Chained segment offsetting for ray-based solid representations

Jonas Martinez, Samuel Hornus, Frédéric Claux, Sylvain Lefebvre

Inria, Loria

Abstract

We present a novel approach to offset solids in the context of fabrication. Our input solids can be given under any representation: boundary meshes, voxels, indicator functions or CSG expressions. The result is a ray-based representation of the offset solid *directly used for visualization and fabrication*: We never need to recover a boundary mesh in our context.

We define the offset solid as a sequence of morphological operations along line segments. This is equivalent to offsetting the surface by a solid defined as a Minkowski sum of segments, also known as a *zonotope*. A zonotope may be used to approximate the Euclidean ball with precise error bounds.

We propose two complementary implementations. The first is dedicated to solids represented by boundary meshes. It performs offsetting by modifying the mesh in sequence. The result is a mesh improper for direct display, but that can be resolved into the correct offset solid through a ray representation. The major advantage of this first approach is that no loss of information – re-sampling – occurs during the offsetting sequence. However, it applies only to boundary meshes and cannot mix sequences of dilations and erosions. Our second implementation is more general as it applies directly to a ray-based representation of any solid and supports any sequence of erosion and dilation along segments. We discuss its fast implementation on modern graphics hardware. Together, the two approaches result in a versatile tool box for the efficient offsetting of solids in the context of fabrication.

Keywords: morphological operations, 3D modeling, fabrication

1. Introduction

Morphological operations [1] – such as erosions and dilations – are important operations in solid modeling [2, 3]. In the context of fabrication, erosions and boolean differences can be used for example to hollow a solid or create a mold, while closing operations can remove small holes in a model. Figure 1 illustrates a few morphological operations obtained by our method.

Many approaches consider the *offset surface* obtained after the dilation or erosion of the solid by the Euclidean ball of radius d centered at the origin. The offset surface is the set of points at distance d from the object boundary. The exterior (resp. interior) offset is the subset of the offset surface lying outside (resp. inside) the solid. The exact computation of offset surfaces for general inputs is difficult. Therefore, a number of approximations have been proposed (see Section 2). However, many of these approximations either restrict the type of input, perform aggressive re-sampling, or require computationally heavy and relatively complex algorithms [4].

In this work we consider sequences of erosions and dilations along line segments. It is worth noting that the result of a sequence of *dilations* along segments is equivalent to a Minkowski sum between the solid and an object known as a *zonotope*. The zonotope is defined as the Minkowski sum of the set of segments.

A zonotope is usually sufficient for our target applications in manufacturing: the main differences with a ball are essentially aesthetic (see Figure 12), and often only impact hidden surfaces when used for molds and hollowing. Nevertheless, there are known algorithms to approximate a ball with a zonotope within a prescribed error bound [5, 6]. Sequences of erosions and dilations along line segments therefore provide a general framework to perform complex morphological operations. This includes closings and openings, obtained by mixing dilations and erosions in sequence.

Our work is focused on obtaining ray-based solid representations [7] for direct visualization and fabrication – typically through slicing and additive manufacturing [8, 9]. We do not attempt to recover a boundary representation of the result. Our modeler takes any solid

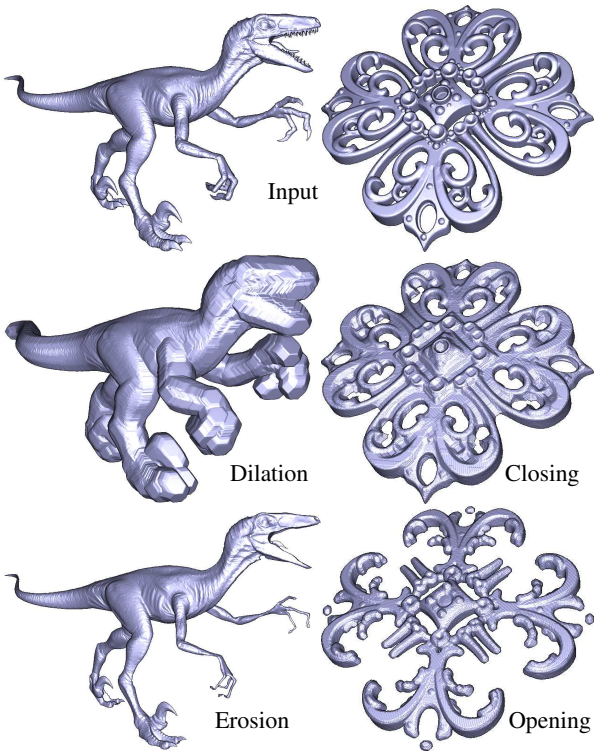


Figure 1: Segment morphological operations with the specialized approach for meshes (left column) and the generic algorithm (right column). The mesh approach shows the dilation and erosion with a truncated octahedron (zonohedra, see Section 6). Notice the small erosion successfully applied to the raptor model. The generic approach is used to perform a closing and an opening also with the truncated octahedron.

representation as input – boundary meshes, voxels, CSG expressions – and converts them into ray-based representations for visualization and fabrication. The conversion occurs at the resolution of the screen or manufacturing process, therefore minimizing the loss of information due to sampling. Our modeler is based on a fast GPU implementation, enabling the construction of ray-based representations at high resolutions (see Section 7).

Contributions. The key observation of our work is that the ray representation of solids is amenable to a simple and fast implementation of morphological operations with *line segments*, and as a consequence, to sequences of morphological operations with zonotopes. To the best of our knowledge, no previous work considers morphological operations between zonotopes and ray-based solid representations. Unlike most of the existing methods, our technique avoids any explicit treatment of topological changes. Erosions and dilations can be combined in any order to achieve complex operations.

We propose two complementary techniques. First, in Section 4 we introduce an efficient algorithm to perform morphological operations on a ray-based representation of a solid. The advantage of this approach is that it applies to any solid that can be captured by a ray-based representation. Its drawback stems from the sampling resolution that approximates the solid at each step. We discuss error bounds for the process in Section 4.3. In our context, and thanks to the high computational efficiency of the presented technique, we can afford the use of a resolution matching that of the manufacturing process of the final object.

Second, we propose in Section 5 a specialized approach for boundary meshes, which postpones the conversion to a ray-based representation to *after* an entire sequence of dilations or sequence of erosions, thereby removing any re-sampling error due to intermediate steps.

The time complexity of the presented algorithms is bounded by the complexity of the solid surface, instead of its volume. Thus, their performance is expected to scale better than voxelization methods. We provide an implementation of all of our algorithms which are both simple to implement and highly parallel.

2. Related work

This section reviews existing approaches for the computation of offset surfaces in general, and then focuses on methods using ray-based representations.

Computing offset surfaces. Early approaches rely on convolutions to compute offset surfaces and Minkowski sums. These methods obtain a superset of primitives of the offset surface that are trimmed and filtered to form the final boundary [10]. Evans and Koppelman [11] compute the Minkowski sum of a polyhedral object along a sequence of translational sweeps, and propose to approximate the Euclidean ball with a zonotope for surface offsetting. To the best of our knowledge this is the only previous approach that considers zonotopes for morphological operations, but it focuses on generating polyhedral results while our focus is on ray-representations. Kaul and Rossignac [12] presented a set of criteria to filter the primitives that do not belong to the Minkowski sum. Peternell and Steiner [13] presented a convolution algorithm for objects with piecewise boundaries. Campen and Kobbelt [14] introduced an exact approach for Minkowski sums between polyhedra that also culls a superset of primitives. Convolution methods usually suffer from geometric robustness issues.

The offset surface can also be extracted from the distance field of the object surface, as it implicitly repre-

sents offset surfaces. Frisken *et al.* [15] presented the adaptively sampled distance-fields, which among other operations, is able to perform surface offsetting. Varadhan and Manocha [16] approximate the Minkowski sum with a distance field isosurface extraction, guaranteeing a Hausdorff distance bound on the approximation. Pavić and Kobbelt [17] traverse an octree and split each cell which is potentially intersected by the offset surface, in order to recover it. Lee *et al.* [18] presented an accurate method to compute the distance field, which is able to render offset surfaces by considering a union of balls. The main drawback of distance field methods is that they usually require high amount of memory in order to ensure accuracy.

Offset surfaces can also be computed from point-based representations. Chen *et al.* [19] generate a set of candidate points that are used to obtain a voxelization of the offset surface. Lien *et al.* [20] and Netaluri and Shapiro [21] perform the Minkowski sum between two point-based surfaces. The approach explicitly distinguishes the interior and boundary points of the Minkowski sum. Recently, Calderon and Boubekeur [22] introduced a morphological analysis framework for point clouds, which is able to perform morphological dilations and erosions. These operations remain expensive on point sets as the interior of the solid is not explicitly available.

A last family of methods generates a voxelization of the offset surface. Li and McMains [23, 24] and Leung *et al.* [25] presented GPU approaches to compute the Minkowski sum of polyhedra by computing pairwise Minkowski sums, and obtaining a voxelization of its union. The memory requirements of these methods rise rapidly as the voxelization resolution increases. In addition, the error tends to be larger than that of a ray-based approach where the sampling directions can freely vary.

Surface offsetting with ray-based representations. To the best of our knowledge, the *dexel structure* [7] was the first introduced ray representation of solids. For a single direction and a uniform grid of rays parallel to that direction, the dexel structure stores the intervals of the rays lying inside the solid; these intervals are called *dexels* (depth elements). The G-buffer [26] extended the dexel structure by storing the surface normal and the identifier of the intersected object. Another ray representation are ray-reps [27, 28], which additionally stores the CSG half-spaces or the B-Rep faces that a ray intersects. The triple ray representation [29] are three ray-reps obtained by sampling rays in the three orthogonal directions. Layered depth images (LDI) [30], used for image-based rendering, represent surfaces by sampling rays in several directions. Layered depth-normal images (LDNI) [31], are a dexel structure also storing the surface normals, and are computed along the three orthogonal directions to represent solids.

There exists few offsetting methods that consider ray representations. Menon and Voelcker [32] suggested approximating the Minkowski sum between A and B by computing the union of some ray-rep instances of A over the boundary of B. In the image space solid sweeping of Hui [33], the solid is transformed to a ray representation, the sweeping of a solid along a trajectory is computed by taking the union of a finite set of ray representations of the solid. However, uniform offsetting is not considered. Chen and Wang [34] presented an offsetting approach that initially generates a superset of primitives from an input polyhedron. Then, it constructs a LDNI and filters the LDNI points of the superset that belong to the offset surface. Wang and Manocha [35], place spheres on the LDI sampled points, and compute their union in the GPU. As the offsetting distance is increased, the number of intersections between spheres rapidly increases, and it is proposed to decompose the offset surface computation into a composition of smaller offsets.

3. Notations, definitions and properties

We introduce below the required definitions for mathematical morphology along line segments and zonotopes. Below, we give the basic notations employed throughout this paper.

$d_H(A, B)$	Hausdorff distance between A and B
$A \setminus B$	Set difference. $A \setminus B = \{p \in A \mid p \notin B\}$
$\overline{A}$	Complement of A. $\overline{A} = \mathbb{R}^n \setminus A$
$\mathbb{B}_r$	Closed ball of radius r centered at the origin
∂A	Boundary of the set A

3.1. Morphology operators

Let $A \subset \mathbb{R}^n$ and $b \in \mathbb{R}^n$. The translation of A by b is:

$$A^b = \{a + b \mid a \in A\}$$

The *morphological dilation* or *Minkowski sum* between two sets $A, B \subset \mathbb{R}^n$ is:

$$A \oplus B = \bigcup_{b \in B} A^b$$

Analogously, the *morphological erosion* is defined as:

$$A \ominus B = \bigcap_{b \in B} A^{-b} = \overline{\overline{A} \oplus (-B)}$$

where $-B = \{-b \mid b \in B\}$. The *morphological opening* of A by B is $(A \ominus B) \oplus B$, and the *morphological closing* of A by B is $(A \oplus B) \ominus B$. In the following, the dilation or erosion between A and B is denoted in general as $A \circ B$. If one expression contains more than one $\circ$, it refers exclusively to either dilation or erosion.

3.2. Zonotope offsetting

A *zonotope* is a Minkowski sum of k line segments $s_1, s_2, \ldots, s_k \in \mathbb{R}^n$, denoted as $\mathcal{Z}$:

$$\mathcal{Z} = s_1 \oplus s_2 \oplus \ldots \oplus s_k$$

For $\mathbb{R}^3$, $\mathcal{Z}$ is referred to as a *zonohedron*. In n-dimensions, a zonotope with k segment generators has combinatorial complexity $O\left(k^{n-1}\right)$ [6].

As morphological operations are associative, and $A \ominus (B \oplus C) = (A \ominus B) \ominus C$, we have:

$$O \circ \mathcal{Z} = (((O \circ s_1) \circ s_2) \circ \ldots) \circ s_k$$

Thanks to this property the offset solid $O \circ \mathcal{Z}$ is obtained by a sequence of k morphological operations with a line segment (see Figure 2), either all dilations or all erosions.

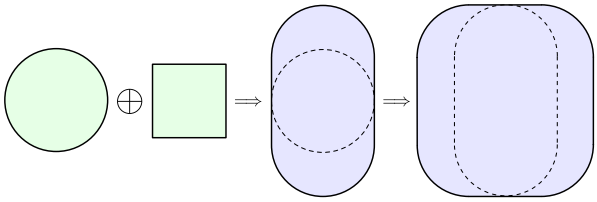


Figure 2: Illustration of a morphological dilation between a disk and a square. The square is a two dimensional zonotope defined by the Minkowski sum of two perpendicular segments of equal length. Incremental dilations with these segments lead to the final result.

3.3. Ray representations

Let O be a compact subset of $\mathbb{R}^3$, $\vec{v}$ a unit direction vector in $\mathbb{R}^3$, and L be a set of lines parallel to $\vec{v}$. We define

$$R(L, O) = \bigcup_{l \in L} O \cap l$$

and view $R(L, O)$ as an organized collection of pairwise disjoint and closed line segments in space.

A *dexel structure* [7] is a three dimensional *discrete* ray representation. Consider a regular grid of squares of side length Δ, lying on a plane with normal $\vec{v}$. Let L_Δ be the set of lines parallel to $\vec{v}$ passing trough the center of the squares. The dexel structure is the set of line segments $R(L_\Delta, O)$. Each line segment in $R(L_\Delta, O)$ is also called a *dexel* [7]. Each grid cell contains the list of dexels coming from the corresponding line in L_Δ and sorted along the increasing $\vec{v}$ direction. When the context is clear, we shorten the notation and write R only, instead of $R(L_\Delta, O)$.

We will often need to rasterize the volume represented by a dexel structure R. To do so, we define the *dexel volume* $\mathcal{P}(R)$ as the Minkowski sum of R with a square χ_R centered at the origin, orthogonal to $\vec{v}$, of side length Δ, with sides aligned with the grid structure of R:

$$\mathcal{P}(R) = R \oplus \chi_R.$$

More precisely, the square χ_R is defined as the Minkowski sum of two half-open segments as follows. Let $(\vec{u}, \vec{w})$ be the orthonormal basis of the grid structure of R in the plane orthogonal to $\vec{v}$. Then $\chi_R = \{x\vec{u} + y\vec{w} \mid -\Delta/2 \leq x, y < \Delta/2\}$. In this way, the Minkowski sums of two dexels of R with χ_R are always disjoint. Note that the closure of $\mathcal{P}(R)$ (the union of $\mathcal{P}(R)$ with its boundary) is an orthogonal polyhedron.

3.4. Segment morphological operations

Consider the segment s with unit vector direction $\vec{v}$ (the same $\vec{v}$ as above) and length $2d > 0$, centered at the origin:

$$s = \left\{p \in \mathbb{R}^3 \mid p = \lambda\vec{v}, \lambda \in [-d, d]\right\}$$

Thus, $O \circ s$ is a segment morphological operation. Observe that $\forall l \in L$ we have that:

$$(O \cap l) \circ s \subset l$$

This implies that segment morphological operations can be performed independently for each line $l \in L$. Given a dexel volume $\mathcal{P}(R)$ of the form $\mathcal{P}(R) = R \oplus \chi_R$ where R is a dexel structure of direction $\vec{v}$ (hence parallel to s), we have that

$$\begin{aligned}
\mathcal{P}(R) \circ s &= (R \oplus \chi_R) \circ s \\
&= \left(\bigsqcup_{l \in L_\Delta} (l \cap R) \oplus \chi_R \right) \circ s \\
&= \bigcup_{l \in L_\Delta} ((l \cap R) \oplus \chi_R) \circ s
\end{aligned}$$

where $\sqcup$ denotes a disjoint union. Then, since χ_R is orthogonal to $\vec{v}$ we can permute the operators:

$$\begin{aligned}\mathcal{P}(R) \circ s &= \left(\bigcup_{l \in L_\Delta} (l \cap R) \circ s \right) \oplus \chi_R \\ &= R' \oplus \chi_R \\ &= \mathcal{P}(R')\end{aligned}$$

where R' is also a dexel structure. Therefore, we can erode or dilate $\mathcal{P}(R)$ with a segment s parallel to $\vec{v}$ by simply eroding or dilating its dexel structure.

4. General algorithm for dexel structures

This section describes our general algorithm for performing *segment* morphological operations on dexel structures. In Section 4.1 we discuss how to offset the dexel structure along a segment *aligned with the ray direction* of the structure. We explain in Section 4.2 how this process can be performed in sequence, each time re-sampling the previous result in a new dexel structure having a different direction. We analyze the error due to this process in Section 4.3. Finally, in Section 4.4 we provide details on our parallel GPU implementation, as there are non trivial considerations in terms of fragment complexity and surface cracks avoidance.

4.1. Morphological operations along the ray direction

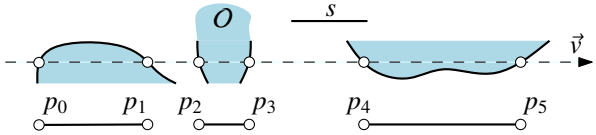
In this section we are given a dexel structure R oriented along some direction $\vec{v}$ and sampled from an input solid O. We give an algorithm to construct the dexel structure $R \circ s$ where s is a line segment parallel to $\vec{v}$ of length $2d$, centered at the origin, as defined above.

The algorithm considers each line l in L_Δ and processes the ordered set of dexels in $l \cap R$. For erosion, we shrink each dexel by displacing their endpoints and filter out those with length shorter than $2d$. For dilation, we similarly enlarge each dexel, which corresponds to "shrinking the empty space" between them. The pseudo-code for these operations is shown in Algorithm 1 and assumes that the dexels in $l \cap R$ are stored as the ordered list of the dexel endpoints: $p_0, p_1, \ldots, p_{2n-1}$ (see Algorithm 1). We write z_i for the depth (ordinate in direction $\vec{v}$) of endpoint p_i so that we have $z_0 < z_1 < \ldots < z_{2n-1}$.

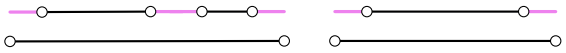
The resulting dexel structure $R' = R \circ s$ is implicitly given by the sequence of reported dexel endpoints in Algorithm 1. Also note that this new dexel structure has at most as many dexels as R.

The time complexity of the algorithm is linear with respect to the number of dexels R. It is also highly parallelizable, as each list of dexels can be processed independently.

Algorithm 1 Morphological operations along a ray. The reported points belong to $\partial(O \circ s)$.



procedure DILATION
  Report point $(p_0 - d\vec{v})$
  for $i = 1$ to $n - 1$ do **do**
    if $z_{2i} - z_{2i-1} > 2d$ **then**
      Report points $(p_{2i-1} + d\vec{v})$ and $(p_{2i} - d\vec{v})$
    end if
  end for
  Report point $(p_{2n-1} + d\vec{v})$



end procedure

procedure EROSION
  for $i = 0$ to $n - 1$ do **do**
    if $z_{2i+1} - z_{2i} > 2d$ **then**
      Report points $(p_{2i} + d\vec{v})$ and $(p_{2i+1} - d\vec{v})$
    end if
  end for



end procedure

4.2. Sequences of segment morphological operations

Given a dexel structure R of a three dimensional input solid O, we want to compute (approximately) a dexel structure of $O \circ \mathcal{Z}$, where $\mathcal{Z} = s_1 \oplus s_2 \oplus \ldots \oplus s_k$. To do so we iteratively apply the segment morphological operations:

$$R_i = R(L_{\Delta_i}, \mathcal{P}(R_{i-1})) \circ s_i \quad \text{for } i = 1 \text{ to } k$$

with $R_0 = R$ (see Section 4.1). At each iteration we generate a surface representation of (the closure of) the current dexel volume $\mathcal{P}(R_{i-1})$ and rasterize it in order to build the next dexel structure (details are given in Section 4.4). The overall process introduces an approximation error which is analyzed in Section 4.3. At the end of the process, $\mathcal{P}(R_k)$ is our approximation of $O \circ \mathcal{Z}$.

4.3. Approximation error

In this section, we indentify a dexel volume with its closure, which is a compact orthogonal polyhedron. In this context, we bound the Hausdorff distance between two orthogonal polyhedra, one being obtained by converting the other into a dexel structure oriented along another arbitrary direction (see Section 4.4).

The Hausdorff distance is transitive, *i.e.* $d_H(A,C) \leq d_H(A,B) + d_H(B,C)$. Thus, by summing the Hausdorff errors in a sequence of segment-offsetting, we obtain a bound on the error between $\mathcal{P}$ and $\mathcal{P} \circ \mathcal{Z}$ for a given zonohedron $\mathcal{Z}$. The theoretical bound is rather crude and therefore would require the sampling rate to increase by $\sqrt{2}$ for each resampling (see the lemma below). A finer analysis is likely to bring the constant $\sqrt{2}$ closer to 1 and in practice we work at the constant screen resolution for viewing or constant manufacturing resolution for the actual manufacturing.

Let $\mathcal{P}_1$ be a dexel volume defined by a dexel structure with direction $\vec{v}_1$ and regular grid cell size Δ_1: $\mathcal{P}_1 = R_1 \oplus \chi_1$ where χ_1 is a square of side length Δ_1, orthogonal to $\vec{v}_1$ and aligned with the dexel grid axes. Let $\mathcal{P}_2$ be defined similarly, where R_2 is the intersection of the lines generated by a regular grid G_2 of cell size Δ_2 along direction $\vec{v}_2$ with $\mathcal{P}_1$: $R_2 = G_2 \cap \mathcal{P}_1$. In other words, R_2 is a sampling of $\mathcal{P}_1$ with a dexel structure in direction $\vec{v}_2$ (see Figure 3 for a two-dimensional illustration).

We say that $\mathcal{P}_1$ is Δ_2-*fat* when $\mathcal{P}_1$ can be expressed as the union of congruent cubes of side length $\sqrt{2}\Delta_2$ with sides parallel to the sides of $\mathcal{P}_1$. In particular, if $\sqrt{2}\Delta_2 \leq \Delta_1$ and no dexel of R_1 has length shorter than $\sqrt{2}\Delta_2$ then $\mathcal{P}_1$ is Δ_2-fat. When $\sqrt{2}\Delta_2 \leq \Delta_1$ but $\mathcal{P}_1$ is not Δ_2-fat, we can enlarge the small dexels of R_1 to make $\mathcal{P}_1$ Δ_2-fat before computing R_2.

Lemma 1. *If $\mathcal{P}_1$ is Δ_2-fat then $d_H(\mathcal{P}_1, \mathcal{P}_2) \leq 1.44\Delta_2$.*

Proof. If no dexel of R_1 has been enlarged, then $R_2 \subset \mathcal{P}_1$, thus $\mathcal{P}_2 = R_2 \oplus \chi_2 \subset \mathcal{P}_1 \oplus \chi_2 \subset \mathcal{P}_1 \oplus \mathbb{B}_{\Delta_2/\sqrt{2}}$:

$$\mathcal{P}_2 \subset \mathcal{P}_1 \oplus \mathbb{B}_{\Delta_2/\sqrt{2}} \quad (1/\sqrt{2} \leq 0.71).$$

If a dexel of R_1 has been stretched to length $\sqrt{2}\Delta_2$ we should account for it. In that case, $R_2 \subset \mathcal{P}_1 \oplus \mathbb{B}_{\sqrt{2}\Delta_2/2}$, which implies

$$\mathcal{P}_2 \subset \mathcal{P}_1 \oplus \mathbb{B}_{\sqrt{2}\Delta_2} \quad (\sqrt{2} \leq 1.42).$$

In the other direction, let p be a point of $\mathcal{P}_1$. Since $\mathcal{P}_1$ is Δ_2-fat, there exists a cube C of side length $\Delta_2\sqrt{2}$ such that $p \in C \subset \mathcal{P}_1$. The cube C contains a ball B of radius $\Delta_2\sqrt{2}/2 = \Delta_2/\sqrt{2}$. When viewed orthographically in the direction $\vec{v}_2$, the ball B is a disk of the same radius. This disk contains a square of side length Δ_2 aligned with the axes of the grid G_2. Therefore, there is at least one line from the sampling pattern G_2 that projects on this disk, which implies the existence of a point $r \in C$ such that $r \in R_2 = G_2 \cap \mathcal{P}_1$.

Let $\alpha = \frac{\sqrt{3}}{\sqrt{2}} + \frac{1}{\sqrt{2}}$ so that $\alpha\Delta_2$ is the sum of the half-diagonal of C and the radius of B. It holds that $C \subset r \oplus \mathbb{B}_{\alpha\Delta_2}$. But $r \oplus \mathbb{B}_{\Delta_2/2} \subset \mathcal{P}_2$, so $p \in C \subset \mathcal{P}_2 \oplus \mathbb{B}_{(\alpha-1/2)\Delta_2}$:

$$\mathcal{P}_1 \subset \mathcal{P}_2 \oplus \mathbb{B}_{(\alpha-1/2)\Delta_2} \quad (\alpha - 1/2 \leq 1.44). \quad \square$$

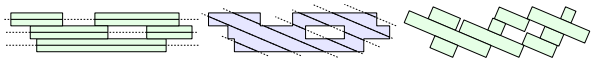


Figure 3: Conversion of a two-dimensional dexel volume *(left)* into another dexel structure along a different direction *(middle)*. The resulting dexel volume *(right)* approximates the initial one.

4.4. GPU implementation

We implement our algorithm entirely on the GPU: from a solid representation, we build a first dexel structure capturing its geometry, a process that we call *dexelization*. We then apply the sequence of line segment morphological operations on the dexel structure. The result of this process is a final dexel structure that can be used for display, direct fabrication, or can be fed back into a CSG engine for further modeling operations.

4.4.1. Dexelization of an object O

Our dexel structure is based on modern implementations [36] of an A-buffer [37]. Given a view direction $\vec{v}$, the object O is rasterized by the GPU using an orthographic projection along $\vec{v}$. Each generated fragment is recorded as an *in* or *out* event, depending on whether it originated from a front facing or back facing triangle with respect to $\vec{v}$. The events are stored in a separate list for each screen pixel. Given a two-manifold (watertight) boundary representation under the form of an indexed face set, the OpenGL rasterization rules [38] ensure that consistent in/out events are generated in every pixel.

After construction each sampling ray of the dexel structure is associated with a sorted list of in/out events. We determine solid intervals by counting the in/out events along the ray. A counter is initialized to zero, decremented when the ray passes an *out* event, and incremented when the ray passes an *in* event. Intervals along the ray that have a positive counter value are considered to be part of the solid and become a dexel.

This counting process elegantly deals with intersecting (closed) meshes, as well as meshes having inner voids. Most importantly, it enables the approach for offsetting polyhedrons described in Section 5.

4.4.2. (Re)dexelization of a dexel volume

As described in Section 4.2, when offsetting along different directions, a new dexel structure has to be constructed by re-sampling the previous dexel volume. In principle, this requires to extract the boundary facets of the first dexel volume, and then to rasterize them into the next dexel structure.

The boundary facets extraction can be implemented very efficiently on the GPU. Unfortunately this approach does not work in practice: the generated mesh contains many T-junctions that produce an artifact known as *cracks* during the rasterization (see Figure 4).

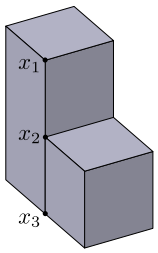


Figure 4: Segments x_1x_2 and x_1x_3 may leave cracks when their incident facets are rasterized.

This is due to numerical imprecisions preventing the rasterizer from properly joining the drawn triangles along the T-junction edges. This has dramatic consequences on the dexel structure as intervals are not properly closed and matter leaks out of the volume. Splitting facets along T-junctions would result in a significantly more complex algorithm, and may not resolve all artifacts.

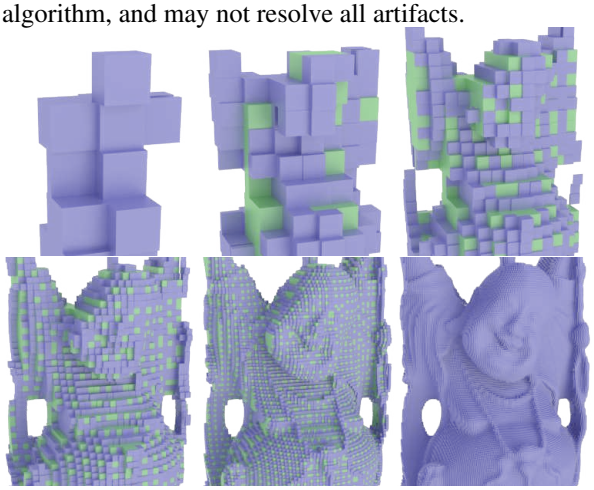


Figure 5: Hierarchical dexel structure. The dexels of the coarser levels are shown in green.

We propose a different implementation that completely eliminates these issues. The idea is to render the dexel structure as a union of box primitives, where each dexel is transformed into a separate, six-sided box. The boxes are slightly enlarged to ensure that no cracks can exist. Applied directly this approach would be impractical, as it would generate a very large amount of fragments in the next dexel structure – exceeding memory requirements and penalizing visualization. We instead produce a hierarchical version of the dexel structure, replacing the common part of four neighboring high resolution dexels by a single lower resolution dexel. Repeated several times, this process quickly groups the inner volumes into large boxes, thereby significantly reducing the number of fragments that have to be rasterized, as illustrated Figure 5. This is implemented in parallel on the GPU, with an algorithm walking four rays at a time and detecting common, coarser intervals.

5. Morphological operations between a zonotope and a polyhedra

In the previous section we discussed a general algorithm applied on dexel structures. While very versatile, its drawback is to impose a re-sampling between each offset direction. As we now discuss, this can be entirely avoided for the special case of polyhedra – the most common representation of solids in CAD/CAM applications.

The salient property of the technique described in this section, is that it avoids any loss of geometric precision until the very final conversion into a dexel structure (a process we call *dexelization*, see §4.4.1). The latter conversion is necessary for visualization, fabrication or further processing. For example the technique affords for interactive pixel-precise *visualization* of the result of dilations or erosions via a perspective dexelization aligned with the virtual camera.

The limitation of the technique is that it cannot mix dilations and erosions in a same sequence. In particular, it cannot be used for computing an exact representation of a morphological closing or opening. If this is desired, then the general dexel algorithm of §4 should be used for the second morphological operation.

5.1. Principle

Let $\mathcal{M}$ be a polyhedron and $\mathcal{Z}$ a zonohedron. This section details a technique to directly compute the dexel structure of the *exact* dilation or erosion of $\mathcal{M}$ by $\mathcal{Z}$, $\mathcal{O} = \mathcal{M} \circ \mathcal{Z}$, without having to compute the polyhedral boundary of $\mathcal{O}$. To do so, we transform the polyhedral boundary of $\mathcal{M}$ into a polyhedral surface S in such a way that the result of the dexelization of S, as described in §4.4.1, is identical to the dexelization of $\mathcal{O}$.

S is a proper manifold without boundary, but it contains a large number of self-intersections and cannot be used to visualize $\mathcal{O}$ without dexelization (see Figure 11).

The core of the idea is to exploit the integer counter of in/out events along the rays. Our algorithm generates S so that additional facets produce pockets of positive or negative matter corresponding to the effect of dilation or erosion along a line segment. The regions where the counter value is positive coincide exactly with the desired dilated or eroded polyhedron.

We focus only on dilation in the rest of this section. Erosion can be computed using complement and dilation: $\mathcal{M} \ominus \mathcal{Z} = \overline{\overline{\mathcal{M}} \oplus \mathcal{Z}}$. The boundary of the complement of a polyhedron is easily obtained by flipping the orientation of its facets.

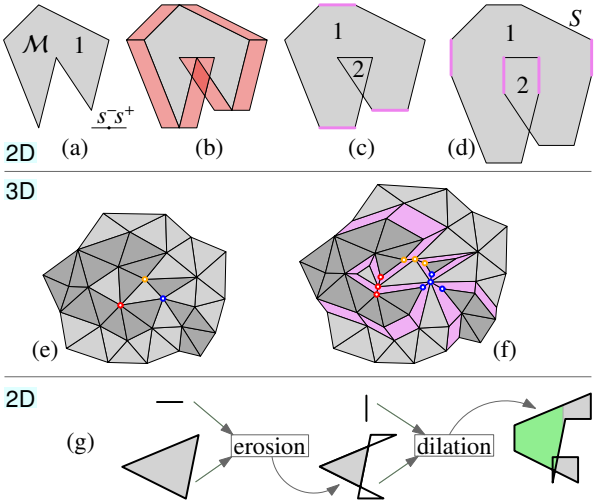


Figure 6: **(a–d)** A 2D example of a polygon undergoing two successive dilations by a segment. *(a)* A polygon and a generator segment. *(b)* A prism is glued (conceptually) to each edge. *(c)* The polygon is simplified. (The numbers indicate the value of the dexelization counter in each region.) *(d)* After dilation by a second, vertical segment. **(e–f)** A 3D illustration of the combinatorics of a piece of a polyhedral surface. *(e)* Dark triangles are back facets. Light triangles are front facets. *(f)* The purple quads are the extrusions of the silhouette edges. Most silhouette vertices are simply duplicated, but colored ones require more copies (see text), so that each connected component of back facets is surrounded by a "collar" of purple quads. *(g)* A 2D counter example: composing our technique on erosion then dilation may give a wrong result (the correct result is shaded green).

5.2. Dilation of a mesh by a segment

We assume that segment s (a generator of zonohedron $\mathcal{Z}$) has its center at the origin and decompose it as $s = s^+ \cup s^-$ where both subsegments share the origin as an endpoint. Our task is to build a surface representation S for $\mathcal{M} \oplus s$. We partition the boundary $\partial\mathcal{M}$ of $\mathcal{M}$ into its sets of back facets B and front facets F with respect to the direction $\vec{s^+}$ from the origin to the other endpoint of s^+. Then, we use the equality

$$\mathcal{M} \oplus s \quad = \quad \mathcal{M} \cup \left(\bigcup_{f \in F} f \oplus s^- \right) \cup \left(\bigcup_{b \in B} b \oplus s^+ \right).$$

Conceptually, we simply add to S the polyhedral boundary of the Minkowski sum of each facet of $\partial\mathcal{M}$ with s^+ or s^- (Figure 6(a–b)). These sums are simple prisms extruded in a direction parallel to segment s. Their facets should be oriented so that their normal vector points outside of the prisms: in this way the dexelization counter shall be properly updated and we are guaranteed that the dexelization of S is equal to that of $\mathcal{M} \oplus s$. This is sufficient to compute the correct dexel structure of $\mathcal{M} \oplus s$. However, in order to continue to dilate $\mathcal{M} \oplus s$ with the other generator segments of $\mathcal{Z}$, while keeping the complexity of the surface S low, we must ensure that only the necessary surface elements are kept in S and that S remains an oriented manifold. We describe these improvements next.

5.3. Dilation of a mesh by a zonohedron

In the polyhedral surface that we obtained above, several pairs of facets do cancel each other: they are geometrically identical but have opposite orientation. These pairs have no effect on the dexelization result and it is preferable to not create them in the first place. They are

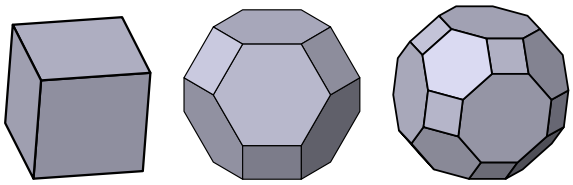
- the extrusions of edges of $\partial\mathcal{M}$ that are *not* silhouette edges with respect to $\vec{s^+}$ so that their adjacent facets are either both front facets or both back facets. (An edge is silhouette w.r.t. to a direction $\vec{d}$ if it has one adjacent front facet and one adjacent back facet w.r.t. $\vec{d}$.)
- the original facets of $\partial\mathcal{M}$; they are "canceled" by a facet of their corresponding prism.

We avoid the creation of these pairs of facets by extruding only silhouette edges and displacing the original facets of $\partial\mathcal{M}$ by $\vec{s^-}$ or $\vec{s^+}$ depending on their front or back status (Figure 6(c–d)). In order to prepare the surface for further dilations with other generator edges, we should make the resulting surface S a manifold as well, so that silhouette edges for other directions can be found correctly. The only problem to obtain a manifold comes from vertices on the silhouette of $\mathcal{M}$ whose adjacent back facets form more than one connected component. A separate copy of such a vertex must be created for each component in order to guarantee that S is manifold. Figure 6(e–f) illustrates how the connectivity of the surface is modified.

Limitations. The technique described above works only as long as we accumulate only dilations or only erosions. In general, applying both operations on S (as would be necessary to compute openings or closings) leads to erroneous results (Figure 6(g)). When a dilated or eroded polyhedron requires further processing, our system dexelizes S and continues the job on as detailed in §4. Another drawback stems from the larger number of fragments generated during dexelization (see Table 2) which might require buffers larger than what the device drivers can allocate.

6. Approximating the ball with zonohedra

Some applications seek to erode or dilate a solid by a ball of radius d. Since our scheme is based on operations



(a) Cube (b) Truncated octahedron (c) Truncated cuboctahedron

Figure 7: Zonohedra approximating $\mathbb{B}_d$, the Euclidean ball of radius d. Their Hausdorff distance to $\mathbb{B}_d$ is (a) $\left(1 - \sqrt{1/3}\right)d \approx 0.422d$, (b) $\epsilon = \left(1 - \sqrt{3/5}\right)d \approx 0.225d$ and (c) $\epsilon = \left(1 - \left(1 + 2\sqrt{2}\right) / \sqrt{13 + 2^{3/2}3}\right)d \approx 0.174d$. The number of generating segments of these zonohedra are respectively 3, 6 and 9.

along line segments, the ball has to be approximated.

The approximation of the Euclidean ball with zonohedra has been extensively studied. Bourgain *et al.* [5] have shown that any n-dimensional zonotope $\mathcal{Z}$ approximating the ball $\mathbb{B}_d$, with a Hausdorff distance error ϵ, has at least $c\epsilon^{-2+6/(n+2)}$ segment generators, where c is a constant depending on the dimension n. Guibas *et al.* [6], introduced an algorithm to find a zonotope enclosing m points with k segment generators and minimizing the sum of the generator lengths, in time $O\left(mk^{n-1} + k^{O(n)}\right)$.

In general, we enforce that the radius of $\mathcal{Z}$ is d. For some applications, it is desirable that the approximate dilation (resp. erosion) is a subset (resp. superset) of the exact one. In this case, we impose that $\mathcal{Z} \subset \mathbb{B}_d$, implying:

$$O \oplus \mathcal{Z} \subset O \oplus \mathbb{B}_d, \quad O \ominus \mathcal{Z} \supset O \ominus \mathbb{B}_d$$

Figure 7 shows some well known zonohedra, and their Hausdorff distance bounds when $\mathcal{Z}$ has radius d. The ball is approximated by a set of generating segments with some length lower than d. For example, in the case of the cube (see Figure 7a), the three generating segments are $\left(d/\sqrt{3}, 0, 0\right)$, $\left(0, d/\sqrt{3}, 0\right)$, and $\left(0, 0, d/\sqrt{3}\right)$. Observe that in general it is not trivial to derive these line segment lengths for an arbitrary zonotope.

The approximated offset is also bounded by these Hausdorff distances:

Lemma 2. $d_H(\mathcal{Z}, \mathbb{B}_d) = \epsilon \implies d_H(O \oplus \mathcal{Z}, O \oplus \mathbb{B}_d) \leq \epsilon$

Proof. Since O, $\mathcal{Z}$ and $\mathbb{B}_d$ are compact sets, $d_H(\mathcal{Z}, \mathbb{B}_d) = \epsilon$ implies that $\mathbb{B}_d \subset \mathcal{Z} \oplus \mathbb{B}_\epsilon$ and $\mathcal{Z} \subset \mathbb{B}_d \oplus \mathbb{B}_\epsilon$. Since dilation is an increasing operator, it comes $O \oplus \mathbb{B}_d \subset O \oplus \mathcal{Z} \oplus \mathbb{B}_\epsilon$ and $O \oplus \mathcal{Z} \subset O \oplus \mathbb{B}_d \oplus \mathbb{B}_\epsilon$, which implies the claimed distance bound. □

The lemma does not hold for erosion as one can find instances of $\mathcal{Z}$ and O such that $d_H(\mathcal{Z}, \mathbb{B}_d) \leq \epsilon$ and $d_H(O \ominus \mathcal{Z}, O \ominus \mathbb{B}_d) \geq 2d - \epsilon$. However, the erosions are close to each other with respect to a kind of "reverse" Hausdorff distance. Namely, the erosion of one by $\mathbb{B}_\epsilon$ is included in the other: $(O \ominus \mathbb{B}_d) \ominus \mathbb{B}_\epsilon \subset O \ominus \mathcal{Z}$ and $(O \ominus \mathcal{Z}) \ominus \mathbb{B}_\epsilon \subset O \ominus \mathbb{B}_d$.

7. Results

In this section we evaluate the performance of the presented algorithms, and the quality of the results. Table 1 displays some statistics on the input models used in our experiments.

Our method allows for arbitrary zonohedra to be used (see Figure 8). Beyond the approximation of the Euclidean ball for regular offsetting, it can be used in NC-machining applications to simulate the tool milling [39] with a shape matching a zonohedron.

The integration of the presented techniques in our ray-based CSG modeler enables complex shapes to be defined. Figure 13 shows a difference between two complex models subsequently undergoing a morphological operation. In Figure 9 we show a more elaborate result defined with several CSG and morphological operations. We are able to combine morphological operations with both methods as long as the mesh-based method operates first (see the filigree model in Figure 1).

7.1. Fabrication quality

Our software typically uses a XY resolution of 50 μm (0.05 mm) for fabrication. The mesh-based offsetting approach results in lossless printout quality. For the GPU-based approach, the precision loss following the successive dexelizations is marginal, as dexelization is always done at printing resolution. Figure 14 shows several results printed on a ZPrinter 450. Observe that the quality of the approximation with respect to offsetting with a ball increases with the number of generating segments for the zonohedra, since the Hausdorff distance decreases (see Figure 12).

7.2. Performance

We carry out our performance tests on an Intel Core i7 4770k with 16GB of memory, and a GeForce Titan Black with 6GB of memory. Performance results are shown in Table 2 and Figure 10. The models and morphological operations used match the figures presented throughout this document.

The performance of the mesh-based approach mainly depends on the radius of the zonohedron and the number of mesh vertices. Its complexity increases as the number of segments for the zonohedron increases, and generates

a large quantity of fragments for the final dexelization step (see Figure 11). Our implementation has a maximum limit of 512MB for storing the fragments (this could in principle be extended up to 2GB). The GPU-based approach complexity is bound by the fabrication resolution.

The performance of our GPU-based method depends on the fabrication resolution and the surface area of the resulting morphological operation. Figure 10 shows that the cost of dilation increases linearly with the number of generated fragments (larger surfaces), while the cost of erosion *decreases* when using larger offsets, which produce smaller surfaces. Unlike the solid offsetting GPU approach of Wang and Manocha [35], the complexity of our GPU-based approach does not rise rapidly as the offsetting distance increases.

	Vertices	Dimensions (mm)	Fragments
Raptor	25080	$64 \times 31 \times 13$	510898
Dancing	19986	$44 \times 24 \times 32$	898600
Dragon	15002	$60 \times 40 \times 27$	755708
2 Lions	100004	$35 \times 61 \times 53$	3775048
Filigree	29872	$70 \times 70 \times 10$	1609138

Table 1: Properties of the input models used for testing. The fragments column displays the number of fragments generated during the dexelization of the input meshes at fabrication resolution (0.05 mm).

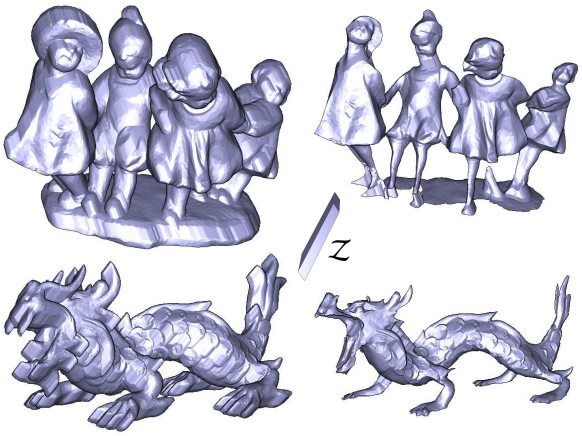


Figure 8: Mesh-based morphological operations between an arbitrary zonohedron $\mathcal{Z}$, induced by three segments with different sizes and some polyhedra. The input shapes are shown in Figure 12.

8. Conclusion

We have presented two complementary mesh-based and GPU-based methods to perform morphological operations with any kind of solid representation in the context of fabrication.

The main advantages of our methods lie in the fact that its performance is not bound by the volume size of the input solid models. They also avoid the complex tracking of topological changes of the offset surface.



Figure 9: Highlighting the small features of a model. We detect the features smaller than the zonohedron $\mathcal{Z}_1$, of radius d_1 of the input shape $\mathcal{M}$, and dilate them by another zonohedron $\mathcal{Z}_2$, of radius d_2, in order to highlight them. That is, we compute the expression $(\mathcal{M} \setminus ((\mathcal{M} \ominus \mathcal{Z}_1) \oplus \mathcal{Z}_1)) \oplus \mathcal{Z}_2$.

Our mesh-based approach ensures lossless representation of the morphological operation but can have high GPU memory requirements during dexelization. Our GPU-based method is more versatile, supports sequences of morphological operations, and has lower GPU memory requirements.

Our technique is well suited to a fabrication context where ray-representations can be directly used for visualization and additive manufacturing (e.g. by direct slicing). It cannot be used to extract boundary representations however. While balls can be well approximated by complex zonotopes, existing techniques might prove more effective if operations with balls of small radius are desired [35].

Our solid extraction method following the offsetting step in the GPU algorithm could be improved to generate a mesh right off of the dexel representation, feeding subsequent offsetting steps in a more direct manner than using a hierarchical dexel structure. This would help to lower the amount of fragments generated for the final fabrication phase.

Finally, tighter approximation error bounds for Lemma 1 could be obtained by also taking into account the dexelization direction and the rotation of the grid of lines L_Δ around $\vec{v}$.

9. Acknowledgment

This work was funded by ERC grant ShapeForge (StG-2012-307877). Jonas Martinez has been partially supported by an ERCIM postdoctoral fellowship.

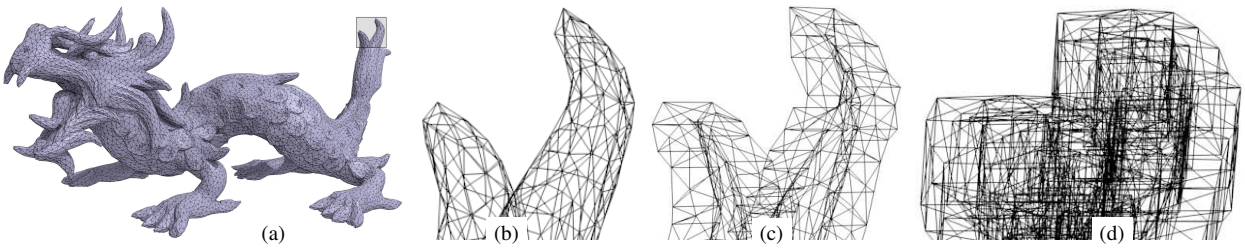


Figure 11: Illustration of the polyhedral surface S, used by the mesh-based approach, whose dexelization gives the erosion of the dragon in Figure 12. *(a)* Input mesh. *(b)* Zoom on the tail. For visualization purposes only the mesh edges are shown. *(c)* Erosion with one segment. *(d)* Erosion with the cube (three segments).

	Cube				Octahedron				Cuboctahedron			
Raptor	Dilation 0.2		Erosion 0.1		Dilation 0.2		Erosion 0.1		Dilation 0.2		Erosion 0.1	
Mesh-based	136 ms	#1.1M	134 ms	#0.8M	203 ms	#1.6M	231 ms	#1.2M	410 ms	#1.8M	557 ms	#1.3M
GPU-based	622 ms	#0.34M	540 ms	#0.3M	1328 ms	#0.31M	1262 ms	#0.26M	1992 ms	#0.36M	1890 ms	#0.31M
Dancing	Dilation 1.75		Erosion 1.75		Dilation 1.75		Erosion 1.75		Dilation 1.75		Erosion 1.75	
Mesh-based	105 ms	#6.6M	109 ms	#7.0M	245 ms	#17.5M	374 ms	#22.2M	461 ms	#23.4M	771 ms	#31.2M
GPU-based	641 ms	#0.54M	614 ms	#0.11M	1859 ms	#0.63M	1312 ms	#0.11M	2583 ms	#0.59M	1855 ms	#0.09M
Dragon	Dilation 2.1		Erosion 1.5		Dilation 2.1		Erosion 1.5		Dilation 2.1		Erosion 1.5	
Mesh-based	114 ms	#10.5M	97 ms	#7.1M	414 ms	#33.5M	333 ms	#24.5M	715 ms	#41.0M	673 ms	#31.7M
GPU-based	995 ms	#0.66M	884 ms	#0.16M	1261 ms	#0.78M	1527 ms	#0.13M	2900 ms	#0.75M	1993 ms	#0.13M
Lions	Dilation 2.2		Erosion 2.2		Dilation 2.2		Erosion 2.2		Dilation 2.2		Erosion 2.2	
GPU-based	1225 ms	#1.4M	835 ms	#0.4M	3983 ms	#1.9M	2501 ms	#0.43M	5608 ms	#1.5M	3511 ms	#0.35M
Filigree	Closing 1.75		Opening 0.98		Closing 1.75		Opening 0.98		Closing 1.75		Opening 0.98	
Mesh+GPU	1416 ms	#1.2M	1148 ms	#0.52M	3916 ms	#1.06M	2389 ms	#0.73M	5974 ms	#1.15M	4334 ms	#1.01M

Table 2: Computation times and number of fragments generated for morphological operations and final dexelization, in milliseconds, with several models. For the filigree model, the closings are achieved by applying a mesh-based dilation followed by a GPU-based erosion of the same amount, and openings by applying a mesh-based erosion followed by a GPU-based dilation.

References

[1] J. Serra, Image analysis and mathematical morphology, Academic Press, Inc., 1983.

[2] J. R. Rossignac, A. A. Requicha, Offsetting operations in solid modelling, Computer Aided Geometric Design 3 (2) (1986) 129 – 148.

[3] T. Maekawa, An overview of offset curves and surfaces, Computer-Aided Design 31 (3) (1999) 165 – 173.

[4] P. Hachenberger, Exact Minkowski sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces, Algorithmica 55 (2) (2009) 329–345.

[5] J. Bourgain, J. Lindenstrauss, Approximating the ball by a Minkowski sum of segments with equal length, Discrete & Computational Geometry 9 (1) (1993) 131–144.

[6] L. J. Guibas, A. Nguyen, L. Zhang, Zonotopes as bounding volumes, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 803–812.

[7] T. Van Hook, Real-time shaded NC milling display, in: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, 1986, pp. 15–20.

[8] L. Zeng, L. M.-L. Lai, D. Qi, Y.-H. Lai, M. M.-F. Yuen, Efficient slicing procedure based on adaptive layer depth normal image, Computer-Aided Design 43 (12) (2011) 1577 – 1586.

[9] H. Pu, C. C. L. Wang, Y. Chen, Intersection-free and topologically faithful slicing of implicit solid, Journal of Computing and Information Science in Engineering 13 (2).

[10] M. Forsyth, Shelling and offsetting bodies, in: Proceedings of the Third ACM Symposium on Solid Modeling and Applications, 1995, pp. 373–381.

[11] R. C. Evans, G. Koppelman, V. T. Rajan, Shaping geometric objects by cumulative translational sweeps, IBM Journal of Research and Development 31 (3) (1987) 343–360.

[12] A. Kaul, J. Rossignac, Solid-interpolating deformations: Construction and animation of PIPs, Computers & Graphics 16 (1) (1992) 107 – 115.

[13] M. Peternell, T. Steiner, Minkowski sum boundary surfaces of 3D-objects, Graphical Models 69 (3–4) (2007) 180 – 190.

[14] M. Campen, L. Kobbelt, Polygonal boundary evaluation of Minkowski sums and swept volumes, Computer Graphics Forum 29 (5) (2010) 1613–1622.

[15] S. F. Frisken, R. N. Perry, A. P. Rockwood, T. R. Jones, Adaptively sampled distance fields: A general representation of shape for computer graphics, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 2000, pp. 249–254.

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

[17] D. Pavić, L. Kobbelt, High-resolution volumetric computation of offset surfaces with feature preservation, Computer Graphics Forum 27 (2) (2008) 165–174.

[18] S.-H. Lee, T. Park, C.-H. Kim, Primitive trees for precomputed distance queries, Computer Graphics Forum 32 (2013) 419–428.

[19] Y. Chen, H. Wang, D. W. Rosen, J. Rossignac, A point-based offsetting method of polygonal meshes, Tech. rep., Georgia Tech University (2005).

[20] J.-M. Lien, Covering Minkowski sum boundary using points with applications, Computer Aided Geometric Design 25 (8) (2008) 652 – 666.

[21] S. Nelaturi, V. Shapiro, Configuration products in geometric modeling, in: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, 2009, pp. 247–258.

[22] S. Calderon, T. Boubekeur, Point morphology, ACM Transactions on Graphics 33 (4) (2014) 45:1–45:13.

[23] W. Li, S. McMains, A GPU-based voxelization approach to 3D

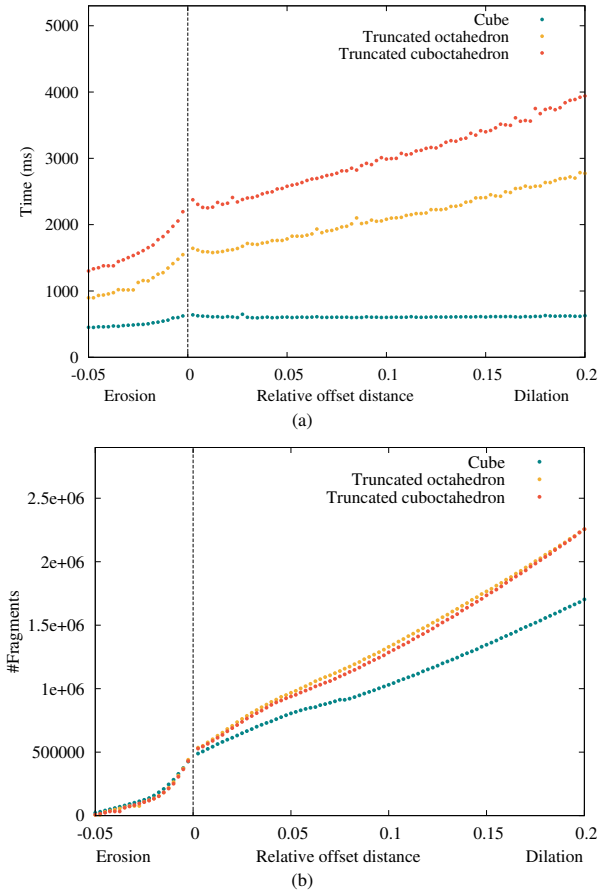


Figure 10: Running times of the GPU-based approach with the dragon model. The abscissa denotes the relative offset distance, which is the offset distance divided by the diagonal length of the model bounding box. The ordinate denotes (a) the running time in milliseconds, (b) the number of fragments of the resulting morphological operation. Please refer to Section 7 for details.

Minkowski sum computation, in: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, 2010, pp. 31–40.

[24] W. Li, S. McMains, A sweep and translate algorithm for computing voxelized 3D Minkowski sums on the GPU, Computer-Aided Design 46 (0) (2014) 90 – 100.

[25] Y.-S. Leung, C. C. L. Wang, Y. Chen, GPU-based super-union for Minkowski sum, Computer-Aided Design and Applications 10 (3) (2013) 475–487.

[26] T. Saito, T. Takahashi, NC machining with G-buffer method, in: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, 1991, pp. 207–216.

[27] J. L. Ellis, G. Kedem, T. C. Lyerly, D. G. Thielman, R. J. Marisa, J. P. Menon, H. B. Voelcker, The ray casting engine and ray representatives, in: Proceedings of the First ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications, 1991, pp. 255–267.

[28] E. Hartquist, J. Menon, K. Suresh, H. Voelcker, J. Zagajac, A computing strategy for applications involving offsets, sweeps, and Minkowski operations, Computer-Aided Design 31 (3) (1999) 175 – 183.

[29] M. O. Benouamer, D. Michelucci, Bridging the gap between CSG and Brep via a triple ray representation, in: Fourth ACM/Siggraph Symposium on Solid Modeling and Applications, 1997, pp. 68–79.

[30] J. Shade, S. Gortler, L.-w. He, R. Szeliski, Layered depth images, in: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, 1998, pp. 231–242.

[31] C. C. Wang, Y.-S. Leung, Y. Chen, Solid modeling of polyhedral objects by layered depth-normal images on the GPU, Computer-Aided Design 42 (6) (2010) 535 – 544.

[32] J. P. Menon, H. B. Voelcker, Mathematical foundations I: Set theoretic properties of ray representations and Minkowski operations on solids, Tech. rep., Cornell University (1991).

[33] K. Hui, Solid sweeping in image space—application in NC simulation, The Visual Computer 10 (6) (1994) 306–316.

[34] Y. Chen, C. C. Wang, Uniform offsetting of polygonal model based on layered depth-normal images, Computer-Aided Design 43 (1) (2011) 31 – 46.

[35] C. C. Wang, D. Manocha, GPU-based offset surface computation using point samples, Computer-Aided Design 45 (2) (2013) 321 – 330.

[36] M. Maule, J. L. Comba, R. P. Torchelsen, R. Bastos, A survey of raster-based transparency techniques, Computers & Graphics 35 (6) (2011) 1023 – 1034.

[37] L. Carpenter, The A-buffer, an antialiased hidden surface method, in: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, 1984, pp. 103–108.

[38] K. A. Mark Segal, The OpenGL graphics system: a specification (version 4.2) (2011).

[39] M. Inui, Fast inverse offset computation using polygon rendering hardware, Computer-Aided Design 35 (2) (2003) 191 – 201.

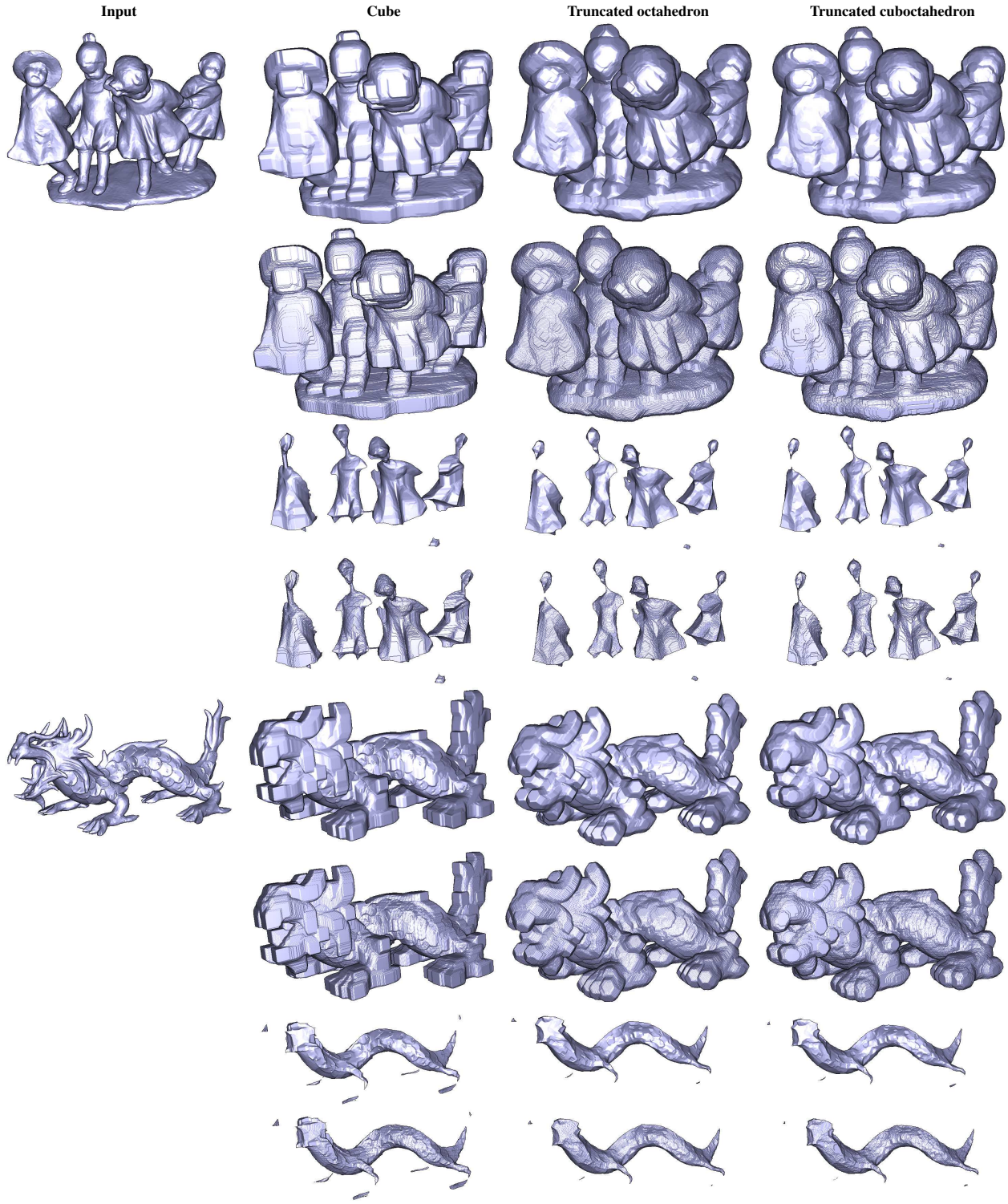


Figure 12: Morphological operations between a zonohedron that approximates the Euclidean ball and polyhedra. The first column shows the input models. The next columns show the dilation/erosion with three different zonohedrons. The odd rows correspond to the mesh-based approach, and the even rows to the GPU-based approach. The dilation/erosion sizes match the ones in Table 2.

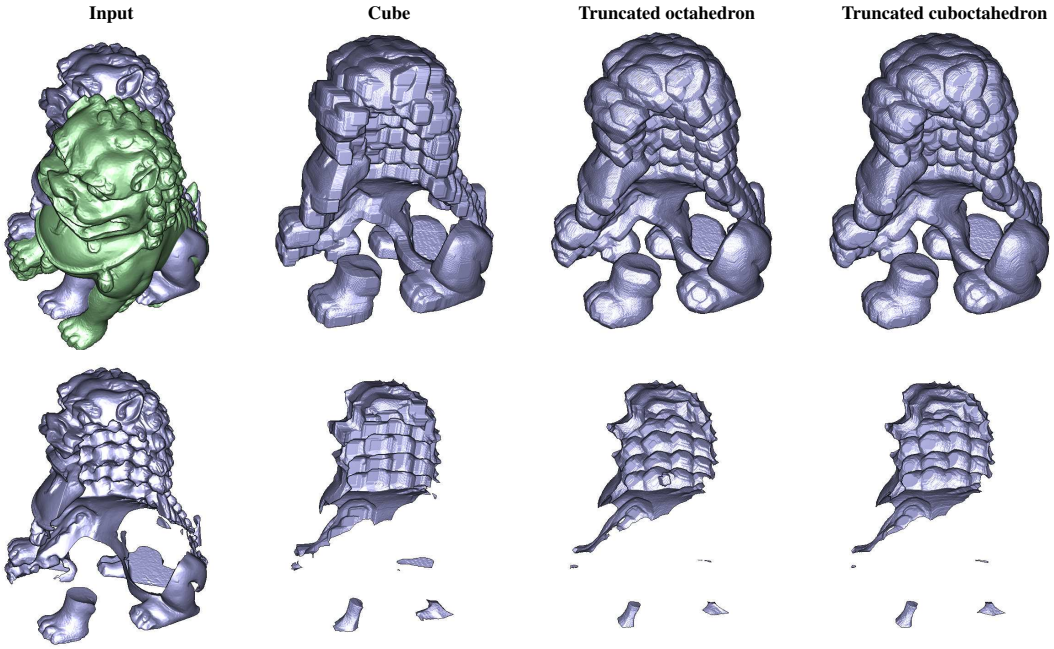


Figure 13: GPU-based morphological operations between a zonohedron that approximates the Euclidean ball, and a CSG scene. The CSG scene is the difference between two rotated lion models. The first column shows the input models, and the result of the intersection below. The next three columns show the dilation/erosion with three different zonohedrons.



Figure 14: Some 3D printed models obtained by our method. (a) The lion model slightly eroded. (b) The complex CSG intersection and dilation of Figure 13, when considering the truncated cuboctahedron. (c) Section cut of the difference between the original buste model and its erosion. (d) The closing of the filigree model with a cube. (e) The original dragon (left) and the dilated dragon (right) of Figure 12. (f) An eroded dragon.