Bounding Proxies for Shape Approximation

STÉPHANE CALDERON and TAMY BOUBEKEUR, LTCI, Telecom ParisTech, Paris-Saclay University

Many computer graphics applications use simpler yet faithful approximations of complex shapes to conduct reliably part of their computations. Some tasks, such as physical simulation, collision detection, occlusion queries or free-form deformation, require the simpler proxy to strictly enclose the input shape. While there are algorithms that can output such bounding proxies on simple input shapes, most of them fail at generating a proper coarse approximant on real-world complex shapes, which may contain multiple components and have a high genus. We advocate that, before reducing the number of primitives to describe a shape, one needs to regularize it while maintaining the strict enclosing property, to avoid any geometric aliasing that makes the decimation unreliable. Depending on the scale of the desired approximation, the topology of the shape itself may indeed have to be first simplified, to let the subsequent geometric optimization be free from topological locks.

We propose a new bounding shape approximation algorithm which takes as input an arbitrary surface mesh, with potentially complex multi-component structures, and generates automatically a bounding proxy which is tightened on the input and can match even the coarsest levels of approximation. To sustain the nonlinear approximation process that may eventually abstract both geometry and topology, we propose to use an intermediate regularized representation in the form of a shape closing, computed in real time using a new fast morphological framework designed for efficient parallel execution. Once the desired level of approximation is reached in the shape closing, a coarse, tight and bounding polygonization of the proxy geometry is extracted using an adaptive meshing scheme. Our underlying representation is both geometry- and topology-adaptive and can be optionally controlled accurately by a user, through sizing and orientation fields, yielding an intuitive brush metaphor within an interactive proxy design environment. We provide extensive experiments on various kinds of input meshes and illustrate the potential applications of our method in scenarios that benefit greatly from coarse, tight bounding substitutes to the actual high resolution geometry of the original 3D model, including freeform deformation, physical simulation and level of detail generation for rendering.

CCS Concepts: •**Computing methodologies** →**Mesh models; Mesh geometry models; Shape analysis;** *Volumetric models;*

Additional Key Words and Phrases: shape approximation, bounding volumes, mathematical morphology, geometric simplification, proxy

ACM Reference format:
Stéphane Calderon and Tamy Boubekeur. 2017. Bounding Proxies for Shape Approximation. *ACM Trans. Graph.* 36, 4, Article 57 (July 2017), 13 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073714

1 INTRODUCTION

When facing complex 3D shapes, numerous geometric and spatial algorithms rely on simplified versions of these shapes to maintain their scalability. These algorithms infer the existence of a *proxy* to produce either a faster result and/or a good enough approximate solution to the problem they aim at solving (e.g., conservative collision-detection or visibility tests). This is usually achieved by formulating and solving the problem with the *proxy* before mapping the result to the original shape (e.g., collision response or drawing call). For a number of applications, the coarse proxy is not only expected to be made of a minimal number of primitives, but also to completely enclose the original shape. Such a *bounding* proxy takes the form of a coarse, closed 2-manifold triangle mesh.

On one hand, the automatic construction of bounding proxies is tedious, as optimizing the few degrees of freedom of its geometry may require highly non-linear energies to recover the bounding property while remaining a faithfull i.e., tight approximation of the input. On the other hand, the interactive design of such objects is even more challenging, as the user needs a form of real time feedback on the proxy geometry while exploring the space of possible bounding approximations. Ideally, the user shall only focus on the (adaptive) level-of-detail of the proxy, letting the generation process preserve the bounding property and minimal polygonal resolution.

Bounding Proxies. Although highly dependent on the application, ideal bounding proxies share a number of desirable properties that make them attractive for multiple scenarios. Each application then emphasizes on some (or all) of these properties:

- **bounding:** crucial for *conservative* collision-detection and occlusion queries, but also for artifact-free motion transfer like cage-based free form deformation (FFD) and physical simulation, where the compute-intensive physical interactions are solved on the *proxy* and the resulting motion is then transferred back to the original input shape,
- **coarser:** a high vertex count in the proxy damages its utility, requiring the system (or user) to manipulate more elements to reach its purpose; ideally, the proxy should have the minimum number of vertices to achieve its goal,
- **tight:** although bounding, it is important to locate the proxy geometry as closely as possible to the mesh to prevent over-conservativity and ultimately cause more computation or memory usage than needed,
- **geometry adaptive:** the proxy structure and resolution should adapt to the geometry of the mesh, reflecting its main components, dominant features and, to a certain extent, topological structures,
- **user adaptive:** the proxy structure, topology and resolution should also account for a user-driven control mechanism, specifying at which scale she intends to use it, remaining coarser on some regions and finer on others; this also translates into topological variations, where the proxy may have a simpler topology than the input object wherever the scale is larger than a particular topological structure (e.g., small tunnel, handles, multiple components),

This work is partially supported by the French National Research Agengy (ANR) under grant ANR 16-LCV2-0009-01 ALLEGORI and by BPI France, under grant PAPAYA.
This is the authors draft version. Resources such as data, source code and supplemental materials will be posted upon final publication date on the project webpage: https://tsi.telecom-paristec.fr/cg, in the "Bounding Proxies" publication section.
The final version will be published in ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017). Go on www.acm.org for accessing the final version with the references recalled below.

DOI: http://dx.doi.org/10.1145/3072959.3073714

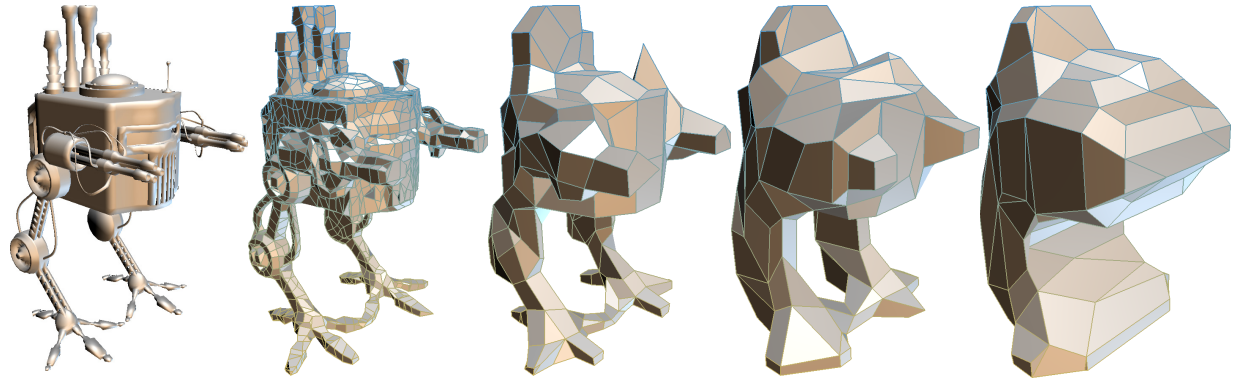


Fig. 1. Starting from a complex mesh (56 components, genus 49, left), our approximation algorithm generates adaptive bounding proxies with both geometry and topology evolving as the desired scale grows (middle left to right). This scale can optionnally be interactively controlled with a scale field.

While previous methods usually rely entirely on a combinatorial or numerical optimization procedure, we argue that these objectives can be achieved thanks to an intermediate representation that models them explicitly: we adopt a *morphological* approach to achieve these properties (Sec. 3). Specifically, we define a simpler yet faithful domain from the input object, in which the proxy is meshed. We design this domain to fulfil the aforementioned desirable properties by construction, using a *spatially varying morphological closing* to obtain a simpler and tight bounding shape. The spatially varying behaviour of this closing offers an intuitive interactive control for the user, who can either prescribe a single scale value for the entire proxy and let the approximation procedure run automatically, or design finely the scale of the proxy at a given location in space to force the algorithm to coarsify or refine it, using an intuitive brush metaphor to rule the underlying scale field interactively.

Granting the closing operator guarantees both the bounding and tightness properties. It also supports *asymmetric* structuring elements i.e., two structuring elements of different shapes can be used for the dilation and erosion steps of the closing. In particular, we show that a cube for dilation, together with a sphere for erosion, while tightening closely the proxy to the original object, give rise, at the same time, to large flat areas, which ease the generation of large coarse(r) polygons in a final proxy structure that maintains the enclosing property (see Fig. 5). By widening the possible shape abstractions, this asymmetric morphological filtering also better perserves salient structures already present in the input (see Fig. 9) and mimics the box-like aspect one can observe in hand-crafted proxies (see Fig. 19).

With this specific closing domain in hand, we provide a feature-aware meshing strategy which generates the (coarse) proxy polygons while preserving the aforementioned properties, the strict-enclosing property being fullfilled in practice and letting foresee a possible formal guarantee, as we discuss in Sec. 6. As a result (Sec. 4), our method generates bounding proxies entirely automatically at any resolution (see Fig. 1) and also allows users to model them interactively, providing them with a high level control over the local scale the proxy shall reflect, before extracting a mesh structure which is ready for the target application.

We illustrate the suitability of our tight and coarse bounding proxies for various applications, including physical simulations, freeform deformations and level-of-detail (LoD) generation (Sec. 5).

Contributions. We propose the following original contributions:

(1) a new shape approximation model based on an *asymmetric morphological closing*, coupled with spatially varying scale and orientation, which comes with remarkable properties that are not achieved with former methods, such as topology simplification and tightness at the same time;
(2) a new GPU algorithm which, for the first time, offers the ability to perform a high resolution morphological alteration at interactive rates;
(3) a fully-featured bounding shape approximation algorithm which combines these new pieces with state-of-the-art voxelization and constrained mesh simplification, able to handle any input 3D shape as long as a coherent inside/outside space partition can be inferred from it.

To our knowledge, our approach is the first to introduce morphological closings for the purpose of shape approximation. While being instrumental for coarse bounding proxies, this morphological regularisation appears to be useful for an even wider range of applications, as we discuss in the end of the manuscript.

2 PREVIOUS WORK

Mesh Simplification. An impressive body of work covers the process of simplifying a mesh by reducing its polygon count while preserving as much as possible its overall shape. Such methods range from progressive decimation [Garland and Heckbert 1997; Hoppe et al. 1993; Thiery et al. 2013], to variational partitioning [Cohen-Steiner et al. 2004; Mehra et al. 2009] and spatial clustering [Legrand and Boubekeur 2015; Lindstrom 2000; Rossignac and Borrel 1993]. They usually differ in their geometric fidelity, speed and scalability. However, they do not focus on producing *bounding* proxies i.e., the bounding property is not accounted for and the behavior of some of them may not even be guaranteed when operating at coarse scale, where intricated geometry and complex topology cannot be properly captured by the local model of the algorithm (e.g., planes, spheres, quadrics, see Sec. 4).

Convex representations. Widely used in *conservative* collision detection and occlusion queries, these representations approximate the input shape by a convex enclosing shape, including spheres, (oriented) bounding boxes or k-dops for instance. They are basically different approximations of the convex hull i.e., the unique convex object of minimal volume enclosing the input shape. While efficient to generate and process, their inherent convexity becomes a bottleneck on highly non-convex shapes, for which many collision/intersection computations could be avoided with a non-convex *proxy* containing a few additional primitives. Although, for some applications, one can decompose the input shape into clusters, approximate each of them with a convex primitive and organize the resulting set in a hierarchical data structure, bounding proxies remain more general and applicable to a wider set of scenarios. Alpha shapes [Edelsbrunner et al. 1983] extend convex hulls by localizing the notion of convexity, but retain arbitrarily high resolution in locally convex regions of the input, which discards them from being good candidates for coarse proxy generation.

Cage generation methods. Bounding proxies are often used in conjonction with space coordinates to perform freeform deformation transfer. They are often referred to as "cages" in such scenarios. Several methods [Ben-Chen et al. 2009; Shen et al. 2004; Xian et al. 2009] rely on *offsets* to compute bounding proxies: given an iso-contour of the input, they extract its *offseted* version and compute a low resolution mesh of its boundary. These methods share the same drawback as those relying on *dilated* version of the input [Faraj et al. 2012]: the more the input is simplified, the looser the resulting proxy is, limiting the extent of a practical simplification (see Sec. 4). Contrary to *dilation*, if the input iso-contour is not a proper distance field, an *offseting* operator does not prevent the creation of new spurious structures [Jackway and Deriche 1996]. Even though, most of these methods use a polygon decimation process to reduce the actual number of primitives, the looseness of the simplified version helps in producing a strict-enclosing and self-intersection-free proxy.

Others methods [Deng et al. 2011; Sander et al. 2000] rely on the direct simplification of the input to generate bounding meshes using progressive edge-collapse algorithms and ensuring a local strict-enclosing property by including local linear constraints to the optimization performed during the simplification. These methods, which do not necessarily intend to produce FFD-ready proxies, offer limited quality and regularity in their output, exhibiting spikes and favoring shape approximation (minimal error to the input) over structure simplicity.

Xian et al. [2012] generate automatic bounding meshes by distributing *oriented bounding boxes* (OBBs) over the input mesh, optimizing and subsequently merging them. During the merging process, if OBBs fail at being fused, the system fallbacks to boolean operations (unions) which may lead to unintended local complex structures in the bounding mesh connectivity, and differs significantly from typical hand-designed cages in the context of freeform deformation. Similarly, Le and Deng [2017] let the user manually place cuts on the input mesh for every section (i.e., planar edge loop) of the intended cage structure and mesh the resulting set of quads using an inter-quad projection.

With *Nested Cages*, Sacht et al. [2015] recently demonstrated high quality bounding mesh creation for a variety of shapes. Given a self-intersection free decimated version of the input, they flow the original shape into the decimated version, and reinflate it back while solving for all the collisions on the decimated proxy. The proxy is effectively pushed back by the reinflating the input shape until it is restored at its exact original state. As a result, the proxy ends up strictly enclosing the input shape and free of self-intersections i.e., it is a proper *cage* for freeform deformation. We conduct a large part of our comparison experiments against this approach which can currently be considered as the state-of-the-art. (see Sec. 4).

Mathematical Morphology. Mathematical morphology (MM) is a non-linear shape analysis framework which measures how shape components get (dis-)connected or disappear when swept with a specific inspection element, called the *structuring element* (or SE). This framework [Serra 1983] is particularly efficient at processing 2D/3D binary grids modelling space occupancy. Specifically, let $\mathcal{G}$ be a binary voxel grid discretizing a given object and $\mathcal{B}$ the binary voxel grid of the SE (typically a sphere or a cube). Then, the two fundamental operators of MM – namely the dilation D and erosion E – are defined as:

$$D_{\mathcal{B}}(\mathcal{G}) = \mathcal{G} \oplus \mathcal{B} \quad \text{and} \quad E_{\mathcal{B}}(\mathcal{G}) = \mathcal{G} \ominus \mathcal{B}$$

with $\oplus$ (reps. $\ominus$) the Minkowski sum (resp. subtraction) operator. Chaining these operators yields advanced morphisms affecting both the geometry and the topology of $\mathcal{G}$, with in particular the morphological opening O and closing C:

$$O_{\mathcal{B}} := D_{\mathcal{B}} \circ E_{\mathcal{B}} \quad \text{and} \quad C_{\mathcal{B}} := E_{\mathcal{B}} \circ D_{\mathcal{B}}.$$

Giving a full survey of MM is beyond the scope of this paper and we refer the reader to the book of Najman and Talbot [2010] for a complete introduction and to the work of Calderon and Boubekeur [2014] for a recent study in the context of point-based modeling. As we will make use of a spatially varying SE, our approach also relates to the work of Söderström and Museth [2010] who use spatially adaptive morphological filters to track interfaces in fluid simulations.

Morphological closings have been used to process scalar fields on meshes [Rössl et al. 2000]. However, to our knowledge our method is the first to exploit morphological closings to process surface meshes in general – and more specifically to compute tight and bounding proxies. Based on the work of El-Sana and Varshney [1998], one could examine the existence of a theoretical link between an alpha shape and a closing with a sphere of radius alpha. However, beyond the previously mentioned inherent limitations of such methods in terms of resolution reduction, alpha shapes provide only approximations of simple spherical closings and do not provide the versatility of a morphological closing, which gives a clean theoretical and practical foundation to spatially varying and exotic shape alterations, as we will discuss later with our asymmetric instance. Moreover, alpha shapes computations are inherently based on a Delaunay triangulation which prevents their use at high input resolution and under interactive constraints. Finally, the balance between the key properties of bounding proxies, namely the bounding constraint and the target coarse resolution polygonization, neither is addressed nor easily achievable with existing alpha-shape transformations.

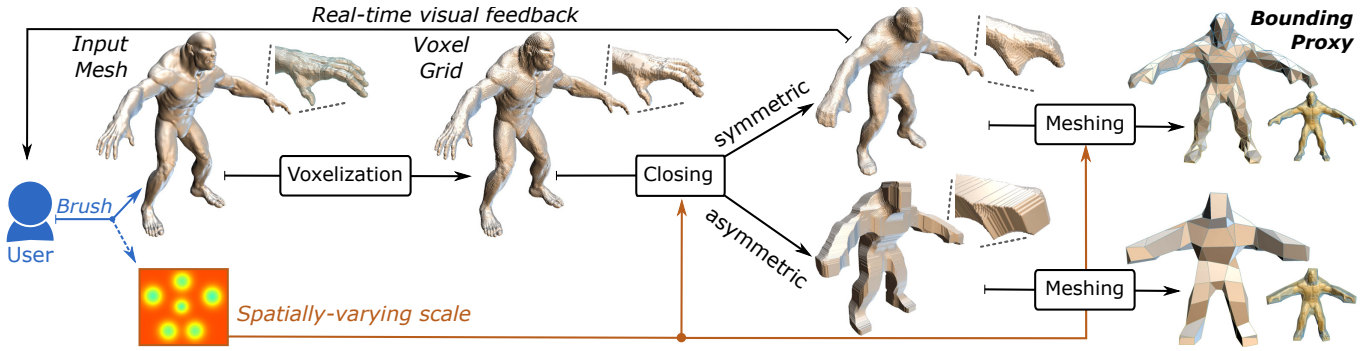


Fig. 2. **Overview:** our bounding shape approximation algorithm uses a morphological closing to decorrelate the geometric and topological simplification from the actual mesh decimation. The bounding proxy geometry can be **optionally** controlled adaptively with a scale field, typically tailored interactively with a virtual brush, that enforces the proxy scale. As our new hierarchical collision algorithm closes the shape in realtime, this closing also provides an interactive feedback. When the polygon count matters more than the approximation quality, an asymmetric closing is performed instead of a symmetric one.

Efficient Spatially Varying Morphology. There exist many algorithms to compute morphological operations on grids. However, while several can compute dilations/erosions efficiently [Gil and Werman 1993; Gil and Kimmel 2002; van Herk 1992], or even in parallel [Boulos et al. 2012; Hopf and Ertl 2000; Jagannathan et al. 2014; Museth 2013; Thurley and Danell 2012] with a separable SE, or an arbitrary one [Soille et al. 1996], they cannot be generalized to a *spatially varying* framework, where the size of the SE (i.e., scale of the morphism) changes over space. Some efficient methods exist for a non-uniform SE but are limited in the choice of the SE (balls [Cuisenaire 2006] or rectangles [Hedberg et al. 2009]) and do not provide parallel-scalablility. In constrast, we propose a method that computes in parallel (e.g., on the GPU) a *spatially varying* dilation (resp. erosion) with an arbitrary (resp. spherical) SE, fast enough to provide interactive feedback to users. We refer to Bouaynaya et al. [2008] for more formal elements on spatially-varying mathematical morphology.

3 METHOD

3.1 Overview

Our algorithm constructs a bounding proxy C from an input mesh $\mathcal{M}$ as follows (see Fig. 2):

(1) **voxelization:** a multi-resolution voxel grid $\mathcal{G}$ is built using a fast, state-of-the-art 3D rasterization of $\mathcal{M}$, followed by a bottom-up mip-map construction,
(2) **closing:** given a 3D scale field $\mathcal{S}$, a spatially varying closing is computed on $\mathcal{G}$, using $\mathcal{S}$ to tailor the size of the two structuring elements of this morphological alteration, namely a sphere (resp. a cube) to dilate $\mathcal{G}$ isotropically (resp. giving rise to cuboidal structures), and a sphere to erode the dilation and tighten $\mathcal{G}$ to $\mathcal{M}$,
(3) **meshing:** the coarse proxy C is meshed by (i) densely meshing $\mathcal{G}$, (ii) simplifying this mesh with a progressive edge-collapse that preserves local bounding conditions and upper bounded to the local scale modeled by $\mathcal{S}$.

The morphological operators are computed with a new fast algorithm that enables real-time feedback on the bounding proxy geometry through the dense closing visualization.

3.2 Voxelization

We typically consider closed 2-manifold, multi-components meshes as the input, although any shape representation providing a clear inside/outside indicator function can be used with our approach. To achieve high performance morphological filtering, we base our framework on a discrete voxel grid $\mathcal{G}$ that is generated from the input mesh on GPU using the conservative voxelization method proposed by Schwarz et al. [2010]; up to the voxel size, small holes in the input are handled at this stage. We pack the resulting 3D binary grid by coding a cube of 8 voxels onto a single byte (uchar8). As the resolution of this grid is then inherited by the upcoming closing and meshing stages, we choose it dense enough to capture the feature structures that may rise from our closing (see the close-ups in Fig. 2). However, when the scale field is chosen coarse enough everywhere, this resolution may be diminished strongly. In practice, we use 512^3 as the initial resolution. In any case, since the voxelization is running in real time, this resolution may be changed dynamically if necessary. The resulting grid $\mathcal{G}$ is used for two purposes: first as the base input of the morphological transformation yielding the proxy geometry (see Sec. 3.3) and second, while being morphologically transformed, as an interactive visual feedback.

3.3 High Speed Closing

Mathematical morphology provides us with the perfect tool to build a coarser, bounding yet tight geometry of $\mathcal{G}$: *closings*. Just as a simple dilation, the closing operator guarantees the bounding condition expected for our proxy geometry. Moreover, by adjusting the size of the structuring element to the local scale value, a closing naturally simplifies the topology of $\mathcal{G}$ when this desired scale becomes too large to maintain fine handles and tunnels. However, contrary to the dilation, it maintains the final geometry close to the input wherever possible, leading to a tight enclosing. Specifically, we define our closings as:

$$\mathcal{G} := E_{\alpha_S} \circ D_{\beta_S}(\mathcal{G})$$

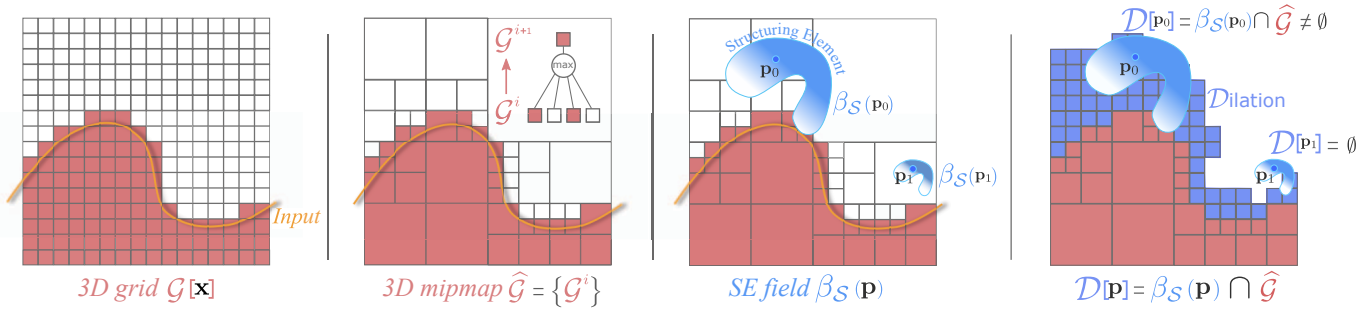


Fig. 3. **Dilation by hierarchical collision.** Left: voxelized input grid $\mathcal{G}$. Middle-left: creation of the 3D mipmap $\hat{\mathcal{G}} = \{\mathcal{G}^i\}$. Middle-right: Structuring Element (SE) field $\beta_{\mathcal{S}}(\mathbf{p})$. Right: at each voxel $\mathbf{p}$, the collision test between $\beta_{\mathcal{S}}(\mathbf{p})$ and the hierarchy $\hat{\mathcal{G}}$, setting $\mathbf{p}$ to 1 if a collision occurs.

Fig. 4. **Closings.** Staring from the Beast model (left), a symmetric closing (middle) with a spherical SE and an asymmetric closing (right) are generated. The latter is tight while exhibiting large flat regions later captured in the coarse proxy meshing stage.

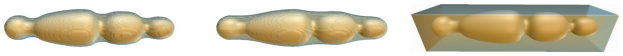


Fig. 5. **Asymmetry.** A symmetric closing (blue shell) with small (left) and larger (middle) scale, and an asymmetric closing (right). Note how with the asymmetric operator the blobby figure is perfectly abstracted to the simplest bounding shape.

with $\alpha_{\mathcal{S}}$ (resp. $\beta_{\mathcal{S}}$) a spherical (resp. spherical *or* cubical) structuring element of varying size $\mathcal{S} : \mathfrak{R}^3 \rightarrow \mathfrak{R}^+$, a positive 3D scale field. To preserve the bounding condition, we assume $\alpha_{\mathcal{S}} \subseteq \beta_{\mathcal{S}}$ in the rest of the manuscript.

Asymmetry. While conventional *closings* use a spherical SE for both the dilation and the erosion (see Sec. 2), we designed our algorithm to support *asymmetric closing*, allowing the use of other geometric primitives (e.g., a cube) for the dilation (see Fig. 3) and keeping a sphere for the erosion (see Fig. 6). This design choice is motivated by the final format of the proxy – a coarse polygonal mesh – for which, under bounding conditions, having large flat areas enables a lower resolution mesh structure at a moderate volume cost (see Fig. 4 and Fig. 5 for illustrations of this phenomenon). This also allows exploring a wider range of shape abstractions in the proxy with box-like features perserved (Fig. 9) for instance and accomodates some specific applications (e.g cage-based deformation in Fig. 19). Since we use this spatially varying closing of $\mathcal{G}$ as the main visual feedback for the user, we formulate it through a parallel algorithm designed for efficient GPU execution, expressing the dilation/erosion operators through a *hierarchical collision* of the structuring elements onto the mipmapped grid $\mathcal{G}$. In the following paragraphs, we describe the successive steps to compute the spatially varying closing through two pyramid constructions (and collisions) interleaved by a contour extraction.

Hierarchy on $\mathcal{G}$. Given the binary grid $\mathcal{G}$, we build a pyramid $\hat{\mathcal{G}} = \{\mathcal{G}^i\}$ with $\mathcal{G}^i : N_i^3 \mapsto \{0,1\}$ and $N_i = \frac{N_0}{2^i}$. We ensure the inclusive property (i.e., bounding behaviour) of the hierarchy with the following construction rule:

$$\forall \mathbf{p} \in N_i^3 \quad \mathcal{G}^i[\mathbf{p}] = \max_{\mathbf{t} \in \{0,1\}^3} \mathcal{G}^{i-1}[2\mathbf{p} + \mathbf{t}] \tag{1}$$

with, $\forall \mathbf{p} \in N_i^3, \mathcal{G}^0[\mathbf{p}] = \mathcal{G}[\mathbf{p}]$.

Spatially Varying Dilation $\mathcal{D}$. Let's consider $v = (i, \mathbf{q})$ a node of $\hat{\mathcal{G}}$ encoding a level $i \in \mathbb{N}$, a position $\mathbf{q} \in \mathbb{N}^3$ and a value $\hat{\mathcal{G}}[v] = \mathcal{G}^i[\mathbf{q}] \in \{0,1\}$ and let $\boxed{v} = \{\mathbf{q} + \mathbf{y}, \mathbf{y} \in [0, 2^i]^3\}$ be the spatial cell representing v. We perform a parallel spatially varying dilation through the hierarchical collision summarized in Alg. 1 and illustrated in Fig. 3. This collision algorithm between a hierarchy $\hat{\mathcal{G}}$ and a field of structuring elements ($\beta_{\mathcal{S}}$) works for any type of SE, even though we only use it here with the simple field: $\beta_{\mathcal{S}}(\mathbf{p}) = \{\mathbf{p} + \mathbf{y}, \mathbf{y} \in [\ \mathcal{S}[\mathbf{p}], \mathcal{S}[\mathbf{p}]]^3\}$ for a cubic SE or $\beta_{\mathcal{S}}(\mathbf{p}) = \{\mathbf{p} + \mathbf{y}, \|\mathbf{y}\| < \mathcal{S}[\mathbf{p}]\}$ for a spherical one.

Hierarchy on $\mathcal{D}_c$. Once $\mathcal{D}$ is computed, we extract its 6-connected contour $\mathcal{D}_c$. Then, given $\mathcal{S}$ and $\mathcal{D}_c$, we build a scale-augmented pyramid $\widehat{\mathcal{D}_c} = \{\mathcal{D}_c^i\}$ with $\mathcal{D}_c^i : N_i^3 \mapsto \{0,1\} \times \mathbb{R}$. We ensure the inclusive property of $\widehat{\mathcal{D}_c}$ with the following construction rule:

$$\forall \mathbf{p} \in N_i^3 \quad \mathcal{D}_c^i[\mathbf{p}] = \max_{\mathbf{t} \in \{0,1\}^3} \mathcal{D}_c^{i-1}[2\mathbf{p} + \mathbf{t}] \tag{2}$$

with $\forall \mathbf{p} \in N_i^3 \ \ \mathcal{D}_c^0[\mathbf{p}] = (\mathcal{D}_c[\mathbf{p}], \mathcal{S}[\mathbf{p}])$ and the max operator being applied *separately* for the binary occupation and the scale. Note that working on the contour $\mathcal{D}_c$ instead of $\mathcal{D}$ reduces the number of cells to visit during the traversal of the hierarchy (less occupied cells) and does not change the final result (as we initialize the grid $\mathcal{E}$ with $\mathcal{D}$).

Spatially Varying Erosion $\mathcal{E}$. Given a node $v = (i, \mathbf{q})$ and its value $\widehat{\mathcal{D}_c}[v] = \mathcal{D}_c^i[\mathbf{q}] = (b, s)$, we define $\boxed{\boxed{v}}$, the *dilation* of $\boxed{v}$ with a sphere of size s. The spatial cell of a *voxel* $\mathbf{p}$ is defined by $\boxed{\mathbf{p}} = \{\mathbf{p} + \mathbf{y}, \mathbf{y} \in [0,1]^3\}$. To perform the second step of the closing operator,

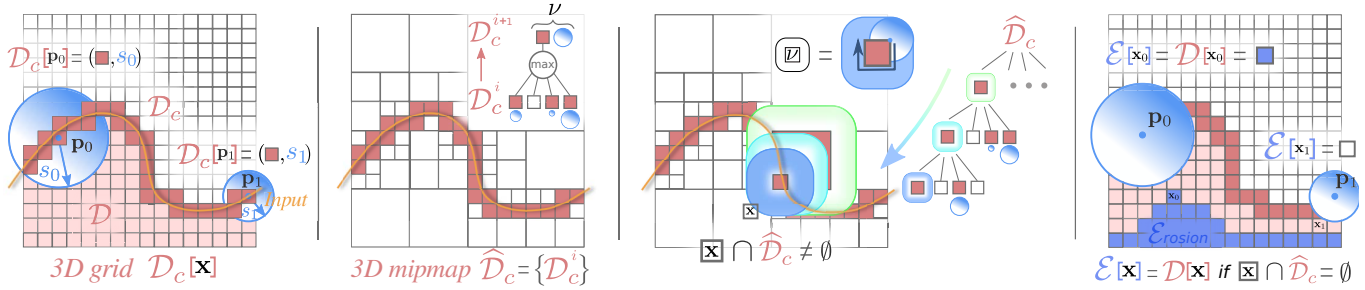


Fig. 6. **Spherical erosion by hierarchical collision.** Left: input $\mathcal{D}$ (i.e., the previous dilation grid) with its contour $\mathcal{D}_c$ that contains both occupation and scale information from the scale field $\mathcal{S}$; here, we exemplify the scale along with the corresponding spherical SE for some voxels of the contour $\mathbf{p}_0$ and $\mathbf{p}_1$. Middle-left: creation of the 3D contour mipmap in which the *max* operator takes into account both the occupation and scale. Middle-right: a collision test between a voxel $\mathbf{x}$ and the hierarchy $\mathcal{D}_c$; we also show, for a node v of the hierachy, the dilated node $\boxed{\mathbb{V}}$, computed from the cell $\boxed{v}$ and used during the hierarchy traversal. Right: finally, a voxel $\mathbf{x}$ inheritates the occupation value $\mathcal{D}[\mathbf{x}]$ from $\mathcal{D}$ if it does not collide with the hierarchy $\mathcal{D}_c$ (set to 0 otherwise).

Algorithm 1 Parallel spatially varying dilation.

Require: $\mathcal{G}$ a binary grid ▷ voxelized input
Require: $\hat{\mathcal{G}}$ a pyramid constructed using Eq. 1
Require: $\mathbf{r} \in \{0,1\}^3$
  $\mathcal{D} \leftarrow \mathcal{G}$
  for all *voxel* $\mathbf{p} \in N_i^3 \mid \mathcal{D}[\mathbf{p}] = 0$ in parallel **do** ▷ outside nodes
    $\tau \leftarrow (i_{max}, \mathbf{0})$ ▷ a stack of nodes of $\hat{\mathcal{G}}$
    while $\tau \neq \emptyset$ **and** $\mathcal{D}[\mathbf{p}] \neq 1$ **do**
      $v = (i, \mathbf{q}) \leftarrow$ peek of τ
      $\tau \leftarrow \tau \setminus v$ ▷ pop the peek of the stack
      if $i = 0$ **then**
        $\mathcal{D}[\mathbf{p}] \leftarrow 1$
      else
        for all sub-cell $v_\mathbf{r} = (i \quad 1,\ 2\mathbf{q} + \mathbf{r})$ of v **do**
          if $\beta_\mathcal{S}(\mathbf{p}) \cap \boxed{v_\mathbf{r}} \neq \emptyset$ **and** $\hat{\mathcal{G}}[v_\mathbf{r}] = 1$ **then**
            $\tau \leftarrow \tau \cup v_\mathbf{r}$ ▷ push the node to the stack
          end if
        end for
      end if
    end while
  end for
  return $\mathcal{D}$

Algorithm 2 Parallel spatially varying erosion.

Require: $\mathcal{G}$ a binary grid ▷ voxelized input
Require: $\mathcal{D}$ a binary grid ▷ spatially varying dilated input
Require: $\widehat{\mathcal{D}_c}$ a pyramid constructed using Eq. 2
Require: $\mathbf{r} \in \{0,1\}^3$
  $\mathcal{E} \leftarrow \mathcal{D}$
  for all *voxel* $\mathbf{p} \in N_i^3 \mid \mathcal{E}[\mathbf{p}] \neq 0$ **and** $\mathcal{G}[\mathbf{p}] = 0$ in parallel **do**
    $\tau \leftarrow (i_{max}, \mathbf{0})$ ▷ a stack of nodes of $\widehat{\mathcal{D}_c}$
    while $\tau \neq \emptyset$ **and** $\mathcal{E}[\mathbf{p}] \neq 0$ **do**
      $v = (i, \mathbf{q}) \leftarrow$ peek of τ
      $\tau \leftarrow \tau \setminus v$ ▷ pop the peek of the stack
      if $i = 0$ **then**
        $\mathcal{E}[\mathbf{p}] \leftarrow 0$
      else
        for all sub-cell $v_\mathbf{r} = (i \quad 1,\ 2\mathbf{q} + \mathbf{r})$ of v **do**
          if $\boxed{\mathbb{V}_\mathbf{r}} \cap \boxed{\mathbf{p}} \neq \emptyset$ **and** $\widehat{\mathcal{D}_c}[v_\mathbf{r}] = (1, \cdot)$ **then**
            $\tau \leftarrow \tau \cup v_\mathbf{r}$ ▷ push the node to the stack
          end if
        end for
      end if
    end while
  end for
  return $\mathcal{E}$

we propose a parallel spatially varying erosion summarized in Alg. 2 and illustrated in Fig. 6.

As a result, $\mathcal{E}$ contains the spatially varying closing and we can finally set $\mathcal{G} := \mathcal{E}$. Note that, contrary to the dilation case, this collision algorithm between a scale augmented hierarchy and a voxel grid works only for a spherical SE field, the whole field simply being encoded in the scale-augmented hierarchy itself.

Additional Speed-Up. To improve even further the efficiency of our algorithm, we break up each spatially-varying morphology $\mathcal{D}$ and $\mathcal{E}$ into two parts. First we compute the grid at half the resolution ($\mathcal{D}_{1/2}$, $\mathcal{E}_{1/2}$), and extract a conservative contour of this result, then we run the full resolution computation solely on this sparse set of contour cells ($\mathcal{D}_{sparse}$, $\mathcal{E}_{sparse}$, see Fig. 7). As such, the burden of the volumetric complexity is reduced by a factor 8, leaving the full resolution to a much lighter surface bound complexity. Fig. 4 shows an example of our asymmetric closing compared to a symmetric closing (with a spherical SE).

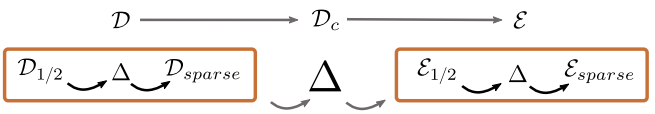


Fig. 7. **Morphological Pipeline.** $\mathcal{D}$ (resp. $\mathcal{E}$) is the full dilation (resp. erosion); $\mathcal{D}_{1/2}$ (resp. $\mathcal{E}_{1/2}$) the dilation (resp. erosion) on a half resolution grid; $\mathcal{D}_{sparse}$ (resp. $\mathcal{E}_{sparse}$) is the full resolution dilation (resp. erosion) computed on a sparse set of contour cells; Δ is an extracted contour, and $\mathcal{D}_c$ the intermediary contour necessary to compute $\mathcal{E}$ from $\mathcal{D}$.

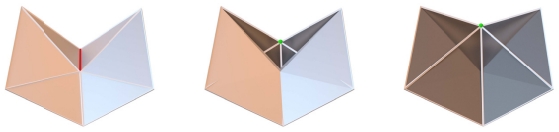


Fig. 8. **Constrained QEM.** From left to right: the neighboring triangles of an edge (in red), the optimal collapse without constraints, which violates the strict inclusion and the optimal collapse with constraints (optimized vertex in green and optimal triangle configuration in opaque material).

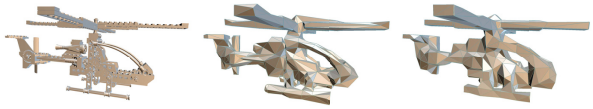


Fig. 9. **Meshing.** A Lego helicopter (120 components, left), our bounding proxy mesh with a symmetric closing (middle) and with an asymmetric closing (right), which better detects large flat and salient structures.

3.4 Meshing

The final stage of our pipeline extracts a coarse meshed proxy C from the voxel closing $\mathcal{G}$. During this operation, we aim at both reaching the (coarse) resolution enforced by $\mathcal{S}$ and maintaining the bounding condition carefully preserved at the previous steps (see Fig. 9). To do so, we start by extracting a dense mesh C which contours *exactly* $\mathcal{G}$ before performing an error-driven progressive edge collapse, combining the classical Quadric Error Metric (QEM) [Garland and Heckbert 1997] with linear constraints (see Fig. 8) guaranteeing local bounding conditions [Deng et al. 2011; Sander et al. 2000], and upper-bounding edge length w.r.t. the scale field. The error stopping criterion is set to 10^{-3} (resp. 10^{-2}) for the symmetric (resp. asymmetric) closings.

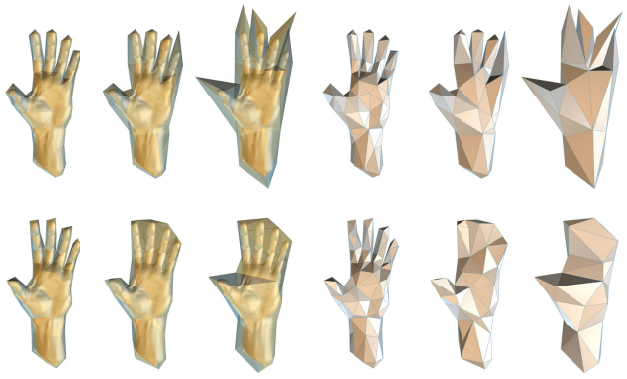


Fig. 10. **Approximation scale.** Top: Nested Cages (layered). Bottom: closing followed by Nested Cages. It requires around **4400s** and 2 nested layers to reach the coarser approximation with Nested Cages (in this case the nested layers are mandatory to get flows that properly converge), while we reach any coarse approximation in a maximum **150s** (a **x30** speed-up) by using an intermediate Closing transform.

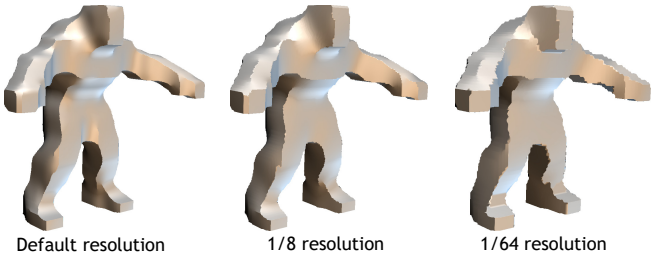


Fig. 11. **Influence of the voxelization resolution** on the closing. Even at coarse resolution, the main dominant structures are still reflected in the final morphology.

4 RESULTS AND COMPARISONS

Performances. We have implemented our method in C++, using CUDA for GPU computations[1]. We report performance measures on an Intel Core2Quad (single thread) at 2.7GHz with 8 Gb of memory and a nVidia GTX680 GPU. In Fig. 12, we report the detailed performance measures of our spatially-varying asymmetric closing operator (timings are similar for the symmetric one), for some models illustrating this paper. We observe that the system remains interactive, even at coarse scale (i.e., large structuring element), which preserves the mandatory feedback during proxy design. Compared to a naive implementation, we reach a ×70 speed-up. For instance on the Beast model we obtain 86ms and 120ms for a symmetric closing of 20 and 37 voxels respectively, while the straightforward GPU Minkowski sum based on splatting (see Sec. 2) takes 1200ms and 8800ms. Note that one of the fastest algorithm [Cuisenaire 2006] to compute a *spatially varying* closing takes around 2 minutes (on CPU due to its sequential nature) with comparable SE sizes and the same grid resolution. We also illustrate in Fig. 11 the evolution of our closing when changing the underlying initial voxel resolution. We conducted initial experiments on a more powerful GPU (nVidia GTX 980ti) also, measuring on average a 2x speed up compared to our GTX 680, which illustrates the parallel scalability of our high speed morphology. The final (offline) proxy meshing stage takes usually between 30 and 100 sec. (CPU execution). Depending on the application, this delay can be significantly reduced by using a lower initial grid resolution. Although this remains the bottleneck of the process, one should note that the proxy geometry is defined at interactive rate thanks to our high speed voxel morphology, letting this meshing stage as a final "baking" step executed only to export the bounding proxy in mesh format toward its target application.

Comparison. In order to evaluate our bounding shape approximation method, we compare three scenarios on a collection of models spanning a variety of geometric and topological characteristics. In Fig. 14, (i) the left column shows the proxy resulting from the simplification of the input using a state-of-the-art QEM mesh simplification [Garland and Heckbert 1997] followed by the Nested Cage (NC) method [Sacht et al. 2015] (in both cases, we use the publicly available implementation); (ii) in the central column, we first compute our morphological closing before simplifying the dense geometry and using *Nested Cages* to compensate for the non-bounding

[1]Our implementation will be made publicly available to support research in this area.

Model [scale;margin]	$\mathcal{D}$ *1/2*	$\mathcal{D}$ *sparse*	$\mathcal{E}$ *1/2*	$\mathcal{E}$ *sparse*	Total +contours +mipmap
mammoth [0.07;0.5]	9.0	7.8	7.16	13.8	68.8
horse [0.048;0.07]	22.2	16.9	16.4	37.8	124.3
filigree [0.25;0.5]	19.7	13.5	23.6	36.3	123.1
goro [0.079;0.5]	13.1	7.5	9.1	17.0	76.7
flower [0.043;0.5]	6.2	2.8	2.5	5.0	47.5
dino [0.046;0.5]	7.9	5.2	4.7	8.7	56.5
neptune [0.039;0.5]	7.6	6.0	6.2	10.8	61.6
chair [0.12;0.5]	13.2	10.4	10.8	18.0	52.4

Fig. 12. **Closing timings.** Performances (in milliseconds) for both cubic dilation ($\mathcal{D}$) and sperical erosion ($\mathcal{E}$) with the collision algorithm traversal on the half grid ($\mathcal{D}$ 1/2 and $\mathcal{E}$ 1/2) and the sparse set of contour cells at full resolution. Building the 3 contours (between each morphological operations) takes about 21 ms. Building the intermediate mipmap on $\mathcal{D}_c$ (the only one being computed at each modification of the scale field $\mathcal{S}$) takes about 10 ms. For each model we indicate the scale, along with the margin added to the input bounding box to compute the morphological operations (they are both expressed w.r.t. the bounding box diagonal of the input).

Model (% orig.; #f)		CQEM	QEM + NC	Closing + QEM + NC	Closing + CQEM
octopus	t	83	682	180	**50**
(0.5; 3082)	v	0.0733	**0.0724**	0.0890	0.0980
boy	t	failed	1278	502	**64**
(2.1; 778)	v		**0.1540**	0.1611	0.1778
animal	t	failed	1901	182	**38**
(0.5; 808)	v		**0.2034**	0.2100	0.2279
beast	t	failed	3496	155	**28**
(0.12; 706)	v		**0.0309**	0.0317	0.0363
filigree	t	failed	failed	420	**70**
(0.14; 2000)	v			**0.1935**	0.2287
chair	t	failed	failed	420	**60**
(0.04; 1024)	v			**0.0340**	0.0347
flower	t	failed	failed	1480	**23**
(0.04; 918)	v			**0.0126**	0.0157
hand skel.	t	failed	failed	61	**36**
(6.2; 680)	v			**0.0796**	0.0975
blade	t	failed	failed	650	**100**
(0.05; 924)	v			**0.0645**	0.0709
mammoth	t	failed	failed	failed	**80**
(0.5; 1080)	v				**0.1022**

Fig. 13. **Comparisons measures.** We reports the timings (t) in seconds for different pipelines, and the volume (v) of the final approximation.

violations caused by the simplification; (iii) in the right column, we use our morphological closing with constrained QEM [Deng et al. 2011; Sander et al. 2000] (or CQEM), as described in Sec. 3.4. For the comparison with NC (without our closing, second column in Fig. 13) we also tested the decimation stage suggested by Sacht et al. [2015] (named *regular* and *adaptive*). Although the QEM decimation usually performs best, when the flow for NC fails, it also fails for the decimations methods used in the original NC publication.

Note that our initial list of desirable properties for good proxies is matched by our appoach and has one main difference with the one proposed by Jacobson et al [2014]. We advocate that depending on the scale of the proxy w.r.t. the input mesh, the topology of the input shall not always be preserved. As can be seen in the *Mammoth*, *Hand Skeleton*, *Flower* and *Chair* models, such a behavior is often useful to reach a low budget of vertices/faces while maintaining a meaningful approximation. Indeed, in our experiments, when the model becomes too complex and the target bounding approximation too coarse, state-of-the art methods fail at producing a result. For instance, the *Nested Cages* shrink flow does not converge completely inside the decimated input. It happens when the decimation itself outputs a non reliable result, where too many intricated geometric structures and/or a complex topology cannot be handeld by the decimation.

Our method, on the contrary, regularizes this complexity and makes any further decimation much more robust. As can be seen in Fig. 14 and Fig. 13, using our *closing* before any decimation is often mandatory to build a bounding proxy using NC. This does not only hold for high genus shapes, but also for multi-components objects transformed in a single bounding proxy, such as the 56-components Robot model in Fig. 1. As objective measures of the proxy optimality (see Fig. 13), we use its volume (v) since we use the volumetric ARAP energy in NC, as well as the generation time (t). From a performance point-of-view, our method is up to 125 times faster than NC, with only a slight difference in tightness: on the Beast model for instance, we output a bounding proxy with a volume only 20% higher, while remaining visually very close.

Also, our approach can be combined with alternatives: by removing geometrical intrication, the NC algorithm computes proxies significatively faster (up to 20 times) since the flow and the reinflation benefit from our morphological regularization (less iterations for the flow, less collisions for the reinflation step). Note that the sole use of CQEM outputs meshes having self-intersections with almost all tested models (see Fig. 13).

Additional examples and comparisons. Most algorithms [Ben-Chen et al. 2009; Faraj et al. 2012; Shen et al. 2004; Xian et al. 2009] besides *Nested Cages* need an *offseting/dilation* step before the decimation stage to work properly (i.e., to fulfill the bounding property). In Fig. 15, we use the CQEM algorithm as a decimation stage, on three different geometric states: the input, a dilation and an asymmetric closing. Used directly on the input, the decimation does not produce a strictly bounding shape (226 self-intersections) and typically exhibits spiky, non-regular structures; thanks to the dilation (that also simplifies the shape), the resulting proxy is strictly bounding

(a) QEM + NC (b) Closing + QEM + NC (c) **Closing + CQEM**

Fig. 14. **Comparisons. (a)** QEM then Nested Cages. **(b)** Our closing then QEM then Nested Cages. **(c)** Our algorithm (closing then CQEM). Closeups are shown in case the Nested Cages shrink flow fails. Quantitative measures are provided in Fig. 13.

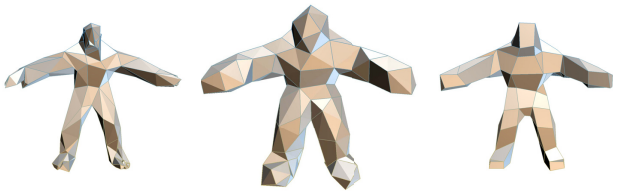


Fig. 15. **Impact of dilation/offseting**: constrained QEM collapse applied to the input mesh (left), to an offset/dilation (middle) and to our asymmetric closing (right).

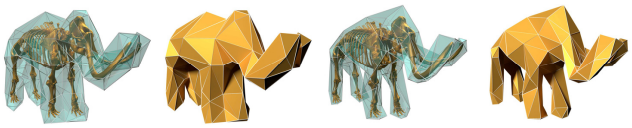


Fig. 16. **Comparison with IATV [Mandad et al. 2015].** Left: the input with the IATV mesh in overlay and the mesh alone. Right: the input with our bounding proxy in overlay and the bounding proxy alone. To ensure the strictly enclosing property while using IATV, the input needs to be dilated first, explaining the loss of tightness in the resulting mesh (its volume is 63% higher than ours). Our method performs 20 times faster.

but loosely fits the input, while our asymmetric closing produces a tight *and* cuboidal proxy, suitable for cage-based FFD for instance.

Another type of methods [Cohen et al. 1996; Mandad et al. 2015; Zelinka and Garland 2002], relying on bounded *error*/*distance* simplification, guarantees that the resulting proxy surface lies at a given *distance* (usually Haudorff based) τ of the input. Although they do not target bounding approximation, they can serve that purpose by first dilating the input using the same τ. In Fig. 16 we show a comparison with the *IATV* algorithm from Mandad et al. [2015]. Even if guarantees are provided by this combined approach, the result loosely fits the input and performs about 20 times slower than ours.

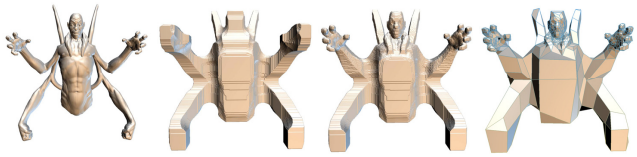
The method from Xian et al. [2012] produces effective cubic-like structures, well-suited for user interaction. However our proxies (with an asymmetric closing) are much tighter and significantly faster to compute. On the same Octopus model, the bounding proxy is computed in 20 minutes with the method of Xian et al., compared to about 30 seconds with our approach. Moreover, based on a very coarse oriented bounding box shape decomposition (to target cage-based FFD), this method cannot produce moderately coarse, tight approximations, contrary to ours (see Fig. 14 for instance).

5 APPLICATIONS

5.1 Interactive Bounding Proxy Design

Our system provides the user with a simple and intuitive means to tailor the proxy geometry through a single component: the scale of the intended bounding object. At first, the user can control a base scale globally, leading to a uniform proxy resolution. Then, as some regions of the object may require finer proxy resolution than others, the user can adjust this scale locally, by interactively *brushing* the desired scale directly on the object (see Fig. 17 and 18). This brush tool comes with three modes, that either increase the base scale, decrease it or reset it to the default value (i.e., classical *eraser* metaphor). Consequently, at any time, this interactive environment provides our previously described approximation machinery with two objects: the input mesh $\mathcal{M}$ and a user defined scale field $\mathcal{S}$ (see Appendix A for the precise definition of the scale field in this case). Since both voxelization and closing are designed to run in real time, the user benefits from an instantaneous feedback on the proxy geometry through the adaptive closing.

As voxel aliasing may disturb this visual feedback during editing, we additionally compute a piecewise smooth normal field by estimating a per-boundary-voxel normal vector and performing a bilateral filtering on them, with domain support adjusted to ignore voxel-wide wavering. As a result, the visual feedback exhibits large dominant feature lines while eliminating the individual voxel edges in the rendering (see Fig. 18). In Fig 17, we illustrate a proxy design sequence using our proxy brush. The user starts by setting a global (background) scale for the bounding proxy and then paints interactively on the model to refine the proxy structure around the head and the hands of the model. The interactive visual feedback provided by the closing geometry is critical here, as the user progressively refines it until reaching the desired adaptive level-of-detail. Moreover, the coarse proxy which is ultimately extracted has a resolution that



(a) Input (b) Closing (c) Closing (d) Proxy

Fig. 17. **Interactive bounding proxy design:** starting from the 4-components Goro model (a), the scale of the bounding proxy is first globally set by the user (b), before being interactively refined in regions of interest using the proxy brush (c) and finally meshed (d).

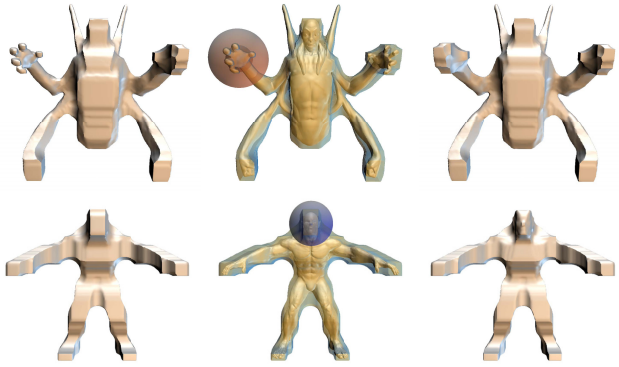


Fig. 18. **Interactive bounding proxy design:** bounding proxy geometry before (left), during (middle) and after (right) *max* brush editing on the Goro model (top) and *min* brush editing on the beast model (bottom) with an asymmetric closing. Here, the piecewise-smooth normal field for aliasing-free shading is activated for a better visual feedback.

can be intuitively guessed from the closing shape and scale, with long edges in coarse scale regions and more degrees of freedom in fine scale areas. An additional feature of our system resides on the fact that the scale field is painted on the surface of $\mathcal{M}$, which allows to accurately specify the location that requires local scale adjustment directly on the input geometry.

5.2 Freeform Space Deformation

Space deformation using cage coordinates is one family of freeform deformation (FFD) methods and has recieved an increasing interest over the last decade, with today's cage coordinate systems providing high quality deformation with good geometric properties such as smoothness, boundary interpolation or local quasi-conformality [Joshi et al. 2007; Ju et al. 2005; Lipman et al. 2008]. By expressing each vertex of the target shape in the coordinate system stemming from a closed bounding mesh (called "cage" in this context), users are able to deform smoothly complex meshes using the small set of cage vertices. However, it still induces a tedious manual construction of the underlying cage geometry and topology for which the design interface described in Sec. 5.1 is particularly well-suited. In particular, the asymmetric nature of our morphological closing gives rise to bounding proxies with box-like features, mimicking hand-crafted cages, and favors large flat areas modeled by fewer polygons, making them good candidates for acting as FFD cages.

In the context of FFD, the cage shall not obscure the underlying mesh, and be exposed in a simple and light way to the user; in particular, with a cuboidal structure, a quite natural quad-dominant mesh can be extracted on our proxies on-the-fly [Tarini et al. 2010], the resulting visual tri-quad mesh structure being exposed to the user for FFD interaction (see Fig. 15; more results are provided in the supplemental materials).

Last, although an axis-aligned structuring element does not produce axis-aligned cuboidal features only, our system supports also an additional volumetric rotation field, used to orient the cubic SE in order to roughly follow the main structures of the input, resulting in more feature-preserving and tighter bounding proxies. This rotation field, represented as quaternions, is computed by solving a grid-based biharmonic equation with a sparse set of constraints (no more than 3 in the examples of Fig. 19) provided by the user. More evolved rotation fields may also be injected instead.

5.3 Physical Simulation

Our proxies can be used as economic substitutes to high resolution meshes for physical simulation. Their bounding property can be leveraged to use cage coordinates systems for soft body simulation (see Fig. 20), or to ensure proper results in cloth simulations with no pass-through artifacts when rendering the high resolution meshes after simulation (see Fig. 21). Thanks to the reduced number of elements of our proxies, solving for the elasticity/plasticity equations and the collisions/self-collisions is at least *x100* faster than with the full resolution models, while a *18x* speed-up is achieved on the cloth simulation. For both setups, the result remains visually plausible despite the approximation (see our supplemental video). We used *Blender* as a physical simulator along with its internal Harmonic Coordinates [Joshi et al. 2007] deformation transfert system.

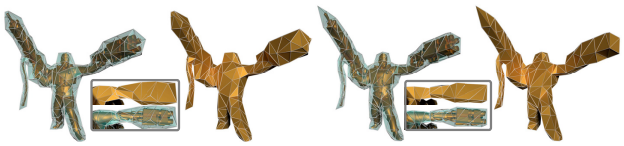


Fig. 19. **Rotation Field.** Left: a bounding proxy computed with a simple axis-aligned closing. Right: with a non-axis aligned asymmetric closing (using a rotation field). The resulting proxy is tighter and better respect the features of the input shape.

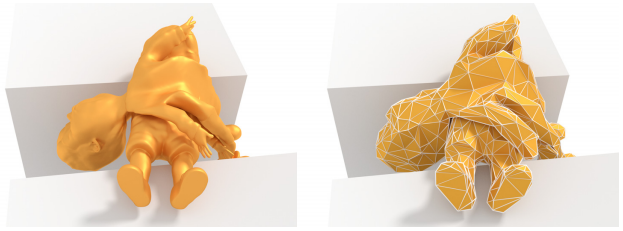


Fig. 20. **Soft body simulation** with our proxies, using cage-based motion transfer. In this experiment, the speed-up was at least of ***x100*** thanks to the lighter proxies.

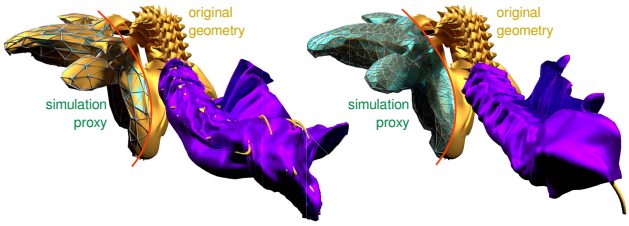
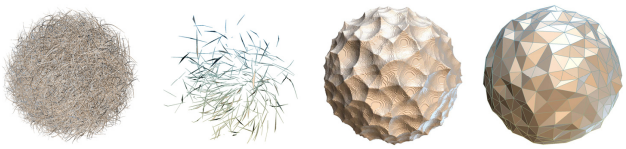


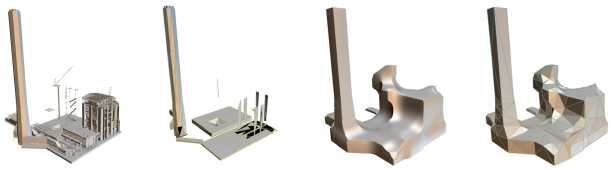
Fig. 21. **Cloth simulation.** Left: simulation with a simple QEM decimation proxy. Right: with our bounding proxy. Our proxy does not exhibit any pass-through artifact of the sheet during the simulation. For this experiment, we achieve a ***x18*** speed-up thanks to the lighter proxies. Note that while both proxies have the same face count (6656), ours has a much better distributed complexity thanks to the regularization of the closing (note the oversized elements on the petals of the QEM proxy).

5.4 LoDs Generation for Many-Components Meshes

Our morphological closing decouples the geometry and topology of a shape from its mesh sampling, regularizing the shape for the upcoming decimation process. Although critical for constrained bounding approximations, this particular behavior can indeed also be leveraged for traditional, *non-bounding* shape approximations, as we can see in Fig. 22 where our morphological approach leads to a better coarse scale model than a direct simplification. For 3D objects made of multiple small components, our strategy allows meshing even very coarse level-of-details considering the shape "as a whole" instead of individual elements. To that respect, the *hairy ball* example (in Fig. 22, top row) illustrates how traditional surface simplification algorithms, often topology-preserving, fail at generating a faithful coarse LoD from the input, on the contrary



(a) Hairball: 2.8M faces, 5k components, simplified to 1k faces.



(b) Powerplant: 12M faces, 176k components, simplified to 478 faces.

Fig. 22. **Coarse LoD extraction from complex multi-components meshes:** even under no bounding condition, our morphological approach can be instrumental. From left to right: input, direct QEM simplification, our intermediate closing and our final proxies (QEM simplification). The top (reps. bottom) row uses a symmetric (resp. asymmetric) closing.

to our method which easily recovers the underlying simple shape emaneting from the high resolution mesh. We can also see, with the *powerplant* model (Fig. 22, bottom row), that our asymmetric closing operator retains and enforces the inherent large dominant flat and salient structures.

6 CONCLUDING REMARKS

We have introduced a new shape approximation algorithm that generates efficiently coarse bounding proxies from complex input meshes. Our approach is based on a morphological closing which acts as an intermediate representation providing, by construction, all the good properties the bounding proxy geometry and topology shall exhibit. We also designed a high performance GPU closing algorithm that runs at interactive framerate, supports spatially-varying structuring elements, both in size and orientation, and asymmetry. We encapsulated it within efficient voxelization and meshing stages to generate tight bounding proxies at any scale automatically. Moreover, we provide an (optional) intuitive and adaptive interactive control mechanism for the proxy scale, topology and resolution by the mean of a *brushed* scale field together with a user-defined rotation field. We demonstrated, on a number of examples, that not only our bounding shape approximation behaves better and more robustly than alternatives for complex shapes, but also that it can be combined with state-of-the-art decimation and caging algorithms to achieve even better results. Last, we illustrated potential applications of our bounding proxies with interactive proxy design, soft body simulation, cloth simulation, FFD and many-components LoD generation.

Currently, our framework supports spatially varying size, shape and orientation for the dilating structuring element. However, carving intricate regions very close to the objects (e.g., between the legs of a character) remains a tedious task for which the interface of our system could give explicit control on the anisotropy of the dilating cube while brushing. The scale field itself could be procedurally generated using a preliminary, application-dependent, shape analysis to dictate where to refine the resolution of the proxy, exploiting different measures such as visibility or perceptual saliency. As suggested by Jacobson et al. [2014], detecting and modelling explicitly symmetries on the input mesh to reflect them in the proxy, in particular when used as a FFD cage, could also be useful to some applications (e.g., editing).

Although our intermediate closing geometry guarantees the bounding condition, the progressive decimation of the meshing step ensures this property only locally and cannot bring global guarantees on self-intersections for instance. In our experiments, we never encountered any problem in practice, as the morphological closing regularizes the geometry of $\mathcal{G}$ to a safe state that strongly discourages such defects. We show in Fig. 23 a model for which we set a smaller scale at a location where the geometry would have been fused otherwise. In this setup, our scale-adaptive decimation prevents long edges – the ones that most likely would have produced self-intersections – to appear at this location. Nevertheless a formal guarantee for the full pipeline would be interesting to develop in future work. In practice though, in case our method would fail at producing a self-intersection-free result, one can always use the *meshfix* [Attene 2010] algorithm followed by a NC pass.

While we focused our work on surface mesh input, there is little to no assumption made once the voxelization is performed. Therefore, one can envision much diverse input representations to our framework, coupled with specific methods to guarantee a proper voxelization, including point clouds equipped with an implicit moving least square operator or largely damaged meshes processed with robust hole filling methods [Sacht et al. 2013].

Last, our approximation algorithm supports an optional orientation field, which is defined rather naively at the moment and for which an effective design environment would be valuable.

ACKNOWLEDGMENTS

We thank Manish Mandad for providing IATV meshing results and Leonardo Sacht, Etienne Vouga and Alec Jacobson for thier code. This work is partially supported by the French National Research Agengy (ANR) under grant ANR 16-LCV2-0009-01 ALLEGORI and by BPI France under grant PAPAYA.

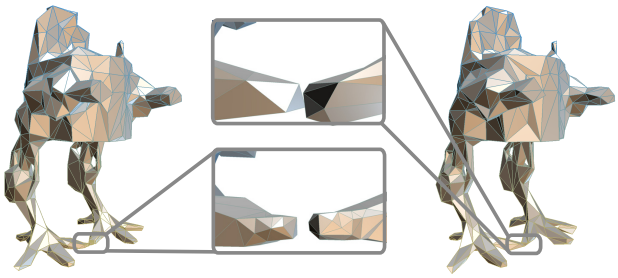


Fig. 23. **Pinch configuration.** Left: our scale-adaptive meshing matches the prescribed scale at the pinch, producing a locally more tessellated result and avoiding non-local self-intersection. Right: when desactivating scale-adaptive meshing, the large edges at both side of the pinch self-intersect.

A BRUSH-BASED SCALE FIELD DEFINITION

We model the positive 3D scale field $\mathcal{S} : \mathfrak{R}^3 \rightarrow \mathfrak{R}^+$ that tailors the bounding proxy scale and resolution as a 3D scalar grid, uniformly initialized to a base scale s_{base} by the user. During interactive editing, when the brush is applied at position $\mathbf{p}$ on $\mathcal{M}$, $\mathcal{S}$ is altered according to a support σ, defining the size of its influence region, and a target scale s as follow:

$$\mathcal{S}[\mathbf{x}] := \begin{cases} \max(\phi(\mathbf{x}, \mathcal{S}[\mathbf{x}], s), \mathcal{S}[\mathbf{x}]) & \text{in increase mode} \\ \min(\phi(\mathbf{x}, \mathcal{S}[\mathbf{x}], s), \mathcal{S}[\mathbf{x}]) & \text{in decrease mode} \\ \phi(\mathbf{x}, \mathcal{S}[\mathbf{x}], s_{base}) & \text{in erase mode} \end{cases}$$

with the modification smoothness being ensured by an interpolation kernel ϕ that regularizes the desired scale around $\mathbf{p}$ w.r.t. its previous value:

$$\phi(\mathbf{x}, s_{prev}, s) = (1 - \gamma(\|\mathbf{x} - \mathbf{p}\|))s_{prev} + \gamma(\|\mathbf{x} - \mathbf{p}\|)s$$

with

$$\gamma(t) = \begin{cases} 1 & \text{if } t \in [0, s[\\ (1 - (\frac{t}{\sigma})^2)^2) & \text{if } t \in [s, s + \sigma[\\ 0 & \text{otherwise} \end{cases}$$

REFERENCES

Marco Attene. 2010. A Lightweight Approach to Repairing Digitized Polygon Meshes. *Vis. Comput.* 26, 11 (2010), 1393–1406.

Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. 2009. Spatial Deformation Transfer. In *Proc. SCA*. 67–74.

N. Bouaynaya, M. Charif-Chefchaouni, and D. Schonfeld. 2008. Theoretical Foundations of Spatially-Variant Mathematical Morphology Part I: Binary Images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30, 5 (2008), 823–836.

V. Boulos, V. Fristot, D. Houzet, L. Salvo, and P. Lhuissier. 2012. Investigating performance variations of an optimized GPU-ported granulometry algorithm. In *Design and Architectures for Signal and Image Processing (DASIP)*. 1–6.

Stéphane Calderon and Tamy Boubekeur. 2014. Point Morphology. *ACM Trans. Graph.* 33, 4 (2014), 45:1–45:13.

Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. 1996. Simplification Envelopes. In *Proc. SIGGRAPH*. 119–128.

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational Shape Approximation. In *Proc. SIGGRAPH*. 905–914.

Olivier Cuisenaire. 2006. Locally adaptable mathematical morphology using distance transformations. *Pattern Recognition* 39, 3 (2006), 405 – 416.

Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. 2011. Automatic Cage Building with Quadric Error Metrics. *Journal of Computer Science and Technology* 26, 3 (2011), 538–547.

H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. 1983. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory* 29, 4 (1983), 551–559.

Jihad El-Sana and Amitabh Varshney. 1998. Topology Simplification for Polygonal Virtual Environments. *IEEE Trans. Vis. Comput. Graph.* 4 (1998), 133–144.

Noura Faraj, Jean-Marc Thiery, and Tamy Boubekeur. 2012. VoxMorph: 3-scale Freeform Deformation of Large Voxel Grids. *Comput. & Graph.* 36, 5 (aug 2012), 562–568.

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proc. SIGGRAPH*. 209–216.

J. Gil and M. Werman. 1993. Computing 2-D min, median, and max filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 15, 5 (1993), 504–507.

Joseph (Yossi) Gil and Ron Kimmel. 2002. Efficient Dilation, Erosion, Opening, and Closing Algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 12 (2002), 1606–1617.

H. Hedberg, P. Dokladal, and V. Öwall. 2009. Binary Morphology With Spatially Variant Structuring Elements: Algorithm and Architecture. *Image Processing, IEEE Transactions on* 18, 3 (2009), 562–572.

Matthias Hopf and Thomas Ertl. 2000. Accelerating Morphological Analysis with Graphics Hardware. In *Proc. Workshop on Vision, Modelling, and Visualization VMV fi00*. 337–345.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1993. Mesh optimization. In *Proc. SIGGRAPH*. 19–26.

Paul T. Jackway and Mohamed Deriche. 1996. Scale-Space Properties of the Multiscale Morphological Dilation-Erosion. *IEEE Trans. Pattern Anal. Mach. Intell.* 18, 1 (1996), 38–51.

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. 2014. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH Courses*.

G Jagannathan, R Subash, and K Senthil Raja. 2014. A Novel Open Source Morphology Using GPU Processing With LTU-CUDA. *International Journal for Advance Research in Engineering and Technology* 2, 1 (2014).

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic Coordinates for Character Articulation. *ACM Trans. Graph.* 26, 3, Article 71 (2007).

T. Ju, S. Schaefer, and J. Warren. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.* 24, 3 (2005), 561–566.

B. Le and Z. Deng. 2017. Interactive Cage Generation for Mesh Deformation. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (SI3D)*. 3:1–3:9.

Hélène Legrand and Tamy Boubekeur. 2015. Morton Integrals for High Speed Geometry Simplification. In *Proc. ACM SIGGRAPH/EUROGRAPHICS High-Performance Graphics*. 105–112.

Peter Lindstrom. 2000. Out-of-core simplification of large polygonal models. In *Proc. SIGGRAPH*. 259–262.

Y. Lipman, D. Levin, and D. Cohen-Or. 2008. Green coordinates. *ACM Trans. Graph.* 27, 3 (2008), 1–10.

Manish Mandad, David Cohen-Steiner, and Pierre Alliez. 2015. Isotopic Approximation Within a Tolerance Volume. *ACM Trans. Graph.* 34, 4 (2015), 64:1–64:12.

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J. Mitra. 2009. Abstraction of Man-made Shapes. *ACM Trans. Graph.* 28, 5 (2009), 137:1–137:10.

Ken Museth. 2013. VDB: High-resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3 (2013), 27:1–27:22.

Laurent Najman and Hugues Talbot (Eds.). 2010. *Mathematical Morphology: From Theory to Applications*. Wiley.

Jarek Rossignac and Paul Borrel. 1993. Multi-resolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics*. 455–465.

Christian Rössl, Leif Kobbelt, and Hans-Peter Seidel. 2000. Extraction of feature lines on triangulated surfaces using morphological operators. In *AAAI Symp. Smart Graphics*, Vol. 00-04. 71–75.

Leonardo Sacht, Alec Jacobson, Daniele Panozzo, Christian Schüller, and Olga Sorkine-Hornung. 2013. Consistent Volumetric Discretizations Inside Self-Intersecting Surfaces. *Computer Graphics Forum* 32, 5 (2013), 147–156.

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested Cages. *ACM Trans. Graph.* 34, 6, Article 170 (2015), 170:1–170:14 pages.

Pedro V. Sander, Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. 2000. Silhouette Clipping. In *Proc. SIGGRAPH*. 327–334.

Michael Schwarz and Hans-Peter Seidel. 2010. Fast Parallel Surface and Solid Voxelization on GPUs. *ACM Trans. Graph.* 29, 6 (2010), 179:1–179:10.

Jean Serra. 1983. *Image Analysis and Mathematical Morphology*. Academic Press, Inc.

Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. 2004. Interpolating and Approximating Implicit Surfaces from Polygon Soup. In *Proc. SIGGRAPH*. 896–904.

A. Söderström and K. Museth. 2010. A Spatially Adaptive Morphological Filter for Dual-Resolution Interface Tracking of Fluids. In *Eurographics*. 5–8.

Pierre Soille, Edmond J. Breen, and Ronald Jones. 1996. Recursive Implementation of Erosions and Dilations Along Discrete Lines at Arbitrary Angles. *IEEE Trans. Pattern Anal. Mach. Intell.* 18, 5 (1996), 562–567.

Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, and Enrico Puppo. 2010. Practical quad mesh simplification. *Computer Graphics Forum* 29, 2 (2010), 407–418.

Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-Meshes: Shape Approximation Using Spherical Quadric Error Metrics. *ACM Trans. Graph.* 32, 6, Article 178 (2013), 178:1–178:12 pages.

Matthew Thurley and Victor Danell. 2012. Fast morphological image processing open-source extensions for GPU processing with CUDA. *IEEE Journal on Selected Topics in Signal Processing* 6, 7 (2012), 849–855.

Marcel van Herk. 1992. A Fast Algorithm for Local Minimum and Maximum Filters on Rectangular and Octagonal Kernels. *Pattern Recogn. Lett.* 13, 7 (1992), 517–521.

Chuhua Xian, Hongwei Lin, and Shuming Gao. 2009. Automatic generation of coarse bounding cages from dense meshes. In *IEEE SMI*. 21–27.

Chuhua Xian, Hongwei Lin, and Shuming Gao. 2012. Automatic cage generation by improved OBBs for mesh deformation. *The Visual Computer* 28, 1 (2012), 21–33.

Steve Zelinka and Michael Garland. 2002. Permission Grids: Practical, Error-bounded Simplification. *ACM Trans. Graph.* 21, 2 (2002), 207–229.