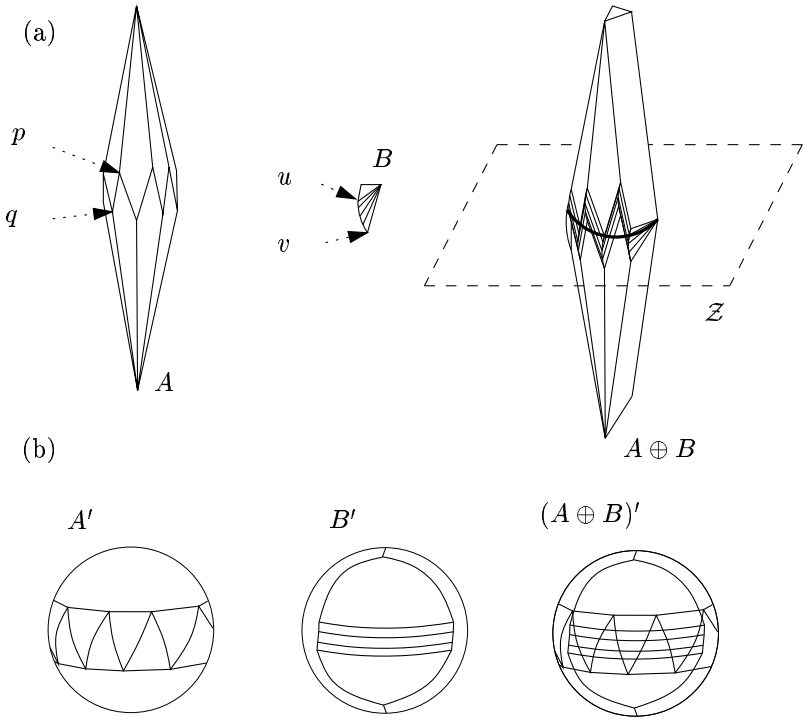

Figure 4: An example of a planar cross-section of a Minkowski sum with quadratic complexity.

4.1 Description of the Algorithm

We present here an algorithm that directly calculates the intersection of the Minkowski sum of two convex polyhedra A and B with a given horizontal plane $\mathcal{Z}$ by calculating the monotonic walk we just introduced, through the superposition of their Gaussian diagrams. We denote $M = A \oplus B$. It is assumed that $\mathcal{Z}$ intersects the interior of the Minkowski sum, because the output polygon is empty otherwise. We assume for the moment that $\mathcal{Z}$ does not contain any vertices of M and that no degeneracies occur in the superposition of the Gaussian diagrams of A and B; the degeneracies will be examined in section 4.2.

There exists a path through M' (being the superposition of A' and B') that visits the arcs and nodes associated with all edges and facets of M that intersect $\mathcal{Z}$ (and *only* those elements). We don't want to calculate the entire superposition and only construct the arcs and nodes of M' we encounter along the way. We denote by K_s the complexity of the output $(A \oplus B) \cap \mathcal{Z}$.

4.1.1 How to get from a node to an arc

Assume that our walk has arrived at a certain node $n_{M'} \in M'$, associated with facet $f_M \in M$. Because f_M cuts $\mathcal{Z}$ and it is convex, exactly two edges of f_M cut $\mathcal{Z}$, one by whose associated arc we arrived at $n_{M'}$ and one by whose associated arc we will advance to the next node of the Gaussian diagram. We must determine which arc $a_{M'} \in M'$ incident to $n_{M'}$ to follow in order to get to the next node of M' (it is either an arc from A' or from B') and we must also determine which cell it lies in: it is an arc of A' that is contained in a cell of B' or vice versa.

Note however that M' does not contain the information needed to determine if an edge corresponding with an arc from M' cuts $\mathcal{Z}$ or not, so we must look back at A and B to obtain this information: the edge of M is the sum of an edge of A and a vertex of B (or the contrary), so we can compute the z coordinate of its two endpoints and compare them with the height of $\mathcal{Z}$.

Let us first consider the case when $n_{M'}$ is a *cn*-node. $n_{M'}$ is given by a node of B' contained in a cell of A'. Then to find the arc that must be followed, we have to test all arcs of B' incident to the node, and it can take up to $O(n)$ time to identify the arc that must be walked. The case of a *nc*-node is symmetric and takes $O(m)$ time. We might visit all *cn*- and *nc*-nodes, but since arcs are incident to two nodes and the walk is monotonic, any arc from A' or B' is at most considered twice on the visited *cn*- and *nc*-nodes during the whole walk.

The case of *aa*-nodes is easier to analyze, since *aa*-nodes are incident to four arcs, and we visit at most $O(K_s)$ such nodes.

Consequently, $O(m + n + K_s)$ time is spent in total determining which arc to walk next.

4.1.2 How to get from an arc to a node

Without loss of generality, suppose that we walk along arc $a_{M'}$ supported by $a_{A'} \in A'$, through cell $c_{B'} \in B'$. If $a_{A'}$ intersects an arc $a_{B'}$ incident to $c_{B'}$, the next node $n_{M'}$ is an *aa*-node, defined by $a_{A'}$ and $a_{B'}$. Otherwise, we encounter a *nc*-node $n_{M'}$, defined by $c_{B'}$ and the endpoint of $a_{A'}$ in the direction of the walk (see figure 5).

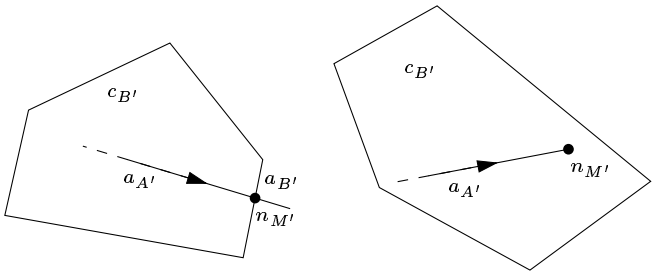

Figure 5: Identifying the next node $n_{M'} \in M'$. Arc $a_{A'}$ is walked through cell $c_{B'}$. (left) an *aa*-node. (right) a *nc*-node.

Notice that the boundary of $c_{B'}$ can be decomposed in two subsequences of arcs, both being θ-monotonic as our path. Consequently, while following arc $a_{A'}$, we can follow at the same time the two sequences of arcs of the boundary of $c_{B'}$ that are respectively above and below $a_{A'}$, until we reach either an intersection between $a_{A'}$ and an arc $a_{B'}$ or the endpoint of $a_{A'}$.

In the case we find an endpoint $n_{A'}$, we will follow a new arc of A' and at the same time keep on traversing, starting from the current arcs, the ordered subsequences of arcs of $c_{B'}$. See figure 6(a).

In the case we reach an intersection between $a_{A'}$ and $a_{B'}$ we either now follow $a_{B'}$ through a cell $c'_{A'}$ and go on symmetrically. Or we keep on following $a_{A'}$ and we leave cell $c_{B'}$. Then we only have to remember the arcs of the two subsequences of the boundary of $c_{B'}$ at that point. Indeed, if the path re-enters the same cell $c_{B'}$ later with another arc $a'_{A'}$, since the path is monotonic, we will examine the two

subsequences, starting from these remembered arcs, until we find the two arcs just above and below $a'_{A'}$, and then we go on as for $a_{A'}$. See figure 6(b).

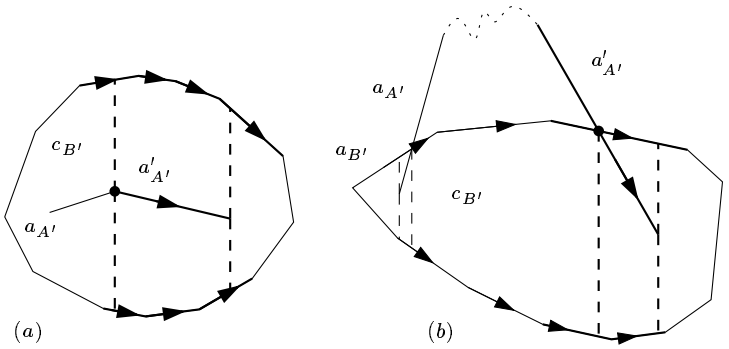

Figure 6: Following the boundary of $c_{B'}$ (a) after an endpoint of $a_{A'}$ (b) after an intersection between $a_{A'}$ and $a_{B'}$

$O(K_s)$ arcs of M' are walked on the path, $O(K_s)$ nodes are computed. By following the boundaries of the cells of A' and B' crossed by the path, an arc of A' or B' can be examined once for each node of M' that ends in one of the two cells incident to it, or intersects it, in which case the cost belongs to the cost of constructing node of M'; an arc of A' or B' can in addition be examined at most twice to locate the entry point of the path into each of its two incident cells. So in total, $O(m + n + K_s)$ time is spent.

4.1.3 Finding an initial arc

The idea for starting the path is to find a first arc whose associated edge of M is cut by $\mathcal{Z}$, belonging to a *shadow* of M. The shadow of M is the boundary of the orthogonal projection of M onto a plane. We will consider a vertical plane: this will ensure that the shadow of M is cut by $\mathcal{Z}$. On M', the chosen shadow of M is associated with the sequence of arcs, cells and nodes of M' cut by a given vertical plane containing the center of the sphere (the vertical direction on the sphere being given by the vector normal to the horizontal plane $\mathcal{Z}$).

Let us consider the great circle of the sphere contained in this vertical plane. Starting with the cell of A' associated with the highest vertex of A, we can find the arcs of A' intersected by the great circle by following the arcs incident to all the cells cut by this circle (see figure 7(a)). In this way, we directly obtain the ordered

sequence of the arcs of A' cut by the great circle. The same process is applied to B. Both arc-lists are associated to a sequence of connected edges from A and B respectively describing the shadow of M from the direction normal to the great circle. The two lists are traversed at the same time, in such a way that the order in which the arcs cut the great circle is preserved, so we get an ordered list of arcs of M' associated to the shadow of M in $O(m+n)$ time.

In fact we can stop as soon as we encounter an arc corresponding in M with an edge intersected by plane $\mathcal{Z}$: without loss of generality, assume that this arc is an arc $a_{A'}$ belonging to A', associated with edge $[p,q]$ of A (see figure 7(b)). It is contained in a cell of B' associated with a vertex v. The edge $[p,q] \oplus v$ intersects $\mathcal{Z}$ if and only if its endpoints $p+v$ and $q+v$ lie on both sides of $\mathcal{Z}$.

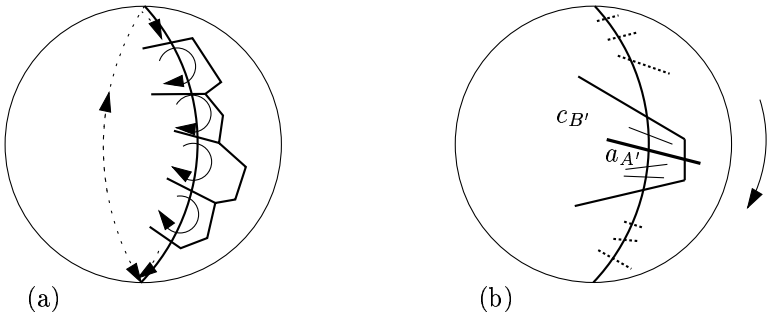

Figure 7: Finding a node $n_{M'}$ to start the walk with. (a) Constructing a sorted shadow-list in linear time. (b) Determining an arc whose associated edge cuts $\mathcal{Z}$.

4.1.4 Global Complexity

Summing up, a total time of $O(m+n+K_s)$ is needed to run the algorithm, where K_s is the size of the output polygon. This is better than the convolution algorithm because K_s is at most equal to K.

4.2 Handling Degeneracies

We choose to properly handle the degeneracies. Remember that we decided to pre-process the input polyhedra to remove adjacent coplanar facets (Section 3.2).

As said in Section 3.3, the following degeneracies can be encountered while constructing $A \oplus B$:

1. a node of A' might be contained in an arc or coincide with another node from B' or inversely, so we have to be able to handle *na-*, *an-* and *nn*-nodes.

2. two arcs overlap: we have to make sure that if we walk one arc, we walk the other one at the same time.

The detection of the degeneracies occurs when getting from an arc to a node (Section 4.1.2).

In fact, the test whether two arcs intersect also checks if the intersection occurs at an endpoint of one (or both) arcs. When a *na-* or a *an*-node is detected in this way, we only have to add the arc in the ordered list of arcs incident to the node, and find the arc whose associated edge is cut by $\mathcal{Z}$, which will be followed next. When a *nn*-node is detected, we traverse the two lists of arcs incident to the nodes and walk the arc whose edge is cut by $\mathcal{Z}$.

Overlapping arcs are also detected when we test for their intersection. Then we compute which endpoint will be encountered first in the walk, and we treat the obtained *na-*, *an-* or *nn*-node as above.

4.3 Numerical Stability

Another problem often encountered when implementing geometric algorithms is the lack of numerical precision in geometric tests. This causes wrong decisions to be made leading to inconsistencies in the state data of the algorithm which causes it to malfunction.

Problems are encountered for example when testing for intersection between arcs when an endpoint of one of the arcs is nearly contained in the supporting plane of the other arc. So in fact, numerical problems occur in nearly degenerate cases.

The basic test for our algorithm is *which_side*, that tests on which side a node lies with respect to the supporting plane of an arc. All tests used in the direct algorithm can be written using one or more such tests:

- determining whether an arc contains a node: does the node lie on the plane that contains the arc, and does it lie between its two endpoints?

- testing for intersection between two (great circular) arcs: it must be verified whether for both arcs, the two endpoints lie at opposite sides of the supporting plane of the other arc.

- determining whether two (parallel) arcs overlap: do both endpoints of one arc lie on the plane defined by the other arc and is any endpoint contained in the other arc?

- determining the order between arcs from different Gaussian diagrams that are incident to a *nn*-node.

In fact, the *which_side* test can be written as the determinant of a matrix of size three. More precisely, let n and a be respectively the node and the arc to test. n_1 and n_2 will denote the endpoints of a.

$$which\_side(n, n_1, n_2) = sgn \begin{vmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ z & z_1 & z_2 \end{vmatrix}$$

where (x, y, z) (resp (x_i, y_i, z_i)) are the coordinates of n (resp n_i).

If the normals of the facets of the polyhedra are not given, they can be obtained from three vertices of the corresponding facet: if n is normal to a facet with vertices p, q, r appearing in counterclockwise order on the boundary of the facet, n is given by

$$n = \frac{(q-p) \times (r-q)}{||(q-p) \times (r-q)||}.$$

However, by calculating this, we would introduce an error. So instead of calculating n, we represent it symbolically using p, q, r.

Note that, for the computation of the sign of the determinant, it is useless to normalize n. n is proportional to

$$\left(\begin{vmatrix} y_q - y_p & z_q - z_p \\ y_r - y_q & z_r - z_q \end{vmatrix}, \begin{vmatrix} x_r - x_q & z_r - z_q \\ x_q - x_p & z_q - z_p \end{vmatrix}, \begin{vmatrix} x_q - x_p & y_q - y_p \\ x_r - x_q & y_r - y_q \end{vmatrix} \right)$$

So $which\_side(n, n_1, n_2)$ is given by the sign of a determinant of a 3×3 matrix whose entries are 2×2 determinants.

$which\_side(n, n_1, n_2)$ answers whether a node n is lying on the supporting plane of an arc n_1n_2. Then, to know whether n is lying on the arc, the test reduces in fact to comparing the signs of the 3×3 determinants of the vectors $(n, n_1, q-p)$ and $(n, n_2, q-p)$, where $[p, q]$ is the edge common to the facets normal to n_1 and n_2. These determinants yield less restriction on the number of bits that can be allowed than the *which_side* test.

We want to make the direct method numerically stable, so, to get rid of the numerical problems we encountered, we decided to implement those tests using exact arithmetics. The determinant method described by Avnaim et al. in [ABD+94, ABD+95] can calculate the determinant sign of matrices of size three without error. If the coordinates of the vertices of the input polyhedra are represented by b bit

integers, the 2×2 determinants appearing in the *which_side* test are represented by at most $2b + 1$ bit integers. Then the exact determination of the sign of the 3×3 determinant will need $(2b+1)+1$ bits (in practice the implementation requires $(2b + 1) + 2$ bits). A standard IEEE double allows exact representation of 53-bit integers, so we can use 25-bit integers to represent our data, which is enough in practice.

5 Experimental Results

To compute the admissible space, we calculate all sections of Minkowski sums for pairs (O, P) of polyhedra described in 3.1. Then we calculate the union of the resulting set of polygons, the complement of which is the wanted admissible space.

The implementation of our program allows the equipments to be placed one by one on a horizontal plane $\mathcal{Z}$. The user can request the equipment to be moved upwards or downwards, in this case $\mathcal{Z}$ is modified and the corresponding horizontal section of the three dimensional admissible space will be calculated again. Rotations can also be applied, but in this case the whole admissible space is modified and must be recomputed, not only its section by $\mathcal{Z}$.

We have tested the program on a realistic although fictional model of a satellite called PRISMESAT and present here the series of placements as well as the time it took to determine the Minkowski sums at each stage. We implemented a simple bitmap algorithm to calculate (the complement of) the union. The time it took to calculate the union of the polygons using this bitmap algorithm is negligeable and is not taken into account. The tests have been performed on a SUN SPARCstation20 running SunOS.

Figures 8, 9 give the sequence of placements performed. Table 1 gives the performances of the algorithm for all stages of the layout scenario. Time is measured in milliseconds CPU-time, and both total and average times are given, the average being calculated on all Minkowski sums computed.

We compare here the practical performances of our algorithm to compute the section of a Minkowski sum, with the performances of the algorithm proposed by Guibas and Seidel [GS87]. The last algorithm uses a topological sweep-line algorithm to determine all intersections of the superposed Gaussian diagrams of two convex polyhedra, constructing the Minkowski sum along the way. This is done in $O(m + n + K)$ time, where m and n are the sizes of the input polyhedra and K the size of the resulting Minkowski sum. Then the time to slice the Minkowski sums by plane $\mathcal{Z}$ is added to the total time.

As can be seen, even though the direct method uses the exact evaluation of signs of determinants which is more costly than standard arithmetic, and is therefore much more robust than the convolution method, it is a lot faster.

In fact we also implemented the method that consists, for computing the Minkowski sum, in computing all vertices $v_A + v_B$ for all mn pairs of vertices $v_A \in A, v_B \in B$ and then calculating their convex hull [AB89, Avn89] with the Quick Hull algorithm [BDH93]. The performances were even poorer.

Equipment	N_p	N_v	N_m	K	N_s	K_s
INIT	54	837	-	-	-	-
ESRI-1	3	32	162	30.8	44	11.0
ESRI-2	3	32	171	30.3	46	10.8
HORN-7-HST	7	188	420	51.3	146	11.9
SVS-ANTENNA	10	727	670	144.5	257	22.3
VOGO	10	296	770	91.5	161	17.7
TVGAVO-ENS	4	70	348	62.3	107	19.3
Total	*91*	*2082*	*2541*	*86.8*	*761*	*17.6*

	Convolution		**Direct Slice**	
	Average	*Total*	*Average*	*Total*
WALL	-	-	-	-
ESRI-1	113.2	18339.1	26.6	4316.5
ESRI-2	121.5	20772.3	25.7	4399.8
HORN-7-HST	161.3	67738.7	39.5	16599.3
SVS-ANTENNA	648.3	434371.9	85.8	57514.4
VOGO	387.5	298412.4	63.0	48481.4
TVGAVO-ENS	250.9	87318.2	54.8	19065.9
Total	*364.8*	*926952.6*	*59.2*	*150377.3*

Table 1: Performances of the used Minkowski algorithms in milliseconds CPU-time. N_p is the number of polyhedra in the equipment and N_v is the total number of vertices in the equipment. N_m is the number of Minkowski sums calculated, N_s the number of Minkowski sums that cut $\mathcal{Z}$, K the average number of vertices of the Minkowski sums, and K_s the average number of vertices of the (nonempty) sections with $\mathcal{Z}$.

6 Conclusions

We have proposed an optimal output-sensitive algorithm to compute a section of a three dimensional Minkowski sum by a plane without computing the whole sum. Degeneracies and numerical problems were solved.

The algorithm was used to compute the admissible space for the placement of antennas on a satellite. The implementation was compared to two methods that calculate the entire Minkowski sum on realistic satellite models and was shown to be very efficient in practice.

We are thinking about extending the functionalities of our program and will discuss here briefly some useful and interesting possibilities.

- Automatic layout. We can try to find some useful heuristics to place equipments automatically.
- A constraint that is more difficult to model is non-visibility. Instruments exist that serve to dissipate the heat generated by other equipments. When more than one such equipment is used then they may not "see" each other. The problem is that this constraint is not monotonic: placing an obstacle between two dissipators makes a previously inadmissible placement admissible again. This might make this constraint difficult to handle when we want to apply some automatic strategy. Another criterion to take into account could be the optimization of the inertia center of the satellite.
- One can also think about a post-processing step, that tries to compact the layout after all equipments have been placed, like is done for marker making in [LM93], for example.

7 Acknowledgments

The authors would like to thank Hervé Brönnimann for his example of a section of a Minkowski sum with quadratic complexity. Furthermore, thanks go to André Gasquet for his help on the specification of the antenna layout problem and on the implementation of the program. Thanks go also to our colleagues of the design office for answering many technical questions, supplying satellite models and testing our program.

References

[AB89] F. Avnaim and J.-D. Boissonnat. Polygon placement under translation and rotation. *RAIRO Inform. Theor.*, 23:5–28, 1989.

[ABD+94] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluating signs of determinants using single-precision arithmetic. Research Report 2306, INRIA, BP93, 06902 Sophia-Antipolis, France, 1994.

[ABD+95] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluation of a new method to compute signs of determinants. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C16–C17, 1995.

[Avn89] F. Avnaim. *Placement et déplacement de formes rigides ou articulées.* Thèse de doctorat en sciences, Université de Franche-Comté, Besançon, France, 1989.

[BDH93] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hull. Technical Report GCG53, Geometry Center, Univ. of Minnesota, July 1993.

[CEGS93] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair and parametric searching. *Discrete Comput. Geom.*, 10:183–196, 1993.

[DM95] Karen Daniels and Victor Milenkovic. Multiple translational containment: Approximate and exact algorithms. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 205–214, 1995.

[GS87] L. J. Guibas and R. Seidel. Computing convolutions by reciprocal search. *Discrete Comput. Geom.*, 2:175–193, 1987.

[LM93] Z. Li and V. Milenkovic. A compaction algorithm for non-convex polygons and its application. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 153–162, 1993.

[LP83] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. Comput.*, C-32:108–120, 1983.

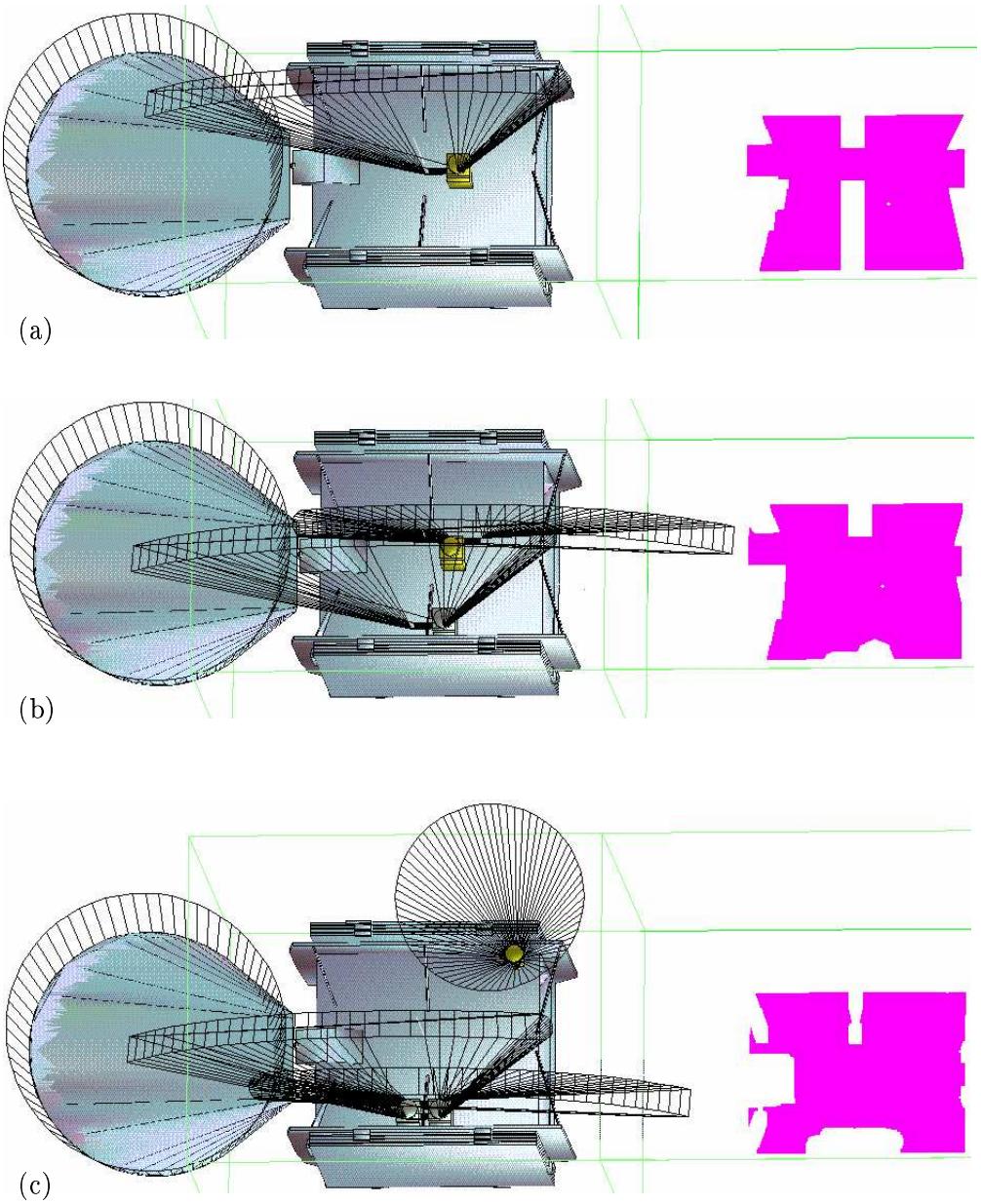

Figure 8: The sequence of placements performed for the imaginary satellite PRISMESAT: the admissible space for the equipment to be placed is drawn at the right side. (a) ESRI-1 (b) ESRI-2 (c) HORN-7-HST.

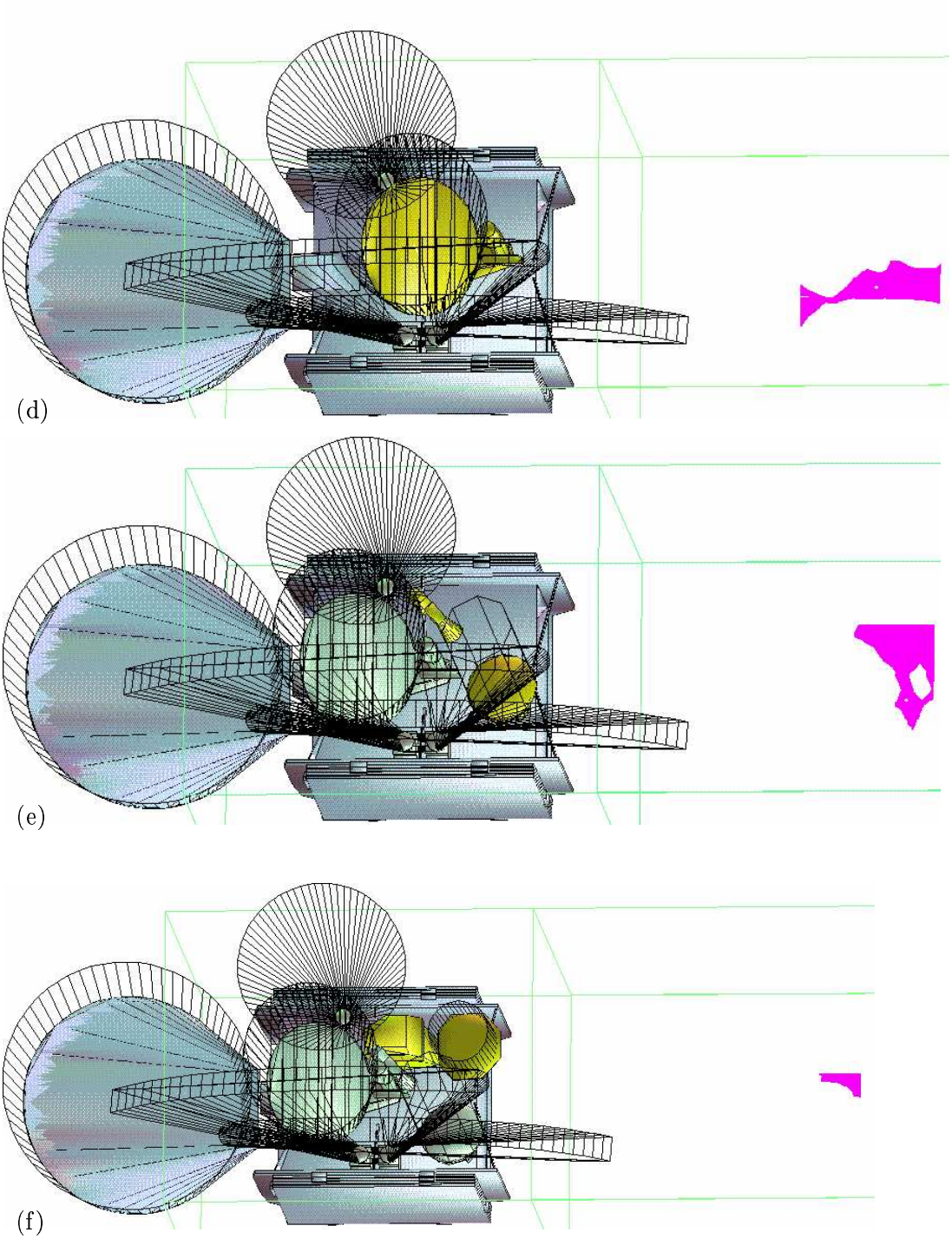

Figure 9: The sequence of placements performed for the imaginary satellite PRISMESAT (continued). (d) SVS-ANTENNA (e) VOGO (f) TVGAVO-ENS.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399