3D Snap Rounding

Olivier Devillers, Sylvain Lazard, William Lenhart

3D Snap Rounding

Olivier Devillers, Sylvain Lazard, William J. Lenhart

RESEARCH REPORT
N° 9149
February 2018
Project-Team Gamble

ISSN 0249-6399 ISRN INRIA/RR--9149--FR+ENG

3D Snap Rounding

Olivier Devillers, Sylvain Lazard, William J. Lenhart*

Project-Team Gamble

Research Report n° 9149 — February 2018 — 22 pages

Abstract: Let $\mathcal{P}$ be a set of n polygons in $\mathbb{R}^3$, each of constant complexity and with pairwise disjoint interiors. We propose a rounding algorithm that maps $\mathcal{P}$ to a simplicial complex $\mathcal{Q}$ whose vertices have integer coordinates. Every face of $\mathcal{P}$ is mapped to a set of faces (or edges or vertices) of $\mathcal{Q}$ and the mapping from $\mathcal{P}$ to $\mathcal{Q}$ can be done through a continuous motion of the faces such that (i) the L_∞ Hausdorff distance between a face and its image during the motion is at most $3/2$ and (ii) if two points become equal during the motion, they remain equal through the rest of the motion. In the worse the size of $\mathcal{Q}$ is $O(n^{15})$ and the time complexity of the algorithm is $O(n^{19})$ but, under reasonable hypotheses, these complexities decrease to $O(n^5)$ and $O(n^6\sqrt{n})$.

Key-words: Rounding – Polyhedron – Fixed size representation

* Williams College, USA

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Arrondi d'une soupe de triangles

Résumé : Soit $\mathcal{P}$ un ensemble de n polygones dans $\mathbb{R}^3$, chacun de complexité constante, et d'intérieurs disjoints. Nous présentons un algorithme d'arrondi tel que l'image de $\mathcal{P}$ soit un complexe simplicial $\mathcal{Q}$ dont les sommets ont des coordonnées entières. Chaque face de $\mathcal{P}$ est envoyée sur un ensemble de faces (ou arêtes ou sommets) de $\mathcal{Q}$ et, de plus, $\mathcal{P}$ peut être tranformé en $\mathcal{Q}$ par un mouvement continu des faces de telle sorte que (i) la distance de Hausdorff L_∞ entre une face et son image pendant le mouvement est au plus $3/2$ et (ii) si deux points deviennent égaux pendant le mouvement, ils restent égaux durant le reste de le mouvement. La taille de $\mathcal{Q}$ est au pire $O(n^{15})$ et la complexité temporelle de l'algorithme est $O(n^{19})$. Cependant, sous des hypothèses raisonnables, ces complexités peuvent être ramenées à $O(n^5)$ et $O(n^6\sqrt{n})$.

Mots-clés : Arrondi – Polyèdre – Représentation à précision fixe

1 Introduction

Rounding 3D polygonal structures is a fundamental problem in computational geometry. Indeed, many implementations dealing with 3D polygonal objects, in academia and industry, require as input pairwise-disjoint polygons whose vertices have coordinates given with fixed-precision representations (usually with 32 or 64 bits). On the other hand, many algorithms and implementations dealing with 3D polygonal objects in computational geometry output polygons whose vertices have coordinates that have arbitrary-precision representations. For instance, when computing boolean operations on polyhedra, some new vertices are defined as the intersection of three faces and their exact coordinates are rational numbers whose numerators and denominators are defined with roughly three times the number of bits used for representing each input coordinate. When applying a rotation to a polyhedron, the new vertices have coordinates that involve trigonometric functions. When sampling algebraic surfaces, the vertices are obtained as solutions of algebraic systems and they may require arbitrary-precision representations since the distance between two solutions may be arbitrarily small (depending on the degree of the surface).

This discrepancy between the precision of the input and output of many geometric algorithms is an issue, especially in industry, because it often prevents the output of one algorithm from being directly used as the input to a subsequent algorithm.

In this context, there exists no solution for rounding the coordinates of 3D polygons with the constraint that their rounded images do not properly intersect and that every input polygon and its rounded image remain close to each other (in Hausdorff distance). In practice, coordinates are often rounded without guarding against changes in topology and there is no guarantee that the rounded faces do not properly intersect one another.

The same problem in 2D for segments, referred to as snap rounding, has been widely studied and admits practical and efficient solutions [1, 5–11, 14]. Given a set of possibly intersecting segments in 2D, the problem is to subdivide their arrangement and round the vertices so that no two disjoint segments map to segments that properly intersect. For clarity, all schemes consider that vertices are rounded on the integer grid. It is well known that rounding the endpoints of the edges of the arrangement to their closest integer point is not a good solution because it may map disjoint segments to properly intersecting segments.. Snap rounding schemes propose to further split the edges when they share a pixel (a unit square centered on the integer grid). In such schemes, disjoint edges may collapse but this is inevitable if the rounding precision is fixed and if we bound the Hausdorff distance between the edges and their rounded images. Furthermore, it is NP-hard to determine whether it is possible to round simple polygons with fixed precision and bounded Hausdorff distance, and without changing the topological structure [12].

In dimension three, results are extremely scarse, despite the significance of the problem. Goodrich et al. [5] proposed a scheme for rounding segments in 3D, and Milenkovic [13] sketched a scheme for polyhedral subdivisions but, as pointed out by Fortune [4], both schemes have the property that rounded edges can cross. Fortune [3] suggested a high-level rounding scheme for polyhedra but in a specific setting that does not generalize to polyhedral subdivisions [4]. Finally, Fortune [4] proposed a rounding algorithm that maps a set $\mathcal{P}$ of n disjoint triangles in $\mathbb{R}^3$ to a set $\mathcal{Q}$ of triangles with $O(n^4)$ vertices on a discrete grid such that (i) every triangle of $\mathcal{P}$ is mapped to a set of triangles in $\mathcal{Q}$ at L_∞ Hausdorff distance at most $\frac{3}{2}$ from the original face and (ii) the mapping preserves or collapses the vertical ordering of the faces. Unfortunately, this rounding scheme is very intricate and, moreover, it uses a grid precision that depends on the number n of triangles: the vertices coordinates are rounded to integer multiples of about $\frac{1}{n}$.

The difficulty of snap rounding faces in 3D is described by Fortune [4]: First, it is reasonable to round every vertex to the center of the voxel containing it (a voxel is a unit cube centered on the integer grid). But, by doing so, a vertex may traverse a face and to avoid that, it might be

necessary to add beforehand a vertex on the face, which requires triangulating it. Newly formed edges may cross older edges when snapping; to avoid this, new vertices are added to these edges, in turn requiring further triangulating of faces. It is not known whether such schemes terminate.

To better understand the difficulty of the problem, consider the following simple but flawed algorithm. First project all the input faces onto the horizontal plane, subdivide the projected edges as in 2D snap rounding, triangulate the resulting arrangement, lift this triangulation vertically on all faces, and then round all vertices to the centers of their voxels. For an input of size n, this yields an output of size $\Theta(n^4)$ in the worst case and an L_∞ Hausdorff distance of at most $\frac{1}{2}$ between the input faces and their rounded images. Unfortunately, this algorithm does not work in the sense that edges may cross: indeed, consider two almost vertical close triangles whose projections on the horizontal plane are disjoint triangles that are rounded in 2D to the same segment; such triangles in 3D may be rounded into properly overlapping triangles. Fortune [4] solved this problem by using a finer grid to round the vertices and avoiding the faces to round vertically.

Contributions. We present in this paper the first algorithm for rounding a set of interior-disjoint polygons into a simplicial complex whose vertices have integer coordinates and such that the geometry does not change too much: namely, (i) the Hausdorff distance between every input face and its rounded image is bounded by a constant ($\frac{3}{2}$ for the L_∞ metric) and (ii) the relative positions of the faces are preserved in the sense that there is a continuous motion that deforms all input faces into their rounded images such that if two points collapse at some time, they remain identical up to the end of the motion. This ensure, in particular, that if a line stabs two input faces far enough from their boundaries, the line will stab their rounded images in the same order or in the same point.

The worst-case complexity of our algorithm is polynomial but unsatisfying as our upper bound on the output simplicial complex is $O(n^{15})$ for an input of size n. However, this upper bound decreases to $O(n^5)$ under some reasonable assumption on the input: roughly speaking that the input is a nice discretization of a constant number of surfaces that satisfy some reasonable hypothesis on their curvature. The corresponding time complexity reduces from $O(n^{19})$ to $O(n^6\sqrt{n})$. Furthermore, it very likely that these bounds are not tight and, in practice on realistic non-pathological data, we anticipate time and space complexities of $O(n\sqrt{n})$.

We present the algorithm in Section 3, its proof of correctness in Section 4, and its complexity analysis in Section 5.

2 Preliminaries

Notation. The coordinates in the Euclidean space $\mathbb{R}^3$ are referred to as x, y, and z and $\vec{\imath}, \vec{\jmath}, \vec{k}$ is the canonical basis. We use several planes parallel to the axes to project or intersect some faces: the xy-plane is called the *floor*, the xz-plane is called the *back wall* and a plane parallel to the yz-plane is called a *side wall*. Projections on the floor and on the back wall are always considered orthogonal to the plane of projection.

Two polygons, edges, or vertices are said to *properly intersect* if their intersection is non-empty and not a common face of both. Two polygons (resp. segments) intersect transversally if their relative interiors intersect and if they are not coplanar (resp. collinear).

General position assumption. For the sake of simplicity, we assume, without loss of generality, some general position on our input set of polygons $\mathcal{P}$. Precisely, we assume:

(α) No faces are parallel to the axes of coordinates and no vertices project in direction $\pm\vec{\jmath}$ on an edge (except the endpoints of that edge).

(β) No supporting plane of a face translated by $\pm\vec{\jmath}$ contains a vertex.

Let $\mathcal{I}$ denote the intersection, if not empty, of the supporting plane of a face with the translation by $\pm\vec{\jmath}$ of the supporting plane of another face. By assumption (β), $\mathcal{I}$ is a line.

(γ) No vertices project in direction $\pm\vec{\jmath}$ onto such a line $\mathcal{I}$.

(δ) For any point A on a face and with half-integer x and y-coordinates, $A \pm \vec{\jmath}$ does not belong to another face. More generally, no line $\mathcal{I}$ crosses any vertical line defined by half-integer x and y-coordinates.

This general position assumption is done with no loss of generality because it can be achieved by a sequence of four symbolic perturbations of decreasing importance: (i) the input faces are translated in the x-direction by ϵ_1, (ii) translated in the y-direction by ϵ_2, (iii) the vector $\vec{\jmath}$ is scaled by a factor $(1+\epsilon_3)$, and (iv) the faces are rotated by an angle ϵ_4 around a line that is not parallel to the coordinate axes. As shown below, enforcing $\epsilon_1 \gg \epsilon_2 \gg \epsilon_3 \gg \epsilon_4$ yields that our perturbation scheme removes all degeneracies.

Consider an intersection $\mathcal{I}$ as defined above; $\mathcal{I}$ can be a line or a plane. If $\mathcal{I}$ is a line L that induces a degeneracy of type (δ), this degeneracy is avoided by a translation (i) in the x-direction if L is not parallel to the yz-plane, and by a translation (ii) in the y-direction, otherwise. Then, perturbations (iii) and (iv) of smaller scales do not reintroduce this degeneracy [2]. If the intersection $\mathcal{I}$ is a plane, this remains the case after perturbations (i) and (ii), but the intersection becomes empty after a small enough perturbation (iii) and it remains empty after perturbation (iv). Hence, degeneracies of type (δ) are avoided by our scheme of perturbations.

Degeneracies of type (β) and (γ) are not affected by perturbations (i) and (ii), but they are avoided by the scaling of type (iii). Indeed, if $\mathcal{I}$ is a line then, viewed in projection on the back wall, the scaling of type (iii) translates the line. Finally, degeneracies of type (α) are not affected by perturbations (i-iii), but they are avoided by a rotation of type (iv).

3 Algorithm

We first describe the main algorithm in Section 3.1 and then two algorithmic refinements in Section 3.2. that we present separately for clarity. Our algorithm has the following property.

Theorem 1. *Given a set $\mathcal{P}$ of polygonal faces in 3D in general position and that do not properly intersect, the algorithm outputs a simplicial complex $\mathcal{Q}$ whose vertices have integer coordinates and a mapping σ that maps every face F of $\mathcal{P}$ onto a set of faces (or edges or vertices) of $\mathcal{Q}$ such that there exists a continuous motion that moves every face F into $\sigma(F)$ such that (i) the L_∞ Hausdorff distance between F and its image during the motion never exceeds $\frac{3}{2}$ and (ii) if two points on two faces become equal during the motion, they remain equal through the rest of the motion.*

3.1 Main Algorithm

The algorithm is organized in 4 steps. In every step of the algorithm, faces are subdivided and/or modified. We denote by $\mathcal{P}_i$ the set of faces at the end of Step i and by σ_i the mapping from the faces of $\mathcal{P}_{i-1}$ to those of $\mathcal{P}_i$ (with $\mathcal{P}_0 = \mathcal{P}$ and $\mathcal{P}_4 = \mathcal{Q}$). These mappings are trivial and not explicitly described, except in Step 1. Let $\sigma = \sigma_4 \circ \cdots \circ \sigma_1$ be the global mapping from the faces of $\mathcal{P}$ to those of the output simplicial complex $\mathcal{Q}$.

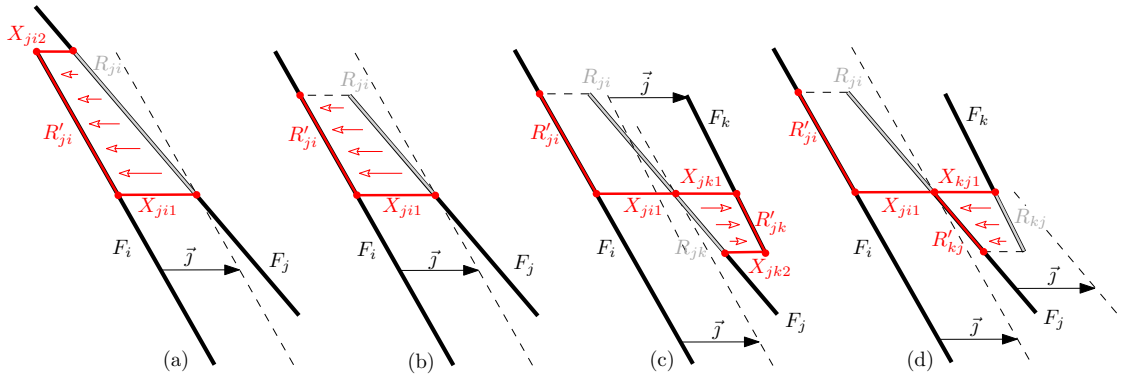


Figure 1: View inside a side wall $x = \text{cst}$. (a-d): The face F_j is partially projected onto F_i $(i < j)$, i.e. R_{ji} is replaced by the faces R'_{ji}, X_{ji1} and, in (a), X_{ji2}. (c): F_j is also partially projected onto F_k $(i < k < j)$. (d): If instead $i < j < k$, it is F_k that is partially projected onto F_j.

1. *Collapse the faces that are close to one another.* Order all the input faces arbitrarily from $\bar{F}_1$ to $\bar{F}_n$. During the process, we modify iteratively the faces. For clarity, we denote by F_i the faces that are iteratively modified, which we initially set to $F_i = \bar{F}_i$ for all i. Roughly speaking, for i from 1 to $n-1$, we project along y the points of $F_{i+1}, \ldots, F_n$ onto F_i but only the points that project at distance at most 1. Furthermore, we create, if needed, side walls that connect the boundary of the projected points to their pre-image. Structural properties of the faces at the end of this step are discussed in Section 3.3. Refer to Figure 1.

   (a) For i from 1 to n and for j from $i+1$ to n, modify F_j by removing the polygonal region R_{ji} that consists of the points $p_j \in F_j$ whose projection onto F_i along the y-direction lies within distance less than 1 from p_j, i.e. $R_{ji} = \{p_j \in F_j \mid \exists\, \alpha \in (-1, 1), \exists\, p_i \in F_i,\ p_j = p_i + \alpha\vec{j}\}$. Let $\tilde{F}_1, \ldots, \tilde{F}_n$ be the resulting faces at the end of the two nested loops and let R'_{ji} be the projection of R_{ji} on $\tilde{F}_i$ along the y-direction.[1]

   (b) For j from 1 to n, consider on $\bar{F}_j$ the set of R_{ji} $(i < j)$ and consider their edges, in turn. We define new faces that connect some edges of R_{ji} and R'_{ji}, which we refer to as *connecting faces* (see e.g., faces $X_{ji\zeta}$ in Figure 1). If edge e is a common edge of R_{ji} and $\tilde{F}_j$, we define a new face as the convex hull of e and its projection on F_i along y. If e is a common edge of R_{ji} and $R_{ji'}$ and if e projects (along y) on $\tilde{F}_i$ and on $\tilde{F}_{i'}$ into two distinct segments e_i and $e_{i'}$, respectively, we define a new face as the convex hull of e_i and $e_{i'}$; however, if e belongs to that face, we split it in two at e.

   (c) For i from 1 to n, subdivide $\tilde{F}_i$ by the arrangement of edges of the R'_{ji}, $j = i+1, \ldots, n$.

   To summarize, we have removed from every input face $\bar{F}_j$ the regions R_{ji}, $i = 1, \ldots, j-1$, we subdivided the resulting faces $\tilde{F}_i$ by the edges of all R'_{ji}, $j = i+1, \ldots, n$, and we created new connecting faces. Finally, we define σ_1 to map every input face $\bar{F}_j$ to the union of the

[1] In the nested loops, for $i = i_0$, we modify F_j for $j > i_0$ by removing R_{ji_0} from F_j and, in particular, we do not modify any F_i for $i \leqslant i_0$. Hence, from the time when any R_{ji_0} is defined, we do not modify F_{i_0} and thus $F_{i_0} = \tilde{F}_{i_0}$.

resulting face $\tilde{F}_j$, all the regions R'_{ji}, $i < j$ (subdivided as in $\tilde{F}_i$), and all the connecting faces that are defined by R_{ji}, $i < j$.

2. *Partition the space into slabs.* Project all the faces of $\mathcal{P}_1$ on the floor, compute their arrangement, lift all the resulting edges onto all faces of $\mathcal{P}_1$, and subdivide the faces accordingly. Then, project all these edges on the back wall and compute their arrangement.

   The closed region bounded by the two side walls $x = c \pm \frac{1}{2}$, $c \in \mathbb{Z}$, is called a *thin slab* $\mathcal{S}_c$, if it contains (at least) a vertex of these arrangements. A *big slab* is a closed region bounded by two consecutive thin slabs. (A big slab may thus be reduced to a plane.)

   We subdivide all faces of $\mathcal{P}_1$ by intersecting them with the side-wall boundaries of all slabs, resulting in $\mathcal{P}_2$.

3. *Triangulate the faces.* We triangulate all the faces of $\mathcal{P}_2$ in every slab in turn. We first consider thin slabs and then big slabs. This order matters because, when triangulating faces in thin slabs, we add vertices on edges, including those on the side-wall boundaries of the thin slabs, which also belong to the adjacent big slabs.

   (a) *Thin slabs.* Project along the x-axis all the faces in a thin slab $\mathcal{S}_c$ on the side wall $x = c$ and compute the arrangement of the projected edges. In that side wall, denote as hot all the pixels that contain a vertex of that arrangement and split every edge that intersects a hot pixel at its intersection with the pixel boundary.[2] Triangulate the resulting arrangement[3] and lift it back (still along the x-axis) onto all the faces in the slab and subdivide them accordingly. All these new vertices are referred to as *dummy vertices* to differentiate them from the other vertices.

   (b) *Big slabs.* Refer to Figure 2. Not considering the dummy vertices of Step 3a, all faces are trapezoids (possibly degenerated to triangles) such that each of the parallel edges lie on each of the side-wall boundaries of the big slab; any two trapezoids are either identical, disjoint, or share exactly one edge or vertex, and the same holds for their projections on the floor. The dummy vertices lie on the trapezoid edges that lie on the side-wall boundaries of the big slab.

   Not considering the dummy vertices, all trapezoids that project on the floor onto one and the same trapezoid are triangulated such that all the diagonals project on the floor onto one and the same diagonal. Trapezoids can only have dummy vertices on the edges on the side walls thus, after splitting a trapezoid in two triangles, each triangle can have dummy vertices on at most one of its edges. For every such triangle, we further triangulate it by adding an edge connecting every dummy vertex to the opposite vertex of the triangle.

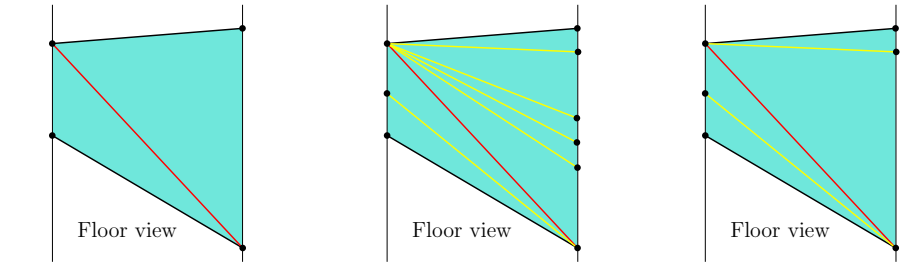
4. *Snap all vertices* to the centers of their voxels.

3.2 Algorithm Refinements

We present here two algorithmic refinements that we did not describe above for clarity.

[2] As in [7], these vertices are associated with the hot pixel so that the center of the pixel they will be snapped to is well defined. This ensures that no intersection is created during the snapping motion, but simply adding one vertex on the edges and strictly inside every hot pixel yields the same result.

[3] Before triangulating, add the hot pixel boundaries to the arrangement so that the triangulating edges do not cross hot pixels. Although triangulating the faces at this stage is useful for the proof of correctness of the algorithm, it improves the complexity without changing the output to triangulate these faces at the end of the algorithm instead; see Section 3.2.



(a) Triangulation of trapezoids not considering dummy vertices.

(b) Same as (a) but with dummy vertices.

(c) Same as (b) but on a single trapezoid.

Figure 2: Triangulations of trapezoids in a big slab.

3.2.1 Subdivision of the faces in thin slabs (Steps 2 and 3a).

When snapping all vertices to their voxel centers in Step 4, any planar polygon in a thin slab $\mathcal{S}_c$ is transformed into a planar polygon in the side-wall $x = c$. Depending on how the vertices move toward their voxel centers, the polygon may not remain planar during the motion but it is planar at the end of the motion. Hence, the output will be unchanged if, in Step 3a, we avoid triangulating the faces, that is, we avoid triangulating the arrangement in the side wall $x = c$ and only lift the new vertices of the arrangement onto the edges in $\mathcal{S}_c$. Still, after snapping the vertices in Step 4, the resulting polygons in the side wall $x = c$ should be triangulated so that the algorithm returns a simplicial complex. Doing so improves the complexity of the algorithm but it is nonetheless convenient for the proof of correctness to consider the triangulations of the faces in Step 3a.

Similarly, in thin slabs, we can avoid subdividing the faces by the vertical projections of edges in Step 2; however, since slabs are defined by the lifted edges, this means in Step 2 to vertically lift the edges on all the faces, without subdividing them, to define the slabs, and to subdivide the faces by the lifted edges only in the big slabs.

3.2.2 Subdivision of the connecting faces (Steps 2, 3a and 3b).

In Step 2, the connecting faces are subdivided by the side-wall boundaries of all slabs. The resulting faces are trapezoids (possibly degenerate to triangles) with two edges of length at most 1 that are parallel to the y-axis. Such trapezoids remain planar when their vertices are moved to their voxel centers in Step 4. Hence, triangulating these trapezoids does not modify the motion of any of its points. Furthermore, subdividing their edges that are parallel to the y-axis does not change the trapezoid motions either.

It follows that, in Steps 2, we do not need to subdivide the connecting faces by the vertical projections of the edges of $\mathcal{P}_1$, in Step 3a, we do not need to lift any dummy vertices on the connecting-face edges that are parallel to the y-axis and in Step 3b, we do not need to triangulate the connecting faces. Still, we should triangulate the connecting faces at the end of the algorithm in order to obtain a simplicial complex. These triangulations can trivially be done without creating proper intersections between the edges because the connecting faces that need to be triangulated are parallelograms with two edges of length 1 and parallel to the y-axis; edges are not properly intersecting at the end of Step 4, thus the edges that lie in such a parallelogram are all equal to one diagonal and we can triangulate the parallelogram accordingly.

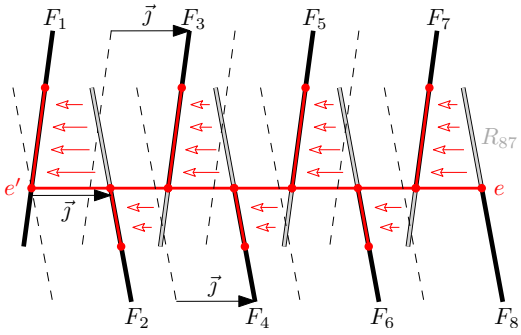


Figure 3: View in a side wall: the boundary edge e of R_{87} can be far way from the edge $e' = \bar{F}_1 \cap (\bar{F}_2 - \vec{\jmath})$ that coincides with e on the back wall.

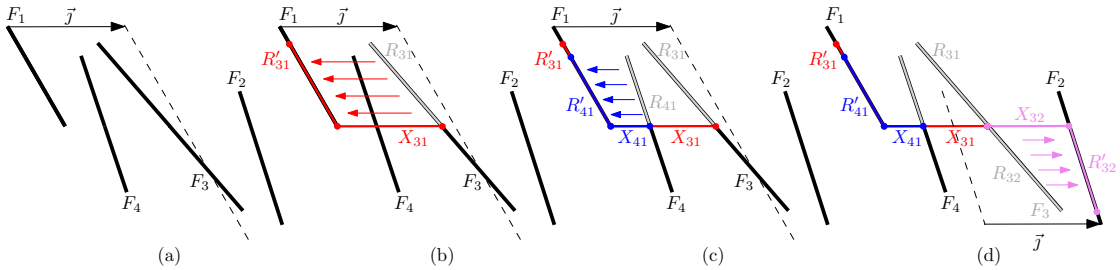


Figure 4: Example of overlapping connecting faces X_{31} and X_{41}, viewed inside a side wall. (a): The four input faces; (b) F_3 is projected onto F_1, then (c) F_4 onto F_1, and then (d) F_3 onto F_2. The remaining part of F_4 is not projected on any of the other faces.

3.3 Properties of the faces of $\mathcal{P}_1$

Observe first that, in Step 1a, R_{ji} is polygonal since its boundary consists of (i) segments of the boundary of F_j, (ii) segments of the boundary of F_i projected onto F_j along the y-direction, and (iii) segments of the intersection of F_j and the translated copies of F_i by vectors $\pm\vec{\jmath}$. This also implies that, in projection on the back wall, an edge e of R_{ji} lies in an edge e' of the boundary of an input face $\bar{F}_u$ or of $\bar{F}_u \cap (\bar{F}_v \pm \jmath)$ for some u and v. However, the distance between e and e' is not bounded from below by a constant, as depicted in Figure 3.

In Step 1b, connecting faces are defined as trapezoids that intersect any side wall in segments of length at most 1 that are parallel to the y-axis. However, it should be noted that connecting faces may overlap, as depicted in Figure 4, and that a connecting face may not contain the edge of R_{ji} that defines it, as depicted in Figure 5.

4 Proof of Correctness

We prove here Theorem 1. We focus on Step 1 of the algorithm in Section 4.1, on Step 4 in Section 4.2, and we wrap up in Section 4.3.

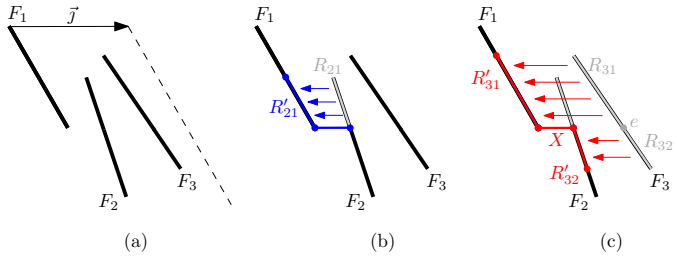


Figure 5: A connecting face X induced by the common edge e of R_{31} and R_{32}, and that does not contain e. View inside a side wall: (a) The three input faces; (b) F_2 is projected onto F_1, then (c) F_3 is projected onto F_1 and F_2.

4.1 Step 1

We first prove the main properties of Step 1 and then a technical lemma, which will be used in Lemma 6.

Lemma 2. *Every point of the faces of $\mathcal{P}$ can be continuously moved so that every face F of $\mathcal{P}$ is continuously deformed into $\sigma_1(F)$ such that (i) the L_∞ Hausdorff distance between F and its image during the motion never exceeds 1 and (ii) if two points on two faces become equal during the motion, they remain equal through the rest of the motion.*

Proof. The motion is decomposed in n successive phases, considering the projection on each F_i in turn. For a particular F_i, the naive way of moving R_{ji} to $\sigma_1(R_{ji})$ is to move R_{ji} to R'_{ji}, by moving each point of R_{ji} along the y-direction and at constant speed to R'_{ji}, and to transform edge e_ζ into face X_ζ for every edge e_ζ of the boundary of R_{ji} that defines a connecting face X_ζ. However, this does not define a function since segments are mapped to faces. The definition of a continuous motion requires a bit of technicality but the straightforward underlying idea is to subdivide R_{ji} by considering, for each edge e_ζ, a tiny quadrilateral bounded by e_ζ. We then transform continuously each tiny quadrilateral bounded by e_ζ into the connecting face X_ζ and move the complement of these quadrilaterals, which is a slightly shrunk version of R_{ji}, into R'_{ji}. This can be done so that when two distinct points become equal during the motion, they remain equal through the rest of the motion. The formal proof goes as follows.

Recall that $\bar{F}_1, \ldots, \bar{F}_n$ denote the input faces of $\mathcal{P}$ and that we initially set $F_i = \bar{F}_i$. In Step 1, for i from 1 to n and for j from $i+1$ to n, we modify F_j by projecting its subset R_{ji} onto $R'_{ji} \subset F_i$, and by adding the corresponding connecting faces. When we start the projection on F_i, faces F_u, for $u \leqslant i$, are no longer modified and we have $F_u = \tilde{F}_u$ until the end of Step 1.

Subdivision of R_{ji}. For all i and all $j \in \{i+1, \ldots, n\}$, we project parallel to y the R_{ji} on the back wall, triangulate the resulting arrangement, lift the triangulation back onto the R_{ji}, and subdivide the R_{ji} accordingly. Let $T_1, \ldots, T_g$ denote the resulting triangles.

By construction, the boundary of $T_u \subset R_{ji}$ consists of parts of (i) boundary edges of R_{ji} that define connecting faces (i.e., any boundary edge incident to R_{ji} and some $R_{ji'}$, and that projects along y onto $\tilde{F}_i$ and $\tilde{F}_{i'}$ into two distinct segments), (ii) boundary edges of R_{ji} that *do not* define connecting faces, and (iii) edges that are induced by the subdivision of the R_{ki} into triangles. Edges of types (ii) and (iii) will be moved onto $\tilde{F}_i$ and they are referred to as *red*, and edges of type (i) will remain invariant, at first, and they are referred to as *blue*. A vertex is considered blue if it is the endpoint of a blue edge (among all its incident edges in $T_1, \ldots, T_g$); otherwise, it is red.

Ordering property. Consider two triangles $T_b \subset R_{bi} \subset \bar{F}_b$ and $T_r \subset R_{ri} \subset \bar{F}_r$ that are on the same side of $\tilde{F}_i$ (with respect to y) and that project onto the same triangle on the back wall. We prove that, **if a blue edge e_b of T_b and a red edge e_r of T_r coincide in that projection, then T_b is farther away than T_r from $\tilde{F}_i$ with respect to y** (i.e. any line parallel to the y-axis, that intersects these triangles, intersects $\tilde{F}_i$, T_r and T_b in that order).

In projection on the back wall, an edge of R_{ji} lies in $\partial \bar{F}_u$ or $\bar{F}_u \cap (\bar{F}_v \pm j)$ with $u \leqslant i$ or $u = j$, and $v \leqslant i$. Indeed, R_{ji} only depends on F_i and F_j, at the time when R_{ji} is defined in Step 1, and then, F_i only depends on $\bar{F}_1, \ldots, \bar{F}_i$ and F_j only depends on $\bar{F}_1, \ldots, \bar{F}_{i-1}$ and $\bar{F}_j$. It follows that $\bar{F}_r$ plays no role in the definition of R_{bi} and thus that the boundary edges of $\bar{F}_r$ do no overlap the edges of R_{bi} in projection on the back wall, by items (α) and (γ) of the general position assumption. In particular, e_r cannot be a boundary edge of $\bar{F}_r$.

Thus, since e_r is a red edge, either (a) e_r lies on a common edge of R_{ri} and some $R_{ri'}$ (on $\bar{F}_r$) such that e_r projects (along y) on $\tilde{F}_i$ and on $\tilde{F}_{i'}$ onto the same segment $e_{ri} = e_{ri'}$, or (b) e_r is induced by the subdivision of the R_{ki} into triangles.

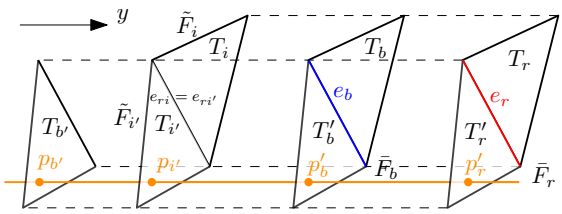


Figure 6: Counter example for proof of the ordering property.

Refer to Figure 6. In case (a), since input faces are interior disjoint, the segment $e_{ri} = e_{ri'}$ is on the common boundary of the input faces $\bar{F}_i$ and $\bar{F}_{i'}$. Hence, e_r is a common edge of $T_r \subset R_{ri}$ and some $T'_r \subset R_{ri'}$ such that they respectively project in Step 1 onto two triangles $T_i \subset \tilde{F}_i$ and $T_{i'} \subset \tilde{F}_{i'}$ that share edge $e_{ri} = e_{ri'}$. Assume for contradiction that edge e_b is in between e_r and e_{ri} (i.e. in their convex hull). Since e_b is a blue edge, the triangle $T'_b \subset \bar{F}_b$ that share edge e_b with T_b projects in Step 1 onto a triangle $T_{b'} \neq T_{i'}$. Consider any line parallel to the y-axis that intersects the interior of the four triangles $T_{b'}, T_{i'}, T'_b$ and T'_r in points $p_{b'}, p_{i'}, p'_b$ and p'_r, respectively. By definition segment $p'_r p_{i'}$ has length less than 1 and contains p'_b. Thus $\|p'_b p_{i'}\| < 1$ and since T'_b projects in Step 1 onto $T_{b'} \neq T_{i'}$, we have $b' < i'$. Furthermore, $i' < r$ by definition of $R_{ri'}$. Thus $b' < i' < r$ and since T'_r projects on $T_{i'}$ instead of $T_{b'}$, $\|p'_r p_{b'}\| > 1$ while $\|p'_b p_{b'}\| < 1$. Hence, $p_{b'}, p_{i'}, p'_b$ and p'_r appear in that order on their supporting line. It follows that $\|p_{i'} p_{b'}\| < 1$ and thus $T_{i'}$ should have been projected on $T_{b'}$ in Step 1, which contradicts the definition of $R_{ri'}$, and thus e_b is not in between e_r and e_{ri}. Since e_r and e_b are on the same side of $\tilde{F}_i$, e_b is farther away than e_r from $\tilde{F}_i$. Case (b) is similar: the only difference is that $T_{i'}$ lies in $\tilde{F}_i$ instead of $\tilde{F}_{i'}$ but T_i and $T_{i'}$ are still incident. The ordering property follows.

Subdivision of $T_1, \ldots, T_g$. In the rest of the proof, unless otherwise specified, we consider i fixed and we consider among all triangles $T_1, \ldots, T_g$, only those that project in Step 1 on $\tilde{F}_i$ (i.e. that belong to R_{ji} for some j) onto the same triangle, and, furthermore, that are on one same side of $\tilde{F}_i$ (with respect to y).

We choose a parameter ϵ small enough and subdivide all these triangles T_j as follows. Refer to Figure 7. The general idea is to define a small part of the triangle in the neighborhood of blue edges and vertices. More precisely, for a blue vertex v incident to one or two red edges of the triangle, we define the retractions of v to be the points on these red edges that are, viewed on the back wall at distance $\lambda\epsilon$ from v, where λ is the distance between v and its projection on

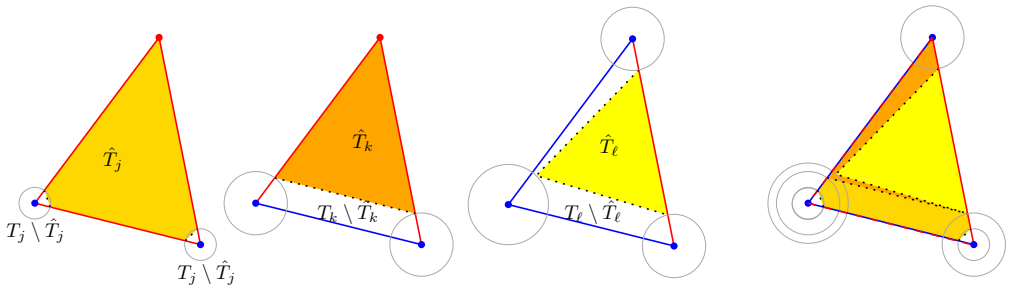


Figure 7: Back wall views of the subdivision of three triangles T_j, T_k, T_ℓ that project on $\tilde{F}_i$ into the same triangle and such that $\tilde{F}_i, T_j, T_k, T_\ell$ appear in that order along y.

$\tilde{F}_i$ along y; for a red vertex, its retraction is the vertex itself; and for a blue vertex incident to two blue edges, we define its retraction to be located on the triangle at some relevant place at distance, viewed on the back wall, $\lambda\epsilon$ from v. Then, we define the retraction of a triangle T_j to be $\hat{T}_j$ the convex hull of its retracted vertices.

Considering the triangles from the closest to the farthest from their projection of $\tilde{F}_i$, the ordering property ensures that the blue vertices that are incident to two blue edges can be retracted as defined above such that, on the back wall, the $\hat{T}_j$ are nested so that the inner of two $\hat{T}_j$ and $\hat{T}_k$ is the one that is the farthest away from $\tilde{F}_i$ (along y). Notice that, among all triangles $T_1, \ldots, T_g$, if T_j and T_k share a red edge (in $\mathbb{R}^3$), then $\hat{T}_j$ and $\hat{T}_k$ share the same vertices on this red edge.

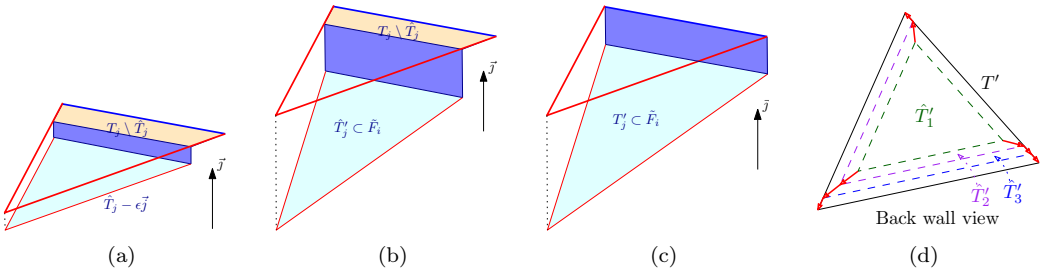


Figure 8: Motion of T_j.

Motions of the T_u. We define the motions of the triangles in four phases. In the first phase (see Figure 8(a)), we infinitesimally deform T_j into the surface defined as the union of $T_j \setminus \hat{T}_j$, a translated copy of $\hat{T}_j$ toward $\tilde{F}_i$ along y, and, for each blue edge or isolated blue vertex of T_j, a small parallelogram that connects these two parts. This can trivially be done continuously without creating intersection for any two triangles T_j and T_k that are disjoint or if their projections on the back wall are distinct. On the other hand, if T_j and T_k share an edge and project (on the back wall) on the same triangle, the shared edge is necessarily red (since input faces have disjoint relative interior). The fact that their subdivision polylines are distinct on the back wall except for a common endpoint on the red edge ensures that the motion can be performed so that the relative interiors of the deforming faces remain disjoint.

In the second phase (see Figure 8(b)), all infinitesimally translated faces $\hat{T}_j \pm \epsilon\vec{j}$ are moved to

$\tilde{F}_i$; this is done trivially my moving each point onto $\tilde{F}_i$ at constant speed parallel to y. The faces $T_j \setminus \hat{T}_j$ remain invariant. This defines the motions of the vertices of the infinitesimal wall added in the first phase; the points in their relative interiors move accordingly to their barycentric coordinates in their faces. Since the subdivision polylines are disjoint in the back wall except at points where the polylines coincide in $\mathbb{R}^3$, any two distinct points remain distinct during the motion except possibly at the end of the phase when they may become equal on $\tilde{F}_i$.

In the third phase (see Figure 8(c)), (i) every face of $T_j \setminus \hat{T}_j$ is shrunk to the blue edge (or isolated blue vertex) that defines it, (ii) the image of $\hat{T}_j$ at the end of the second phase, that is the projection of $\hat{T}_j$ onto $\tilde{F}_i$ (parallel to y), is expanded to the projection of T_j on $\tilde{F}_i$, and (iii) the image of the infinitesimal walls of the first phase are moved to the connecting faces (or edges) between T_j and its projection onto $\tilde{F}_i$.

Refer to Figure 8(d) and consider all the triangles T_j that project (parallel to y) onto one and the same triangle T' on $\tilde{F}_i$. In the definition of the subdivision of the T_j, the $\hat{T}_j$ are nested in the back wall. Denote by $\hat{T}'_1 \subset \cdots \subset \hat{T}'_h \subset T'$ their projections (parallel to y) on $\tilde{F}_i$ ordered from the innermost to the outermost. For simplicity, define $\hat{T}'_{h+1} = T'$. All these polygons are close to T' (for the Hausdorff distance) and there is thus a natural correspondance between their vertices.

On $\tilde{F}_i$, we move each vertex of $\hat{T}'_j$ at constant speed toward its corresponding vertex on $\hat{T}'_{j+1}$. For simplicity, we consider these motions one after the other, for $j = 1$ to h. During the j-th motion, the points inside $\hat{T}'_j$ move accordingly to their barycentric coordinates in $\hat{T}'_j$, and the points in $T' \setminus \hat{T}'_{j+1}$ are invariant. The points on $T_k \setminus \hat{T}_k$ move on T_k accordingly to the motion of their projection (parallel to y) on $\tilde{F}_i$. This defines the motion of the points on the boundary of all the faces that are parallel to y and the points in their relative interiors move accordingly to their barycentric coordinates. It is straightforward that any two distinct points remain distinct during this motion except possibly at the end of the phase when some points (viewed on the back wall on the boundary of T') may become equal.

Finally, the fourth phase goes as follows. Consider, as in Figure 5, a common edge e of R_{ji} and $R_{ji'}$ (on $\bar{F}_j$) such that e projects (along y) on $\tilde{F}_i$ and on $\tilde{F}_{i'}$ into two distinct segments e_i and $e_{i'}$, respectively, that are on the the same side of $\bar{F}_j$ with respect to y (i.e., e does not belong to the convex hull of e_i and $e_{i'}$). In Step 1b, we define a connecting face as the convex hull of e_i and $e_{i'}$. However, since e is a blue edge and has thus not yet been moved at this stage, R_{ji} and $R_{ji'}$ have currently been deformed into a set of faces that contains the convex hulls of e and e_i, and the convex hull of e and $e_{i'}$. These two faces overlap between e and, say, e_i. In this phase, we simply retract these overlapping parts into segment e_i.

Hausdorff distance. The property that the Hausdorff distance between a face and its image during the motion never exceeds 1 (for the L_∞ metric) is straightforward since, (i) any line parallel to y that intersects a triangle T_j also intersects its image during the motion, (ii) the image of T_j during the motion remains in the convex hull of T_j and its projection (parallel to y) on $\tilde{F}_i$ and (iii) all the points of T_j are at distance at most 1 along y from $\tilde{F}_i$ (by definition of R_{ji}). □

Lemma 3. *If a line L parallel to the y-axis intersects the relative interior of a face of $\mathcal{P}_1$ in a single point p then the distance along L from p to any other face of $\mathcal{P}_1$ is at least 1.*

Proof. Observe first that any point in a connecting face lies on a segment of length at most 2, parallel to the y-axis and with its endpoints on two non-connecting faces. Indeed, connecting faces are defined either as (i) the convex hull of an edge on $\tilde{F}_j$ and its projection on $\tilde{F}_i$ along y at distance at most 1, or (ii) the convex hulls of an edge on $\bar{F}_j$ and, respectively, its projections on $\tilde{F}_i$ and $\tilde{F}_{i'}$ in opposite directions along y and each at distance at most 1.

If a line L parallel to the y-axis intersects the relative interior of a face F of $\mathcal{P}_1$ in a single point p, this face is not a connecting face. Assume for a contradiction that L intersects another face F' of $\mathcal{P}_1$ at distance less than 1 from p. If F' is a connecting face then, by the above observation, L also intersects a non-connecting face at distance at most 1 from p. Furthermore, this distance is exactly 1 only if the above segment has length exactly 2 with p its midpoint; but this is impossible since p is in the interior of F, and faces do not transversally intersect, by Lemma 2. Thus, L intersects a non-connecting face at distance less than 1 from p. In other words, we can assume that F' is a non-connecting face.

Since non-connecting faces are not parallel to the y-axis, there exists a line L' parallel to the y-axis (and close to L) that intersects the relative interiors of both F and F' in two points at distance less than 1. This is impossible after Step 1, which concludes the proof. □

4.2 Step 4

In the following, we consider in the snapping phase of Step 4 a continuous motion of the vertices such that every vertex moves on a straight line toward the center of its voxel at a speed that is constant for each vertex and so that all vertices start and end their motions simultaneously. The motion of the other points in a face move accordingly to their barycentric coordinates in the face. Note that, in every voxel that contains a vertex, the motion is an homothetic transformation whose factor goes from one to zero. During that motion, we consider that thin and big slabs respectively shrink and expand accordingly.

We recall the standard snap-rounding result for segments in two dimensions. A pixel is called *hot* if it contains a vertex of the arrangement of segments.

Theorem 4 ([7, Thm. 1])**.** *Consider a set of segments in 2D split in fragments at the hot pixels boundaries and a deformation that (i) contracts homothetically all hot pixels at the same speed*[4] *and (ii) moves the fragments outside the hot pixels according to the motions of their endpoints. During the deformation, no fragment endpoint ever crosses over another fragment.*

Lemma 5. *When moving all vertices to the center of their voxels in Step 4, no two faces, edges, or vertices of $\mathcal{P}_3$ properly intersect in thin slabs.*

Proof. Consider all the faces of $\mathcal{P}_3$ in a thin slab $\mathcal{S}_c$ and the arrangement of their projections (along the x-axis) onto the side wall $x = c$. In that side wall, a pixel that contains a vertex of the arrangement is hot and every edge (in that side wall) that intersects a hot pixel is split at the pixel boundary (Step 3a). By Theorem 4, when moving in that side-wall all the vertices to the centers of their pixels, the topology of the arrangement does not change except possibly at the end of the motion, where edges and vertices may become identical.

It follows that the property that every face of $\mathcal{P}_3$ in $\mathcal{S}_c$ projects onto a single face of the arrangement in the side wall $x = c$, which holds by construction at the beginning of the motion (Step 3a), holds during the whole motion of the vertices in 3D and of their projections in the side wall $x = c$.

Furthermore, the motion preserves the ordering of the x-coordinates of the vertices in $\mathcal{S}_c$, until the end when they all become equal to c. Together with the previous property, this implies that, in a thin slab, during the snapping motion, (i) no vertices and edges intersect the relative interior of a face and (ii) if two edges intersect in their relative interior, it is at the end of the motion and they become identical. Furthermore, (iii) no vertices intersect the relative interior of an edge because, in Step 3a, we have split every edge that intersects a hot pixel in projection in the side wall $x = c$. This concludes the proof. □

[4] The proof in [7] considers separately motions in x and in y but the same argument applies for simultaneous homothetic contractions in x and y.

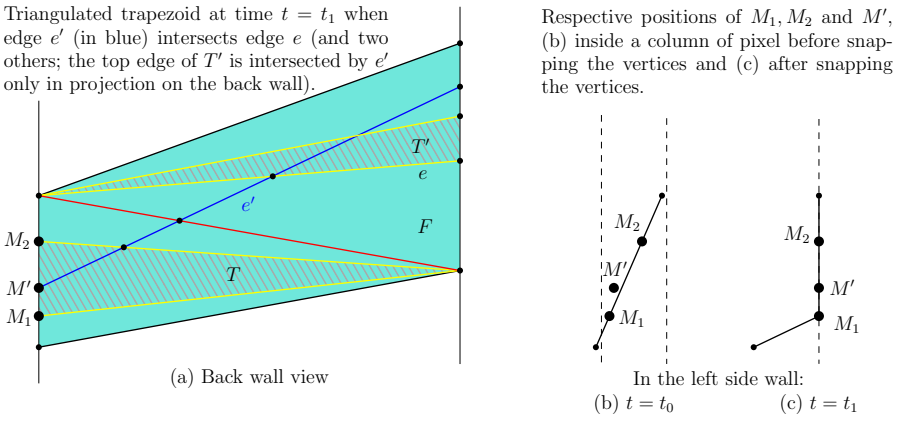


Figure 9: For the proof of Lemma 6.

Lemma 6. *When moving all vertices to the center of their voxels in Step 4, no two faces, edges, or vertices of $\mathcal{P}_3$ properly intersect in big slabs.*

Proof. By construction (Step 2), all the vertices in a big slab are on its side-wall boundaries and, in these side walls, no two edges or vertices properly intersect during the motion, by Lemma 5. Thus, we only have to consider edges that connect the two side-wall boundaries of a big slab and show that such edges do not properly intersect during the snapping motion. Note that input faces are not vertical by assumption and connecting faces are not vertical in big slabs since they are parallel to the y-axis.

Initially, these edges project on the floor onto edges that do not properly intersect pairwise (by definition of the slabs in Step 2). Thus, by Theorem 4, the projections on the floor of two edges either (i) coincide throughout the whole motion, or (ii) they do not properly intersect and do not coincide throughout the whole motion except possibly at the end when they may coincide. In the first case, throughout the whole motion, the edges belong to the same moving vertical plane and they do not properly intersect since they do not initially; indeed, since faces are not vertical, edges may intersect in a vertical plane only if they are boundary edges of trapezoids of $\mathcal{P}_2$, and such edges do not properly intersect on the back wall by definition of big slabs. Hence, only in the latter case (ii), two edges may properly intersect during the motion, and the first time this may happen is at the end of the motion and then, the two edges belong to the same vertical plane.

Similarly, applying Theorem 4 to the back-wall projection of the boundary edges of the trapezoids, we get that if two boundary edges of trapezoids properly intersect in 3D during the motion, it is at the end and they must coincide in the back-wall projection. Since two edges that coincide in two projections are equal, we get that boundary edges of trapezoids cannot properly intersect throughout the motion. It remains to prove that there is no proper intersections that involve the edges triangulating the trapezoids.

Consider for a contradiction two edges e and e' that properly intersect in a vertical plane V at time t_1, the end of the motion. Since boundary edges of the trapezoids do not properly intersect, we can assume without loss of generality that one of the two edges, say e, is a triangulation edge. Consider at time t_1, the triangulation of the (deformed) trapezoid F that contains e. We prove that (at time t_1) **edge e' properly intersects (at least) one of the two boundary edges of F.**

Assume for a contradiction that e' properly intersects none of the two boundary edges of

F and refer to Figure 9(a). Consider all the edges of the triangulation of F that are properly intersected by e' and the sequence of triangles (of that triangulation) that are incident to these edges; let T and T' denote the first and last triangles of that sequence. All these triangles except possibly one, T or T', must be in the vertical plane V; this is trivial for all triangles but T and T' and, if neither T nor T' lies in V, then edge e' properly intersects the surface formed by these triangles, contradicting the property that t_1 is the first time a proper intersection may occur.

As in Figure 9(a), assume without loss of generality that T, the bottommost triangle of the sequence, lies in the vertical plane V at time t_1. Let M' be the endpoint of e' that lies in T and let M_1M_2 be the edge of T that supports M'. At time t_1, M_1 and M_2 are vertically aligned and M' is in between them. Thus, before the snapping motion starts, at time $t = t_0$, M_1, M_2 and M' must lie in the same vertical column of pixel (in the side wall) and M' must be vertically in between M_1 and M_2 (see Figure 9(b)). However, the distance along the y-axis between M' and segment M_1M_2 is at least 1 by Lemma 3. Thus, M' and $M' \pm \vec{\jmath} \in M_1M_2$ are initially on opposite sides of the column of pixels and thus that have half-integer x and y-coordinates, which contradicts item (δ) of our general position hypothesis.

Hence, at time $t = t_1$, edge e' properly intersects one of the boundary edges, say r, of trapezoid F. Since boundary edges do not properly intersect, e' must be a triangulation edge of its trapezoid F' and we can apply the same argument as above on edges e' and r, instead of e and e'. We get that r properly intersects a boundary edge r' of F', which is a contradiction. □

4.3 Wrap up, proof of Theorem 1

First, by construction, the algorithm outputs faces that have integer coordinates.

Second, there is a continuous motion of every input face F into $\sigma(F)$ so that the Hausdorff distance between F and its image during the motion never exceeds $\frac{3}{2}$ for the L_∞ metric. Indeed, by Lemma 2, the Hausdorff distance never exceeds 1 between F and its image during the motion Step 1; in Steps 2 and 3 the faces are only subdivided; and the Hausdorff distance between any face of $\mathcal{P}_3$ and and its image during the motion of Step 4 clearly never exceeds $\frac{1}{2}$ since vertices are moved to the centers of their respective voxels.

Third, if two points on two faces become equal during the motion, they remain equal through the rest of the motion. This is proved in Lemma 2 for the motion of Step 1 and this also holds for the motion of Step 4 since, by Lemmas 5 and 6, if two faces, edges or vertices intersect during this motion, they share a common face of both, whose motion is uniquely defined by its vertices (actually, we show in the proofs of Lemmas 5 and 6 that no two distinct points become equal except possibly at the end of the motion).

Finally, the algorithm outputs a simplicial complex by Lemmas 5 and 6.

5 Complexity

We first analyse in Section 5.1 the complexity of our algorithm in the worst case (Proposition 7) and then, in Section 6, its complexity under some reasonable assumptions on the input (Proposition 10). We finally argue in Remark 12 that we can anticipate time and space complexities of $O(n\sqrt{n})$ in practice on realistic non-pathological data.

5.1 Worst-Case Complexity

We define the z-cylinder of a face F as the volume defined by all the lines parallel to the z-axis that intersect F; similarly for x and y-cylinders. Over all input faces F, let f_d be the maximum number of input faces that are (i) intersected by one such cylinder of F and (ii) at distance at

most d from F. Denote $f = f_\infty$ the maximum number of faces intersected by one such cylinder. Let w_x be the the maximum number of input faces that are intersected by any side wall $x = c$.

Proposition 7. *Given a set of $O(n)$ polygons, each of constant complexity, the algorithm outputs a simplicial complex of complexity $O(nw_x^2 f^7 f_1^5)$ in time $O(n^2 w_x f^9 f_1^7)$.*

Proof. For analyzing the complexities of the algorithm and of its output, we bound the number of edges that we consider for splitting the input faces. However, for each face, we first count the number of edges without considering any intersection; in other words, for each face, we bound the number of lines supporting the edges instead of the complexity of the induced arrangement. To avoid confusion, we refer to these edges as unsplit edges.

Number of slabs. After projection on the back wall, the edges of R_{ji} and R'_{ji}, in Step 1, are pieces of the boundary edges of the input faces $\bar{F}_k$ and of the segments of intersection $\bar{F}_k \cap (\bar{F}_\ell \pm \vec{\jmath})$. In a y-cylinder of a face F, only f faces project on the back wall and thus there are $O(ff_1)$ such edges. Thus, at the end of Step 1, every input face ends up supporting $O(ff_1)$ unsplit edges. In Step 2, we thus lift $O(f^2 f_1)$ unsplit edges onto every face. Every unsplit edge on a given face F may only intersect, after projection on the back wall, edges that lie on the faces that intersect the y-cylinder of F; there are $O(f)$ such faces and $O(f^2 f_1)$ unsplit edges on each of them, thus every unsplit edge may intersect $O(f^3 f_1)$ edges on the back wall. There are $O(nf^2 f_1)$ unsplit edges in total, hence, in Step 2, the back wall arrangement has complexity $O(nf^5 f_1^2)$. (Similarly, the floor arrangement has complexity $O(nf^3 f_1^2)$.) The number of thin and big slabs is thus $O(nf^5 f_1^2)$.

Complexity in thin slabs. For any thin slab $\mathcal{S}_c$, we analyze the complexity of the output in the side wall $x = c$. We do not consider here any triangulation edge inside the faces as discussed in Section 3.2.1. Thus, the edges in $\mathcal{S}_c$ are subdivisions of the edges of $\mathcal{P}_1$ and subdivisions of the edges defined as the intersection of the faces of $\mathcal{P}_1$ and the side walls $x = c \pm \frac{1}{2}$. More precisely, these edges are pieces of either (a.i) edges of $\mathcal{P}_1$ that lie on the input faces, (a.ii) edges of intersection of an input face with a side wall $x = c \pm \frac{1}{2}$, (b.i) edges of connecting faces that are parallel to the y-axis, or (b.ii) edges of intersection of a connecting face with a side wall $x = c \pm \frac{1}{2}$. Note that, although we do not consider the faces triangulated in $\mathcal{S}_c$, the edges are subdivided according the arrangement of their projections on the side wall $x = c$; however, we consider them unsplit for now.

As mentioned above, every input face supports $O(ff_1)$ unsplit edges at the end of Step 1. Thus, if $F_{\mathcal{S}_c}$ denotes the number of input faces that are intersected by the thin slab $\mathcal{S}_c$, there are $O(F_{\mathcal{S}_c} ff_1)$ edges of types (a) in $\mathcal{S}_c$. On the other hand, the $O(ff_1)$ unsplit edges on the back wall yield an arrangement of size $O(f^2 f_1^2)$; viewed on the back wall, edges of type (b) are incident to the vertices of this arrangement, so the number of edges of type (b) in $\mathcal{S}_c$ is bounded by $O(F_{\mathcal{S}_c} f^2 f_1^2)$.

We bound the number of intersection between these edges at the end of the algorithm. However, as noted in Section 3.2.2, we do not consider intersections that involve edges of type (b) because such intersections necessarily belong to the hot pixels defined by the edge endpoints. We thus consider here only edges of type (a). Every edge on a given input face F may only intersect, after projection on the side wall $x = c$, edges that lie on faces that intersect $\mathcal{S}_c$ and the x-cylinder of F;[5] there are $O(f_1)$ such faces and $O(ff_1)$ edges on each of them, thus every edge may intersect $O(ff_1^2)$ edges on the side wall $x = c$. There are $O(F_{\mathcal{S}_c} ff_1)$ edges of type (a) in $\mathcal{S}_c$, thus, there are $O(F_{\mathcal{S}_c} f^2 f_1^3)$ intersections between edges of type (a).

[5] It is crucial here to only consider edges of type (a) because an edge e of type (b) may intersect edges that are not in the x-cylinder of the input faces that define the connecting face of e.

In addition to these $O(F_{\mathcal{S}_c} f^2 f_1^3)$ intersections, there are $O(F_{\mathcal{S}_c} f^2 f_1^2)$ edges (and thus edge endpoints) in $\mathcal{S}_c$. The number of hot pixels in the side wall $x = c$ is thus $H_c = O(F_{\mathcal{S}_c} f^2 f_1^3)$ at the end of Step 3a.

The number of vertices (dummy or not) in $\mathcal{S}_c$ at the end of Step 3a is the number of hot pixels, H_c, times the number of edges in $\mathcal{S}_c$, $O(F_{\mathcal{S}_c} f^2 f_1^2)$. However, the size of the arrangement in the side wall $x = c$ after snapping the vertices to their voxel centers in Step 4 is of the order of the number hot pixels, H_c, and it remains as such after triangulating the faces (see Section 3.2.1).

Complexity in big slabs. The number of edges in a big slab after the triangulation of Step 3b is (i) the number of edges of connecting faces in the slab, that is $O(w_x f f_1)$ (the number $O(f f_1)$ of unsplit edges on input faces times the number $O(w_x)$ of input faces that intersect the slab) plus (ii) the number $O(w_x)$ of input faces that intersect the slab times the number $H_{c_1} + H_{c_2}$ of hot pixels in the adjacent thin slabs $\mathcal{S}_{c_1}$ and $\mathcal{S}_{c_2}$. This sums up to $O(w_x(H_{c_1} + H_{c_2}))$.

Complexity of the output. The total complexity of the output is thus $O(w_x)$ times the sum over all thin slabs of the number of hot pixels H_c in $\mathcal{S}_c$. Denoting by $\mathring{F}_{\mathcal{S}_c}$ the number of input faces that are entirely inside $\mathcal{S}_c$, we have that $F_{\mathcal{S}_c} \leqslant \mathring{F}_{\mathcal{S}_c} + 2w_x$ and that the sum over all thin slabs of $\mathring{F}_{\mathcal{S}_c}$ is $O(n)$. Hence, the sum over all $N = O(nf^5 f_1^2)$ thin slabs of the numbers of hot pixels $H_c = O(F_{\mathcal{S}_c} f^2 f_1^3)$ is $O((n + Nw_x) f^2 f_1^3) = O(nw_x f^7 f_1^5)$. Times w_x gives the complexity of the output: $O(nw_x^2 f^7 f_1^5)$.

Time complexity. All the arrangements and triangulations performed by the algorithm can be done in time complexities that match their worst sizes. However, this does not match the complexity of the output because the complexity of $\mathcal{P}_3$ may be larger before than after snapping. As noted above, the complexity of $\mathcal{P}_3$ in a thin slab is $O(H_c F_{\mathcal{S}_c} f^2 f_1^2)$. On the other hand, the complexity of $\mathcal{P}_3$ in a big slab is, as above, $O(w_x(H_{c_1} + H_{c_2}))$. The sum of these complexities over all slabs is bounded by $nf^2 f_1^2$ times the total number $O(nw_x f^7 f_1^5)$ of hot pixels, that is $O(n^2 w_x f^9 f_1^7)$. □

6 Complexity under some Assumptions

We consider, in Proposition 10, the complexity of our algorithm for approximations of "nice" surfaces, defined as follows.

Definition 8. *A (ε, ζ)-sampling of a surface $\mathcal{S}$ is a set of vertices on $\mathcal{S}$ so that there is at least 1 and at most ζ vertices strictly inside any ball of radius ε centered on $\mathcal{S}$. It is straightforward that a (ε, ζ)-sampling of a compact surface has $\Theta(n)$ vertices with $n = \frac{1}{\varepsilon^2}$.*

The Delaunay triangulation of a set of points $\mathcal{P}$ restricted to a surface $\mathcal{S}$ is the set of simplices of the Delaunay triangulation of $\mathcal{P}$ whose dual Voronoi faces intersect $\mathcal{S}$. If $\mathcal{P} \subset \mathcal{S}$, we simply refer to the restricted Delaunay triangulation of $\mathcal{P}$ on $\mathcal{S}$.

A surface $\mathcal{S}$ is k-monotone (with respect to z) if every line parallel to the z-axis intersects $\mathcal{S}$ in at most k points. Let Δ and k be any two positive constants. A surface $\mathcal{S}$ is nice *if it is a compact smooth k-monotone surface such that the Gaussian curvature of $\mathcal{S}$ is larger than a positive constant in a ball of radius Δ centered at any point $p \in \mathcal{S}$ where the tangent plane to $\mathcal{S}$ is vertical.*

The following lemma is a technical though rather straightforward result.

Lemma 9. *The restricted Delaunay triangulation $\mathcal{T}$ of a (ε, ζ)-sampling on a compact surface has complexity $O(n) = O(\frac{1}{\varepsilon^2})$. Any plane $x = c$ intersects at most $O(\sqrt{n}) = O(\frac{1}{\varepsilon})$ faces of $\mathcal{T}$. Furthermore, for any face f of $\mathcal{T}$, the set of vertical lines through f intersects at most $O(n^{\frac{1}{4}}) = O(\frac{1}{\sqrt{\varepsilon}})$ faces of $\mathcal{T}$.*

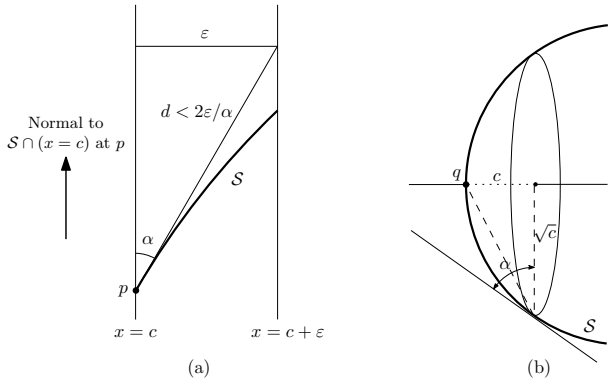


Figure 10: For the proof of Lemma 9.

Proof. Since $\mathcal{S}$ is k-monotone, we can assume without loss of generality that any plane $x = \text{cst}$ intersects $\mathcal{S}$ in at most one component. Indeed, this assumption will affect the actual overall complexity by a factor at most $\frac{k}{2}$.

Observe first that the edges of $\mathcal{T}$ have length less than 2ε. Indeed, if there is a Delaunay edge of length at least 2ε, there is a ball centered on $\mathcal{S}$ that contains this edge and that contains no vertices strictly inside it. This ball has radius at least ε, contradicting the (ε, ζ) sampling hypothesis. The (ε, ζ)-sampling of $\mathcal{S}$ ensures that there are $O(1)$ vertices at distance at most 2ε from every vertex, hence the triangulation has complexity $O(n) = O(\frac{1}{\varepsilon^2})$.

Intersection with a plane $x = c$. Since the edges of $\mathcal{T}$ have length less than 2ε, any edge that intersects a plane $x = c$ has a vertex in between the two planes $x = c \pm \varepsilon$. We prove in the following the area of $\mathcal{S}$ in that region is in $O(\varepsilon)$ and thus that the number of sampling points in that region is $O(\frac{1}{\varepsilon}) = O(\sqrt{n})$.

Observe first that if $\mathcal{S}$ intersects a plane $x = c$ in a curve of perimeter ℓ and if, at any point p on that curve, the plane tangent to $\mathcal{S}$ makes an angle at least α with $x = \text{cst}$, then the area of $\mathcal{S}$ in between $x = c$ and $x = c + \varepsilon$ is in $O(\frac{\ell\,\varepsilon}{\alpha})$ when ε tends to zero (see Figure 10(a)).

Consider a point $q \in \mathcal{S}$ where the tangent plane is parallel to a side-wall plane $x = \text{cst}$ and assume without loss of generality that p is located at the origin. Refer to Figure 10(b). By the hypotheses, there exists a constant $\delta > 0$ such that, for any $0 < c < \delta$, (i) the intersection of $\mathcal{S}$ with the plane $x = c$ has perimeter at most $\sqrt{c}$ up to some constant and (ii) the plane tangent to $\mathcal{S}$ at any point on that curve and the plane $x = c$ make an angle at least $\alpha > \frac{c}{\sqrt{c}} = \sqrt{c}$ up to some constant. Hence the area of $\mathcal{S}$ in between $x = c$ and $x = c - \varepsilon$ is in $O(\sqrt{c}\,\frac{\varepsilon}{\sqrt{c}}) = O(\varepsilon)$.

On the other hand, for any $c > \delta$, the intersection of $\mathcal{S}$ with the plane $x = c$ has perimeter $\ell = O(1)$ and the plane tangent to $\mathcal{S}$ at any point on that curve and the plane $x = c$ make an angle at least $\alpha = \frac{\delta}{\sqrt{\delta}} = \sqrt{\delta} = \Omega(1)$. Hence the area of $\mathcal{S}$ in between $x = c$ and $x = c + \varepsilon$ is in $O(\frac{\ell\,\varepsilon}{\sqrt{\delta}}) = O(\varepsilon)$, which concludes the proof that any plane $x = c$ intersects at most $O(\sqrt{n}) = O(\frac{1}{\varepsilon})$ faces of $\mathcal{T}$.

Intersection with the z-cylinder of a face. Consider a face f of $\mathcal{T}$ and its z-cylinder defined as the set of vertical lines through f. Similarly as above, the z-cylinder of f intersects a face f' only if a vertex of f' lies in the z-cylinder enlarged by ε. Since the edges of f have length at most 2ε, this enlarged z-cylinder is contained in the z-cylinder defined by a square of edge length 3ε in the xy-plane. In that z-cylinder, the height span of $\mathcal{S}$ is $O(\sqrt{\varepsilon})$ (see Figure 10(b)),

hence the area of $\mathcal{S}$ in the cylinder is $O(\varepsilon\sqrt{\varepsilon})$. The number of sampling points in the cylinder is thus $O(\frac{\varepsilon\sqrt{\varepsilon}}{\varepsilon^2}) = O(\frac{1}{\sqrt{\varepsilon}}) = O(n^{\frac{1}{4}})$. □

Proposition 10. *Given the arrangement of the restricted Delaunay triangulations of the (ε, ζ)-samplings of a constant number of nice surfaces, the algorithm outputs a simplicial complex of complexity $O(n^5)$ in time $O(n^6\sqrt{n})$.*

Proof. Consider the restricted Delaunay triangulations of the (ε, ζ)-samplings of two surfaces. By definition of (ε, ζ)-samplings, it is straightforward that any triangle of one triangulation intersects a constant number of triangles of the other triangulation. Hence, the complexities of Lemma 9 hold for the arrangement of the two triangulations, and similarly for a constant number of triangulations. Thus, for the arrangement of triangulations, Lemma 9 yields $w_x = O(\sqrt{n})$ and $f_1 < f = O(\sqrt[4]{n})$ (as defined in Section 5.1) and plugging these values in the complexities of Proposition 7 yields the result. □

Remark 11. *It should be stressed that the above complexities do not depend on the grid size, including in the hidden constants. This is important because for a fixed grid size, the number of voxels that are intersected by a given compact surface (or its convex hull) is a constant C hence, independently of n, the complexity of the snap-rounded simplicial complex will be in $O(C) = O(1)$. It follows that, for a fixed grid size, it would be pointless to consider the asymptotic complexity of the simplicial complex in terms of n. However, this makes sense if this complexity is independent from the grid size, which is the case in the above proposition.*

Remark 12. *In practice on realistic non-pathological data, one can anticipate a better complexity of $O(n\sqrt{n})$ for the size of the output and for the time complexity.*

Indeed, the x, y or z-cylinders of most faces will intersect a constant number of other faces and we can expect that the few that intersect a non-constant number of faces will not impact the final complexities; hence we can consider that $f_1 < f = O(1)$ in Proposition 7. Moreover, considering the proof of Proposition 7 with $f_1 < f = O(1)$, we get that the number of hot pixels in a thin slab $\mathcal{S}_c$ is $H_c = O(F_{\mathcal{S}_c})$, that the number of edges in $\mathcal{S}_c$ is $O(F_{\mathcal{S}_c})$, and that the number of edges in a big slab is $O(w_x)$. In practice, this should yield, before snapping, arrangements of complexity $O(w_x + F_{\mathcal{S}_c})$ in a big and thin slab instead of the worst-case complexities $O(w_x F_{\mathcal{S}_c})$ and $O(F_{\mathcal{S}_c}^2)$ for big and thin slabs, respectively. Summing over all slabs, this yields an anticipated practical complexity of $O(nw_x)$ where $w_x = O(\sqrt{n})$ as in Lemma 9.

7 Conclusion

Natural questions arise. The algorithm we presented is simple enough that it should be implementable without particular difficulties. However, its worst-case complexity, even under reasonable assumptions (Propositions 7 and 10), is prohibitive in practice. Hence, the question of whether our estimated practical complexity of $O(n\sqrt{n})$ is correct on real data is crucial for applications. Furthermore, it would be interesting to design heuristics for improving the practical efficiency of the algorithm. Another issue is that faces can drift arbitrarily far when the snap rounding scheme is applied repeatedly. Several approaches were presented to address this issue in 2D [8, 10, 14] but the problem in 3D is naturally entirely open.

Acknowledgements

The authors would like to thank André Lieutier for having initially pointed out the problem and its significance for industry, and for many discussions on the problems, Hazel Everett, Mark de

Berg, Danny Halperin, Raimund Seidel, and Dave Bremner.

References

[1] Mark de Berg, Dan Halperin, and Mark Overmars. An intersection-sensitive algorithm for snap rounding. *Computational Geometry*, 36(3):159–165, 2007. doi:10.1016/j.comgeo.2006.03.002.

[2] Olivier Devillers, Menelaos Karavelas, and Monique Teillaud. Qualitative Symbolic Perturbation: Two Applications of a New Geometry-based Perturbation Framework. *Journal of Computational Geometry*, 8(1):282–315, 2017. doi:10.20382/jocg.v8i1a11.

[3] Steven Fortune. Polyhedral modelling with multiprecision integer arithmetic. *Computer-Aided Design*, 29(2):123 – 133, 1997. Solid Modelling. doi:doi.org/10.1016/S0010-4485(96)00041-3.

[4] Steven Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. *Discrete & Computational Geometry*, 22(4):593–618, 1999. doi:10.1007/PL00009480.

[5] Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 284–293. ACM, 1997. doi:10.1145/262839.262985.

[6] Daniel H. Greene and F. Frances Yao. Finite-resolution computational geometry. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 143–152. IEEE, 1986. doi:10.1109/SFCS.1986.19.

[7] Leonidas J. Guibas and David H. Marimont. Rounding arrangements dynamically. *International Journal of Computational Geometry & Applications*, 8(02):157–178, 1998. doi:10.1142/S0218195998000096.

[8] Dan Halperin and Eli Packer. Iterated snap rounding. *Computational Geometry*, 23(2):209–225, 2002. doi:10.1016/S0925-7721(01)00064-5.

[9] John Hershberger. Improved output-sensitive snap rounding. *Discrete & Computational Geometry*, 39(1):298–318, Mar 2008. doi:10.1007/s00454-007-9015-0.

[10] John Hershberger. Stable snap rounding. *Computational Geometry*, 46(4):403–416, 2013. doi:10.1016/j.comgeo.2012.02.011.

[11] John D. Hobby. Practical segment intersection with finite precision output. *Computational Geometry*, 13(4):199–214, 1999. doi:10.1016/S0925-7721(99)00021-8.

[12] V. Milenkovic and L. R. Nackman. Finding compact coordinate representations for polygons and polyhedra. *IBM Journal of Research and Development*, 34(35):753–769, 1990.

[13] Victor Milenkovic. Rounding face lattices in d dimensions. In *Proceedings of the 2nd Canadioan Conference on Computational geometry*, pages 40–45, 1990.

[14] Eli Packer. Iterated snap rounding with bounded drift. *Computational Geometry*, 40(3):231 – 251, 2008. doi:doi.org/10.1016/j.comgeo.2007.09.002.

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399