Sphere-Meshes: Shape Approximation using Spherical Quadric Error Metrics

Jean-Marc Thiery Émilie Guy Tamy Boubekeur

Telecom ParisTech – CNRS LTCI – Institut Mines-Telecom

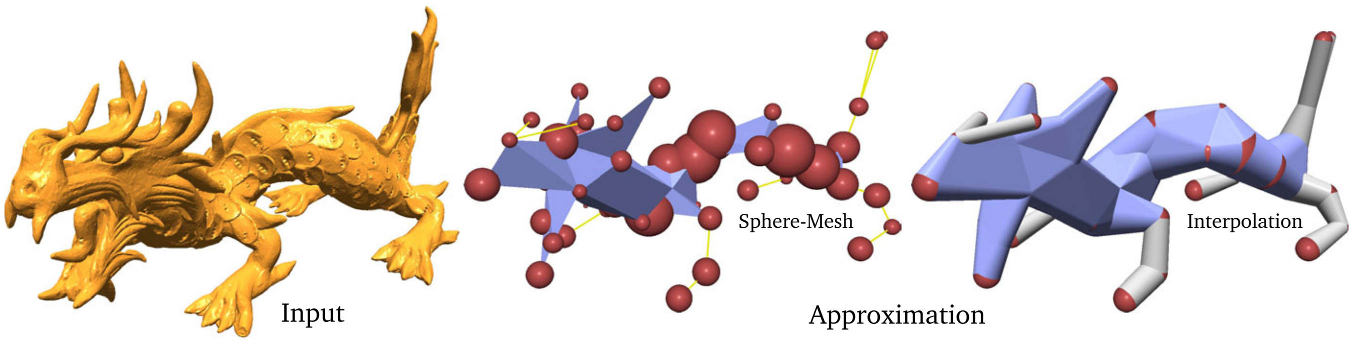


Figure 1: *Starting from a surface mesh (left, 200k triangles), we propose an approximation algorithm which generates a sphere-mesh (middle) defining an extremely simplified model (here with 50 spheres) as a sphere interpolation over a set of edges and polygons (right). The input mesh is shown in orange (left), the sphere-mesh is displayed in the middle (spheres in red, edges in yellow, triangles in blue) and the interpolated sphere-mesh geometry is shown on the right (edge interpolation in grey, triangle interpolation in blue).*

Abstract

Shape approximation algorithms aim at computing simple geometric descriptions of dense surface meshes. Many such algorithms are based on mesh decimation techniques, generating coarse triangulations while optimizing for a particular metric which models the distance to the original shape. This approximation scheme is very efficient when enough polygons are allowed for the simplified model. However, as coarser approximations are reached, the intrinsic piecewise linear point interpolation which defines the decimated geometry fails at capturing even simple structures. We claim that when reaching such extreme simplification levels, highly instrumental in shape analysis, the approximating representation should explicitly and progressively model the volumetric extent of the original shape. In this paper, we propose *Sphere-Meshes*, a new shape representation designed for extreme approximations and substituting a *sphere* interpolation for the classic point interpolation of surface meshes. From a technical point-of-view, we propose a new shape approximation algorithm, generating a sphere-mesh at a prescribed level of detail from a classical polygon mesh. We also introduce a new metric to guide this approximation, the *Spherical Quadric Error Metric* in $\mathbb{R}^4$, whose minimizer finds the sphere that best approximates a set of tangent planes in the input and which is sensitive to surface orientation, thus distinguishing naturally between the *inside* and the *outside* of an object. We evaluate the performance of our algorithm on a collection of models covering a wide range of topological and geometric structures and compare it against alternate methods. Lastly, we propose an application to deformation control where a sphere-mesh hierarchy is used as a convenient rig for altering the input shape interactively.

CR Categories: Computing Methodologies [Computer Graphics]: Shape Modeling—Mesh Models; Computing Methodologies [Computer Graphics]: Shape Modeling—Shape Analysis.

Keywords: shape approximation, simplification

Links: DL PDF

1 Introduction

Approximating 3D shapes using a minimal set of geometric primitives offers a wide range of applications, from shape analysis to interactive modeling. Starting from a dense surface mesh, simplification methods are a popular class of algorithms to generate such approximations. They can essentially be classified in three categories: (i) *clustering* methods [Rossignac and Borrel 1993][Lindstrom 2000][Schaefer and Warren 2003][Cohen-Steiner et al. 2004], which decompose the original surface into a collection of regions and substitute each region with a single representative (e.g., point or face), (ii) *decimation* methods [Hoppe et al. 1993][Garland and Heckbert 1997], which iteratively remove surface samples and relocate their neighbors to optimize for the original shape and (iii) *resampling* methods [Turk 1992; Alliez et al. 2003; Yan et al. 2009] which compute a new, potentially coarser, point distribution on the surface and establish a new connectivity.

Alternatively, one can also consider the volume bounded by the surface and provide a simplification by means of the Medial Axis Transform (MAT) [Blum 1967] for instance, leading to a representation with simplified volumetric structures [Amenta et al. 2001; Dey and Zhao 2004; Chazal and Lieutier 2005; Sud et al. 2007; Miklos et al. 2010].

In this paper, we introduce *sphere-meshes*, an approximation model inspired from both worlds, which are a connected set of *spheres* that are linearly interpolated along simplices (i. e., edges or triangles, see Fig. 1). In particular, we propose an algorithm which computes such an approximation at any desired level of detail from a polygon mesh. Indeed, a classical polygon mesh is a special case of a sphere-mesh, where vertices are spheres with zero radius. Coarsening the approximation, sphere-meshes progressively evolve from a surface to a volumetric object, using the radius of the spheres to model the *thickness* of the shape (see Fig. 2).

In this automatic approximation process, we optimize for the spheres by introducing the *spherical quadric error metric* (Sec. 2) to optimally fit a sphere to a subset of the tangent planes of the input surface (i. e., vertex or triangle tangent space). Our metric accounts for the normal orientation and distinguishes naturally between the inside and the outside of the 3D shape. We use this new error metric in a bottom-up approximation algorithm (Sec. 3) to progressively compute level-of-details of the input with sphere-meshes, with optional local approximation control. As a result, we show that at extreme simplification levels, a sphere-mesh succeeds at faithfully representing the input shape while classical polygon approximations fail quickly (see Sec. 4). Finally, we propose an application of our shape approximation model by using it as an automatic intermediate high-level control structure for interactive freeform deformation (Sec. 5). Although our representation is compatible with triangle surface meshes, tetrahedral meshes and medial axis, we focus on the surface case.

1.1 Related work

Mesh simplification One core component of our approach, inspired from classical mesh decimation methods [Garland and Heckbert 1997], is a *quadric error metric* defined to measure and optimize the difference between the original shape and its simplification. Similarly, our representation is designed for shape approximation, to capture the shape of an object with very few elements. This focus is prominent in simplification methods, which aim at finding a small amount of good representative polygons from a dense set. To do so, one can either use ordered edge collapse operations [Hoppe et al. 1993; Garland and Heckbert 1997] up to a prescribed resolution, cluster spatially the input surface elements before triangulating the cluster "averages" [Rossignac and Borrel 1993; Lindstrom 2000] or use a variational framework to segment the shape and fit simple primitives to each region, such as planes [Cohen-Steiner et al. 2004], spheres [Wang et al. 2006], cylinders/cones [Wu and Kobbelt 2005] and quadrics [Yan et al. 2006].

We argue that *extreme* mesh simplification requires defining volumetric elements instead of surfaces, while providing a simple topological structure between them.

Shapes from volumetric primitives Approximating shapes with a set of simple geometric primitives can be performed in numerous ways in digital shape modeling, with applications including for example shape recognition, multi-resolution visualization or collision detection. For instance, the medial axis transform [Blum 1967] (MAT) of a 3D surface mesh is the set of all 3D points having more than one closest point on its boundary and takes the form of a topological skeleton made of edges and faces together with a radius function. Alternatively, constructive solid geometry (CSG) methods model the shape of an object as a tree carrying simple geometric primitives on its leaves and boolean operations on its internal nodes. Extremely efficient at representing certain classes of manufactured objects, these models are usually defined from scratch and do not cope easily with automatic shape approximation. Beyond MAT and CSG, the representation of volumes as the union of primitives has also been recently studied for spheres [Wang et al. 2006], ellipsoids [Lu et al. 2007], as well as axis-aligned or oriented bounding boxes [Lu et al. 2007]. Bradshaw et al. proposed to compute a sphere tree from an approximation of the medial axis [Bradshaw and O'Sullivan 2002; Bradshaw and O'Sullivan 2004; Stolpner et al. 2012]. In a different context, spatial hierarchies have been extensively used in real time multi-resolution visualization [Rusinkiewicz and Levoy 2000] and physics [James and Pai 2004]. Often based on simple bounding primitives (e.g., spheres or boxes) organized in a spatial binary tree, such structures aim at quickly culling empty space but only provide poor quality shape approximation at their coarser levels.

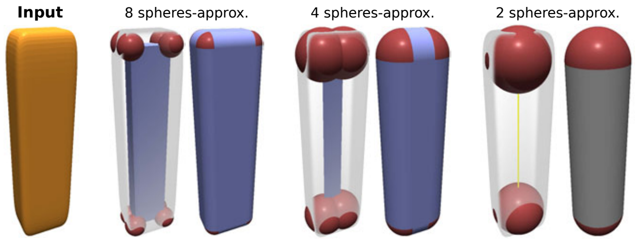


Figure 2: ***Transition from surface to volumetric representation:*** *For each approximation, we show the sphere-mesh structure with semi-transparent input and the interpolated sphere-mesh geometry.* ***8-spheres approx.:*** *the base structure, made of triangles, is homeomorphic to the input surface.* ***4-spheres approx.:*** *the base structure, made of a single sheet of triangles, is no longer homeomorphic to the input.* ***2-spheres approx.:*** *two spheres linked by a single edge.*

In contrast, we aim at representing the input 3D object with few fitting primitives while we also consider their *interpolation* over the simplices of a mesh sub-structure.

Sphere Skeletons Shape representations based on sphere interpolations have recently gained interest in shape modeling with the ZBrush tool [Pixologic 2001], popular in the SFX industry. With this tool, a skeleton of spheres – called *ZSpheres* – is manually constructed to define a shape at coarse grain, before refining its surface with displacements. With *B-Meshes*, Ji et al. [2010] improved this class of representations, in particular with a better mesh extraction.

In contrast to these methods, we propose to approximate *automatically* an existing, possibly dense mesh (e.g., scanned geometry), with an interpolation of spheres. Additionally, our sphere-mesh representation extends the interpolation beyond skeletons, using polygons as well to model non-tubular regions.

1.2 Overview

In this paper, we make the following contributions:

1. the *sphere-mesh* representation composed of a sphere set with additional connectivity information – each simplex corresponding to the linear interpolation of the four dimensional points $(q_i; r_i)$, with $q_i = (x_i, y_i, z_i)$ the sphere centers and r_i their radii; this representation extends existing skeleton-based sphere interpolations;
2. the *spherical quadric error metric* (SQEM) guiding the sphere-mesh approximation and whose minimizer is a sphere fitting a set of planes in the least squares sense;
3. a shape approximation algorithm which computes a sphere-mesh efficiently, from an input triangle mesh, with optional local approximation control through an importance map;
4. an interactive freeform deformation framework using sphere-meshes as automatic multi-resolution control structures.

2 Sphere-mesh representation

We aim at approximating 3D shapes as a base mesh composed of edges E and triangles T, indexing a sphere set $S = \{S(q_i, r_i)\}_i$ with centers $q_i \in \mathbb{R}^3$ and radii r_i, this "thickness" r_i being linearly interpolated along E and T. Intuitively, a segment $[S_i, S_j]$ models the union of the interpolated spheres between S_i and S_j ($[S_i, S_j] = \cup_{u\in[0,1]}\{S(uq_i + (1 \quad u)q_j; ur_i + (1 \quad u)r_j)\}$), and corresponds to the convex hull of $S_i \cup S_j$. The same property holds for higher dimensional primitives such as triangles and tetrahedra. From a morphological point of view, a *sphere-mesh* $\{S, E, T\}$ corresponds to a *Minkowski sum* of the polygon mesh defined by the sphere centers with a sphere having a spatially-varying radius. One can also see sphere-meshes as a parametric counterpart to *kernel implicit surfaces*.

Notations In the following, $\mathcal{M}_{mn}$ denotes the set of real $m \times n$-matrices, $\mathcal{S}^n$ the set of symmetric matrices of $\mathcal{M}_{nn}$, and M^{ij}_{kl} the $(k \quad i + 1) \times (l \quad j + 1)$-submatrix of M, whose top left corner element is M_{ij} and the bottom right element is M_{kl}. $a \times b$ denotes the cross product between two 3D vectors a and b, and $a^t \cdot b$ their dot product. $\{p, n\}^\perp$ denotes the plane that is orthogonal to n and intersects p: $\{p, n\}^\perp \equiv \{x \in \mathbb{R}^3 | n^t \cdot (p \quad x) = 0\}$.

2.1 Spherical quadric error metrics

We start by introducing the underlying geometric metric in our approach, the *spherical quadric error metric* (SQEM), which is based on the signed distance $d(S(q, r), \{p, n\}^\perp)$ from the sphere $S(q, r)$ to the (oriented) plane $\{p, n\}^\perp$ (see Fig. 3 (*i*)):

$$d(S(q, r), \{p, n\}^\perp) = n^t \cdot (p \quad q) \quad r \qquad (1)$$

This distance differs from the classical distance from an *unoriented* point p (as opposed to a plane $\{p, n\}^\perp$) to a sphere (i. e., $|p \quad q| \quad r$). It also takes into account the orientation of the normals, and distinguishes naturally between convex and concave regions (see Fig. 3, (*ii*) and (*iii*)). Note, that the squared value of the distance from a point to a sphere (i. e., $(|p \quad q| \quad r)^2$) cannot be expressed as a quadric w.r.t. q and r.

We associate the set of spheres with center q and radius r to vectors in $\mathbb{R}^4$ by writing $\mathbf{s} := (q; r) \simeq S(q, r)$, and we write $\bar{p} = (p; 0), \bar{n} = (n; 1) \in \mathbb{R}^4$ in the following. Using this notation, $d(\mathbf{s}, \{p, n\}^\perp) \equiv d(S(q, r), \{p, n\}^\perp) = \bar{n}^t \cdot (\bar{p} \quad \mathbf{s})$.

The spherical quadric error metric $SQEM_{p,n}(\mathbf{s})$ that represents the squared distance from the plane $\{p, n\}^\perp$ to a variable sphere $\mathbf{s}$ is a quadric w.r.t. $\mathbf{s}$, and $SQEM_{p,n}(\mathbf{s}) = d(\mathbf{s}, \{p, n\}^\perp)^2 = (\bar{n}^t \cdot \bar{p} \quad \bar{n}^t \cdot \mathbf{s})^2 = (\bar{n}^t \cdot \bar{p})^2 + (\bar{n}^t \cdot \mathbf{s})^2 \quad 2(\bar{n}^t \cdot \bar{p})\bar{n}^t \cdot \mathbf{s}$, which

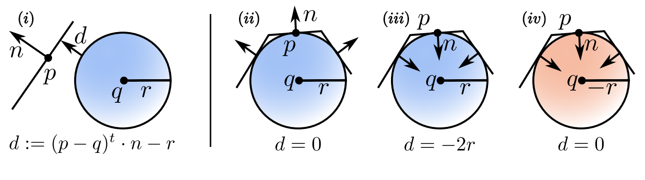


Figure 3: ***Signed distance from a sphere to a plane*** *(i), which takes the orientation of the normals into account. It favours the fitting of convex surfaces (ii) while penalizing the fitting of concave surfaces (iii). During the optimization, we forbid spheres with* ***negative*** *radius (here $\quad |(p \quad q)^t \cdot n|$) fitting concave regions (iv).*

boils down to the following:

$$SQEM_{p,n}(\mathbf{s}) = \mathbf{Q}(\mathbf{s}) = \frac{1}{2}\mathbf{s}^t \cdot A \cdot \mathbf{s} \quad b^t \cdot \mathbf{s} + c \qquad (2)$$

with $A = 2\left[\begin{array}{c|c} n \cdot n^t & n \\ \hline n^t & 1 \end{array}\right] \in \mathcal{S}^4$, $b = 2(n^t \cdot p)\left[\begin{array}{c} n \\ \hline 1 \end{array}\right] \in \mathbb{R}^4$ and $c = (n^t \cdot p)^2 \in \mathbb{R}$.

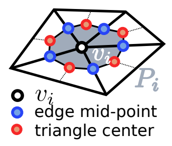
In the following, we refer to such a quadric $\mathbf{Q}$ in $\mathbb{R}^4$ by writing its components $\mathbf{Q} \equiv (A, b, c) \in \mathcal{S}^4 \times \mathbb{R}^4 \times \mathbb{R}$ explicitly. The sum of two quadrics $\mathbf{Q}_1 \equiv (A_1, b_1, c_1)$ and $\mathbf{Q}_2 \equiv (A_2, b_2, c_2)$ is computed by summing up their different components: $\mathbf{Q}_1 + \mathbf{Q}_2 \equiv (A_1 + A_2, b_1 + b_2, c_1 + c_2)$, and the multiplication of a quadric by a scalar is computed by multiplying each component: $\lambda(A, b, c) \equiv (\lambda A, \lambda b, \lambda c)$. It follows trivially that $(\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{s}) = \mathbf{Q}_1(\mathbf{s}) + \mathbf{Q}_2(\mathbf{s})$ and $(\lambda\mathbf{Q})(\mathbf{s}) = \lambda\mathbf{Q}(\mathbf{s})$.

2.2 Shape approximation

Our goal is to partition the input mesh into regions I_k (sets of vertices), such that each region geometry P_{I_k} is approximated by a sphere $\mathbf{s}_k$ and the integral of the squared distance from the mesh to its approximation is minimized. The cost of such a partition is

$$\mathfrak{C}(\{I_k, \mathbf{s}_k\}_k) = \sum_k \int_{\xi\in P_{I_k}} d^2(\mathbf{s}_k, \{p_\xi, n_\xi\}^\perp) d\sigma_\xi \qquad (3)$$

Note that since we target shape approximation, this energy does not enforce the spheres to remain strictly inside the shape. We equip each vertex v_i of the input mesh with its so-called barycentric cell P_i (opposite figure) given by one third of its adjacent triangles (denoted by $T_1(v_i)$), and define the squared $\mathcal{L}_2$ distance from a sphere $\mathbf{s}$ to P_i as the integral over P_i of the squared distance to $\mathbf{s}$:



$$d(\mathbf{s}, P_i)^2_{\mathcal{L}_2} = \int_{\xi\in P_i} d^2(\mathbf{s}, \{p_\xi, n_\xi\}^\perp) d\sigma_\xi$$

Note that the squared distance to the sphere is constant on each adjacent triangle t_j (since all oriented points on t_j describe the same plane) and is given by its spherical quadric, denoted by $\mathbf{Q}_{t_j}(\mathbf{s})$ (see Eq. 2). The squared distance from P_i to a sphere $\mathbf{s}$ is simply given by a weighted sum of the spherical quadrics $\mathbf{Q}_{t_j}$ of the triangles that are adjacent to the vertex v_i:

$$d(\mathbf{s}, P_i)^2_{\mathcal{L}_2} = \sum_{t_j\in T_1(v_i)} \frac{area(t_j)}{3}\mathbf{Q}_{t_j}(\mathbf{s}) \triangleq \mathbf{Q}_i(\mathbf{s}) \qquad (4)$$

Each region P_{I_k} being defined as the union of the barycentric cells of its vertices I_k, the squared distance from a sphere $\mathbf{s}$ to P_{I_k} is given by summing up the different squared distances: $d(\mathbf{s}, P_{I_k})^2_{\mathcal{L}_2} = \sum_{i\in I_k} d^2(\mathbf{s}, P_i) = \mathbf{Q}_{I_k}(\mathbf{s})$ where $\mathbf{Q}_{I_k} = \sum_{i\in I_k} \mathbf{Q}_i$ is the sum of the spherical quadrics of the cells of the vertices in the set I_k.

Using this notation, the cost of a partition $\{I_k, \mathbf{s}_k\}_k$, defined as the squared $\mathcal{L}_2$ distance from the input surface to the set of spheres, is

$$\mathfrak{C}(\{I_k, \mathbf{s}_k\}_k) = \sum_k \mathbf{Q}_{I_k}(\mathbf{s}_k) \qquad (5)$$

Quadric Minimization Each spherical quadric $\mathbf{Q} \equiv (A, b, c)$ has a global minimum (since A is a symmetric positive semi-definite matrix by construction), which may not be unique, however. Since we limit ourselves to the set of spheres that have a positive radius (spheres with a negative radius describe concavities, and are located outside the object, see Fig. 3 (iv)), the minimization of this quadric is performed in the half-space $\mathbb{R}^3 \times \mathbb{R}^+$.

The minimizer is given by $A^{-1} \cdot b$ if A is invertible. When the global minimizer in $\mathbb{R}^4$ is located in the other half-space (i. e., the fourth coordinate r is negative), the minimizer on the restriction (i. e., $r >= 0$) is found in the hyper-plane $r = 0$. Otherwise this minimizer would have a local neighborhood entirely contained in $r >= 0$, and would be therefore a local minimum, which is impossible since a non-degenerate quadric has exactly one local, hence global, minimum.

If A is not invertible, the set of minimizers is a vector space that can be of dimension 1, 2, or 3.

3 Approximation Algorithm

With our SQEM in hand, we now describe how to approximate, at a desired level of detail, an input triangle surface mesh with a sphere-mesh, using this metric to tailor a bottom-up decimation algorithm. Since a sphere-mesh models a surface as the outer boundary of the interpolation of its spheres, an ideal input to our approximation algorithm is a closed orientable surface. Nonetheless, as shown later (see Sec. 4), our approach can deal with flawed input including holes and non-manifold edges.

3.1 Basic Algorithm

Similar to [Garland and Heckbert 1997], we reduce the input mesh by collapsing its edges iteratively in a greedy fashion, ordering the reduction operations by the cost $\mathbf{Q}_I(s)$, I being the connected set of vertices collapsed altogether, and $\mathbf{s}$ being the sphere approximating the region.

When considering the collapse of an edge uv of the mesh, we create the quadric $\mathbf{Q}_{uv} = \mathbf{Q}_u + \mathbf{Q}_v$, find the sphere that best approximates the constructed region $\mathbf{s}_{uv} = argmin_{\mathbf{s}}\{\mathbf{Q}_{uv}(\mathbf{s})\}$, and set the corresponding collapse cost c_{uv} of uv to $\mathbf{Q}_{uv}(\mathbf{s}_{uv})$. The suggested edge-collapse $[uv] \rightarrow \mathbf{s}_{uv}$ is then put into a priority queue $\mathcal{Q}$ with its associated cost c_{uv}.

At first, the priority queue $\mathcal{Q}$ is initialized with all possible edge-collapses. When pruning the best element $[uv] \rightarrow \mathbf{s}_{uv}$ from $\mathcal{Q}$, the edge uv is collapsed, a new vertex is created with the corresponding quadric $\mathbf{Q}_{uv}$, all possible edge-collapses with its neighbors are put into the queue and former neighboring ones are removed. The algorithm stops when the number of vertices (i. e., sphere centers) to delete is reached.

When looking for the minimizer of $\mathbf{Q}_{uv} \equiv (A, b, c)$, several cases need to be considered:

- if A is invertible: we approximate the region with a sphere in the domain $\mathbb{R}^3 \times [0; R]$ (see Par. **Radius bound**).
- if A is not invertible (e. g., the region is planar): we approximate the region with a sphere along the segment $[uv]$, still restricting the radius to be in $[0; R]$ (i. e., in the domain $[uv] \times [0; R]$).

For the second case, we write $q = u + \lambda\vec{\mu}$ ($\vec{\mu} = \vec{uv}$) and $\mathbf{Q}(q, r) = \frac{1}{2}(\lambda, r)^t \cdot \tilde{A} \cdot (\lambda, r) - \tilde{b}^t \cdot (\lambda, r) + \tilde{c}$, with $\tilde{A} = \left[\begin{array}{c|c} \vec{\mu}^t \cdot A_{33}^{11} \cdot \vec{\mu} & {A_{43}^{41}}^t \cdot \vec{\mu} \\ \hline {A_{43}^{41}}^t \cdot \vec{\mu} & A_{44}^{44} \end{array}\right]$, $\tilde{b} = \left[\begin{array}{c} {b_3^1}^t \cdot \vec{\mu} - \vec{\mu}^t \cdot A_{33}^{11} \cdot u \\ \hline b_4^4 - {A_{43}^{41}}^t \cdot u \end{array}\right]$, and $\tilde{c} = c - {b_3^1}^t \cdot u + \frac{1}{2}u^t \cdot A_{33}^{11} \cdot u$ ($\tilde{A} \in \mathcal{S}^2, \tilde{b} \in \mathbb{R}^2, \tilde{c} \in \mathbb{R}$). This energy is a 2-dimensional quadric w.r.t. (λ, r) that we want to minimize on the domain $[0; 1] \times [0; R]$, and whose global minimizer in $\mathbb{R}^2$ is $(\lambda, r) = \tilde{A}^{-1} \cdot \tilde{b}$. If this minimizer does not belong to the square $[0; 1] \times [0; R]$, the minimizer to its restriction is once again located on its boundary (i. e., $\{\lambda = 0; r \in [0; R]\}$, $\{\lambda = 1; r \in [0; R]\}$, $\{r = 0; \lambda \in [0; 1]\}$ or $\{r = R; \lambda \in [0; 1]\}$), resulting in a simple second order polynomial minimization in dimension 1. If, for some reason (e. g., the region is planar), $\tilde{A}$ is not invertible, we collapse the edge to its mid-point ($\lambda = 1/2$), and find the optimal value for the radius in $[0; R]$, which is a problem that is, this time, always properly conditioned.

Additionally, similarly to Garland and Heckert [1997], we prevent edge-collapses that result in the inversion of the orientation of the triangles that are involved in the operation.

Mesh data structure We use the data structure proposed by De Floriani et al. [2004], that allows to encode manifold meshes with a minimum memory overhead while maintaining high performance when degenerating to a non-manifold mesh. The connectivity of the resulting mesh is directly induced by the input connectivity and the set of successive edge-collapses.

Radius bound Approximating a surface using a large sphere can be cumbersome if the surface portion is not large enough itself (too little information provided), and the resulting sphere can cover a large part of the outside of the object in this case. For example, an infinite number of spheres can fit a single plane, since the only requirement is that this plane touches the sphere.

To solve this problem, we bound the diameter of the sphere by the directional width $\mathcal{W}$ of the region [Gärtner and Herrmann 2001] (i. e., the smallest extent of the region, when considering all possible directions).

To do so efficiently, we pre-sample the unit sphere uniformly with a fixed number of directions $\vec{k_j}$ (30 in our implementation, plus the 3 canonical axes). Each region P_u stores its interval $\mathfrak{I}_j(P_u)$ along all directions $\vec{k_j}$ ($\mathfrak{I}_j(P_u) = [m_u^j; M_u^j]$ with $m_u^j = \min_{x \in P_u}(x^t \cdot \vec{k_j})$ and $M_u^j = \max_{x \in P_u}(x^t \cdot \vec{k_j})$), and the intervals of the union of two regions P_u and P_v can be obtained by iterating over all directions $\vec{k_j}$ ($\mathfrak{I}_j(P_u \cup P_v) = [\min(m_u^j, m_v^j); \max(M_u^j, M_v^j)]$). Since, at first, each region P_i is composed of only the barycentric cell of one vertex v_i, these are initialized with $\mathfrak{I}_j(P_i) = [\min_{x \in P_i}(x^t \cdot \vec{k_j}); \max_{x \in P_i}(x^t \cdot \vec{k_j})]$. The directional width $\mathcal{W}(P)$ of the region P is approximated by $\tilde{\mathcal{W}}(P) = \min_j |\mathfrak{I}_j(P)|$. To allow coarser approximations at the first stages of the reduction, we set the maximum diameter of the sphere representing a region P_u to be slightly larger ($R(P_u) = \frac{3}{4}\tilde{\mathcal{W}}(P_u)$ in practice).

This bounding heuristic shows several benefits: (i) a sphere representing a planar region is constrained to be a point, as the directional width of a plane is zero; (ii) for convex shapes, a sphere is *likely* (although not guaranteed) to remain inside the shape when it is tangent to the geometry (as the directional width of a set equals that of its convex hull [Gärtner and Herrmann 2001]). Finally, this heuristic proved empirically to be more efficient than other alternatives such as bounding boxes, either axis-aligned or aligned according to the principal component analysis of the region.

Neighborhood enrichment As for all decimation-based simplification methods, collapsing two vertices of the mesh v_a and v_b is often useful if they are close enough in the original mesh i.e., $|v_a - v_b| < \epsilon$. Although it may introduce topological changes,

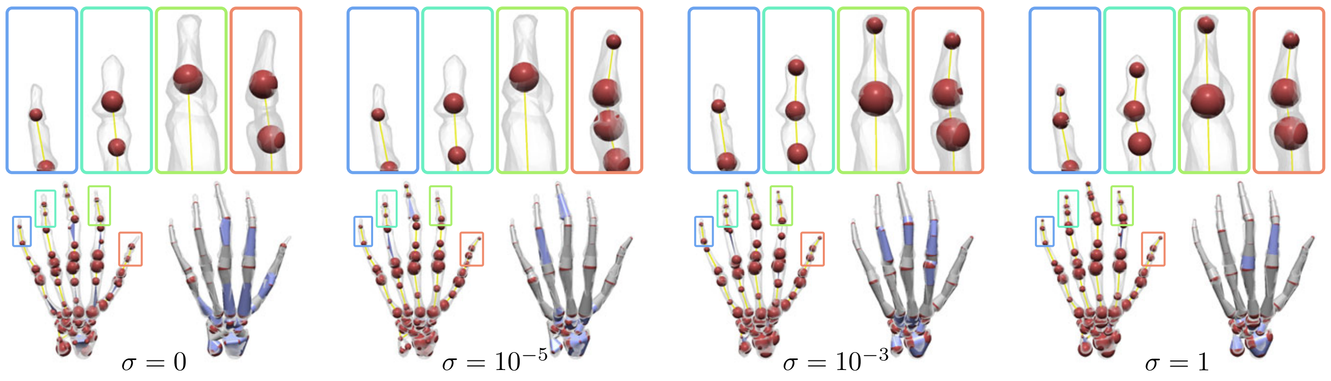


Figure 4: ***Influence of the* importance *parameter* σ:** *Fine details are better preserved as we increase the influence of the total curvature kernel. When σ is null (left), the results correspond to the standard remeshing introduced in Sec. 3. All sphere-meshes have 77 spheres.*

this strategy is especially effective if the shape contains large opposite regions that should be collapsed together (e.g., large shell-like parts such as the chair model or the wings of the Pegaso in Fig. 5). The parameter ϵ is intuitive and part of common decimation-based simplification frameworks.

3.2 Importance-driven distribution

In the context of shape approximation, it is common to allow the user to control the *relative importance* of the features in the process. We propose to parameterize our approximation process by introducing a weighting kernel K_σ in the definition of the cost of a partition (previously defined in Eq. 3):

$$\mathfrak{C}_\sigma(\{I_k, \mathbf{s}_k\}_k) = \sum_k \int_{\xi \in P_{I_k}} K_\sigma(\xi) d^2(\mathbf{s}_k, \{p_\xi, n_\xi\}^\perp) d\sigma_\xi \quad (6)$$

In the previous definition, K_σ describes the respective importance of all points on the manifold. If $K_\sigma = 1 \; \forall \xi$, the formulation of the cost of a partition reduces to the original one previously introduced in Eq. 3.

The spherical quadric $\mathbf{Q}_i$ of the vertex v_i is given by averaging the quadrics $\mathbf{Q}_{t_j}$ of its adjacent triangles $t_j \in T_1(v_i)$ as before, this time taking the integral of the *kernel* into account:

$$\mathbf{Q}_i = \sum_{t_j \in T_1(v_i)} \left(\int_{\xi \in P_i \cap t_j} K_\sigma(\xi) d\sigma_\xi\right) \mathbf{Q}_{t_j} \quad (7)$$

In Fig. 4 we present various results obtained when setting importance kernels based on the total curvature ${\kappa_1}^2 + {\kappa_2}^2$ (which naturally favors highly-protruding geometry):

$$K_\sigma(\xi) = 1 + \sigma \cdot BBD^2 \cdot (\kappa_1(\xi)^2 + \kappa_2(\xi)^2) \quad (8)$$

BBD being the bounding box diagonal of the model. The multiplicative factor BBD^2 is used in Eq. 8 to ensure scale invariance.

For affine kernels, the various integrals $\int_{\xi \in P_i \cap t_j} K_\sigma(\xi) d\sigma_\xi$ can be computed trivially. Note that $\kappa_1(\xi)$ and $\kappa_2(\xi)$ are not piecewise linear, if one considers that the two-dimensional curvature tensor should be the one quantity that should be interpolated linearly on the triangles when considering discrete manifolds. In our implementation however, we consider that the total curvature is linearly interpolated on the triangles to simplify the computation of the kernel integrands.

Other properties, e. g., local feature size [Amenta and Bern 1999] or conformal factor [Ben-Chen and Gotsman 2008], could be used as importance kernels. We focused on the total curvature-based kernels and left alternative ways to control local importance for future work.

4 Results

We implemented our shape approximation method in C++ and report performances on an Intel Core2 Duo running at 2.5 GHz with 4GB of main memory. The entire algorithm is controlled by two parameters: the target number of spheres and σ (set to 1 unless otherwise mentioned).

4.1 Interpolated geometry

In this section, on top of sphere-mesh structures, we provide sphere-mesh geometries corresponding to the linear interpolation of spheres along edges and triangles. This interpolated geometry corresponds to the union of spheres, interpolated edges and interpolated triangles. Each interpolated edge corresponds to a cone cut by a plane orthogonal to the edges at each extremity and each interpolated triangle corresponds to a triangular prism made of 3 faces extruding the triangle edges and 2 triangles for the lower and upper crusts of the interpolation. Alternatively, a discrete interpolation can be generated by sampling many spheres along edges and triangles, in the manner of the ZBrush tool [Pixologic 2001].

4.2 Performances

In Fig. 5, we present results of our automatic shape approximation method for 22 different input models, covering a wide range of geometric features, topology and quality. For all examples but the last, we show the input shape, the resulting sphere-mesh structure (sphere, edges, triangles) and the shape approximation emerging from the linear interpolation of the spheres along the structure (interpolated geometry in grey for edges and blue for triangles). We can observe that, even under a drastic approximation restricted to tens of spheres, each sphere-mesh succeeds at capturing, in an adaptive manner, the essence of the shape. This becomes even clearer when displaying the interpolated sphere-mesh geometry: spherical, conic and cylindrical components are quickly captured with the sphere-mesh, even in the presence of numerous fine scale features. We can also observe a number of cases where a tubular structure

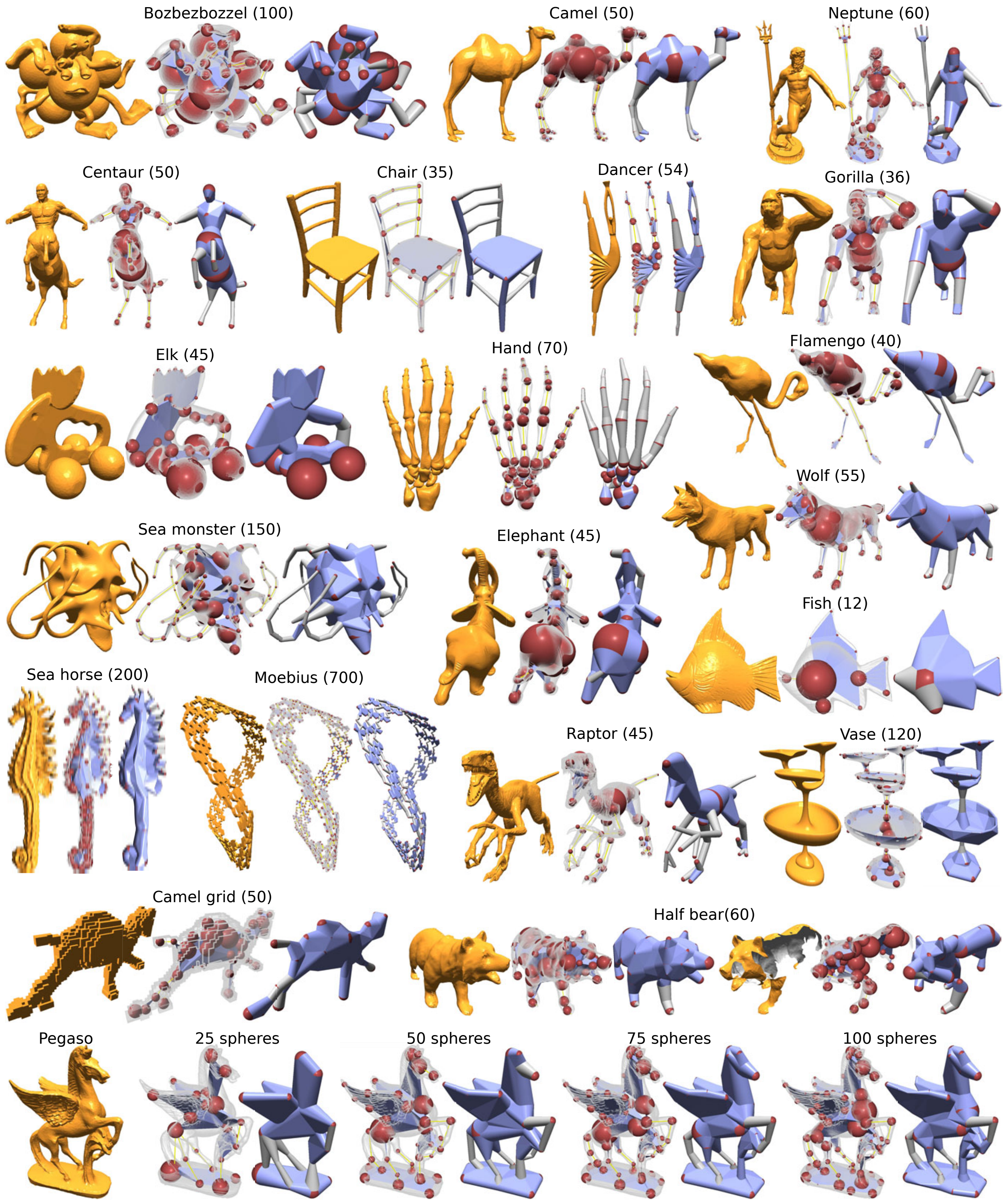


Figure 5: ***Sphere-mesh approximation results*** *with the algorithm described in Sec. 3 on various meshes at different levels of simplification. Results obtained with $\sigma = 1$. The number of spheres of the approximation is shown between the parentheses.*

Input Model (#V / #T)	Init. (ms)	Dec. (ms)	Sphere-mesh (#S / #E / #T)	H	M12	M21	QEM Simplification (#V / #T)	H	M12	M21
Boz. (22774 / 45564)	1701	1844	100 / 22 / 146	**3.020**	**0.472**	**0.438**	100 / 194	11.703	1.519	0.591
Camel (2020 / 4040)	122	113	50 / 19 / 34	**7.201**	**0.465**	**0.384**	50 / 96	12.477	1.071	0.576
Neptune (28052 / 56112)	1714	2358	60 / 15 / 72	**2.650**	**0.384**	**0.388**	60 / 124	10.186	0.943	0.518
Centaur (3401 / 6796)	209	212	50 / 17 / 42	**3.557**	**0.373**	**0.401**	50 / 94	15.626	1.629	0.523
Chair (9935 / 19894)	568	745	35 / 16 / 22	**1.517**	**0.313**	**0.345**	35 / 72	14.974	1.404	0.364
Dancer (7942 / 15884)	472	551	54 / 11 / 67	**1.352**	**0.166**	**0.182**	54 / 102	5.970	0.517	0.319
Gorilla (1917 / 3830)	121	112	36 / 5 / 37	**6.109**	**0.530**	**0.446**	36 / 68	9.353	1.610	1.122
Elk (5194 / 10388)	309	370	45 / 2 / 69	**2.560**	**0.273**	**0.289**	45 / 90	20.802	2.199	0.936
Hand (11724 / 23464)	696	884	70 / 53 / 16	**1.980**	**0.324**	**0.352**	70 / 130	14.074	1.140	0.561
Flamengo (26394 / 52839)	1403	1941	40 / 12 / 32	**2.837**	**0.417**	**0.479**	40 / 74	35.135	5.738	0.507
Sea monster (39485 / 78966)	2210	3169	150 / 60 / 161	**6.081**	**0.367**	**0.364**	150 / 292	11.445	0.947	0.402
Elephant (24955 / 49918)	1263	1818	45 / 6 / 68	**3.357**	**0.564**	**0.620**	45 / 90	6.371	1.038	0.684
Wolf (3401 / 6796)	220	212	55 / 12 / 75	**3.618**	**0.380**	**0.405**	55 / 106	8.144	1.079	0.441
Fish (7376 / 14748)	532	521	12 / 3 / 6	**4.272**	**0.762**	**0.735**	12 / 20	14.661	1.395	1.036
Sea horse (162248 / 324524)	18859	16271	200 / 18 / 344	**1.324**	**0.257**	0.271	200 / 400	4.668	0.324	**0.231**
Moebius (21126 / 42523)	1275	1615	700 / 316 / 366	**0.762**	**0.078**	**0.074**	700 / 1510	1.675	0.242	0.123
Raptor (12908 / 25852)	1423	879	45 / 23 / 30	**4.310**	**0.428**	**0.425**	45 / 84	18.069	1.555	0.577
Vase (51801 / 103598)	6442	5203	120 / 8 / 197	**2.093**	**0.321**	0.395	120 / 230	4.698	0.539	**0.265**
Camel grid (5752 / 11508)	296	503	50 / 5 / 72	**4.199**	**0.900**	**0.629**	50 / 92	17.104	1.853	0.730
Half bear (9202 / 17969)	530	642	60 / 5 / 66	14.784	**0.400**	2.559	60 / 80	**14.296**	0.939	**0.362**
Pegaso (15319 / 30658)	899	1153	100 / 11 / 170	**2.554**	**0.381**	**0.393**	100 / 204	6.113	0.578	0.424
–	–	–	75 / 12 / 119	**4.676**	**0.485**	**0.479**	75 / 148	6.835	0.790	0.496
–	–	–	50 / 12 / 71	**4.973**	**0.621**	**0.600**	50 / 94	7.554	1.179	0.634
–	–	–	25 / 7 / 34	**6.128**	**1.237**	1.081	25 / 46	21.096	2.478	**1.056**

Table 1: *Performance and timings for our sphere-mesh approximation algorithm (models of Fig. 5). All models were computed with $\sigma = 1.0$. The initialization time comprises the mesh structure construction from a file and the initialization of the priority queue with all possible edge-collapses. Decimation is performed until no edge remains (computation of the whole multi-resolution structure).* #S / #E / #T*: number of spheres, wire edges, and triangles in the output sphere-mesh. We compare our approximation with QSlim for the same number of primitives (smallest error in **bold**).* H*: Hausdorff distance.* M21*: mean distance from the approximation to the original model.* M12*: mean distance from the original model to the approximation. All distances are expressed in percentages of the input model bounding box diagonal.*

would not properly capture the shape at coarse scales (e. g., *Chair*, *Elk* and *Vase* models), highlighting the usefulness of polygons (on top of edges) in the sphere-mesh representation.

Our approach robustly handles fine components (e. g., *Flamingo* model), complex topologies (e. g., *Moebius* model) and non-uniformly distributed geometric structures with rapidly varying sizes (e.g., *Sea horse* model). Indeed, one desirable property of extreme approximation methods is the ability to ignore small structures to quickly "abstract" a complete shape component with a single primitive. In this context, we can observe that near spherical components are promptly captured with a single sphere (e. g., *Elk* and *Elephant* models), while near tubular structures are modeled with edge chains (e.g., arms and legs). The last row of Fig. 5 illustrates the natural multi-resolution structure that comes with our approach, with smaller components emerging progressively while reaching finer level-of-details.

We also performed experiments on pathological cases: the *Camel grid* model shows how a poor quality input mesh, with numerous shape singularities, is smoothly approximated at coarse scale with a sphere-mesh; the *Half bear* model illustrates how the algorithm behaves for incomplete data sets (right part of the model). Note in particular that the shape approximation quality is not damaged in the regions where the input is complete.

Additionally, we analyze the influence of noise in Fig. 6: although the global structure of the approximation is preserved when adding more and more noise, it is the local volume approximation which suffers the most from the input quality degradation. Indeed, when the input is very noisy, the SQEM minimization leads naturally to zero radius spheres (i.e., points), and thus becomes equivalent to QEM minimization in that case. The impact of σ also becomes less critical.

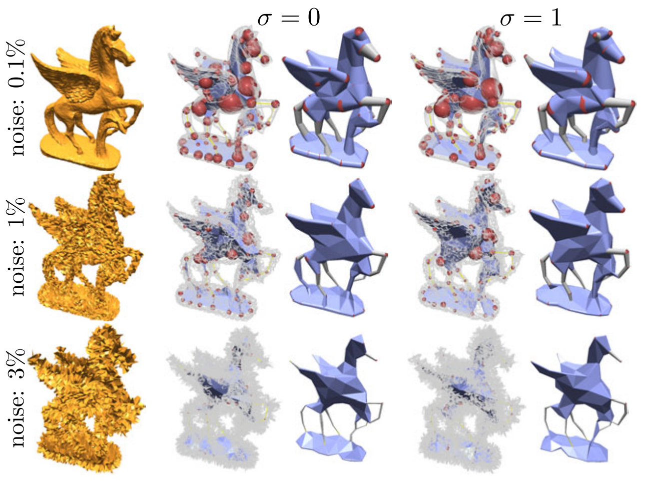


Figure 6: ***Noise sensitivity:*** *evolution of the sphere-mesh approximation with an increasing amount of noise (random per-vertex displacement expressed in percentage of the bounding box diagonal).*

Worst case scenarios Fig. 7 shows what we identified to be the worst case scenarios for our approach: thin-shell models containing large concave parts and disconnected components encompassing each other.

At the final stages of the simplification, spheres fitting a large region with low curvature can bulge out from the shape (red boxes in Fig. 7) due to the locality of the surface optimization process. The *Concentric Spheres* example represents a thin-shell sphere with the outer sphere approximated by a single large sphere, which covers

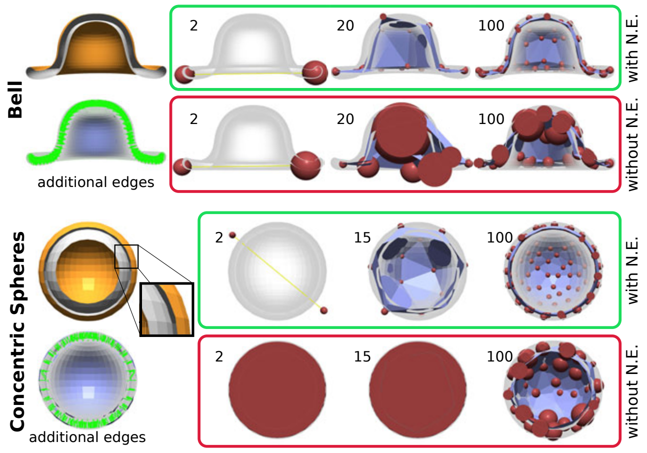


Figure 7: ***Results on thin shell models*** *(cross sections with back faces shown in grey).* Green boxes: *Results using our* neighborhood enrichment *strategy (additional potential collapses in green).* Red boxes: *Results using the initial connectivity, without* neighborhood enrichment*.* ***Top:*** **Bell** ***model.*** *Using the* neighborhood enrichment *allows thin shell volumes to be approximated with single sheets made of double sided triangles, mimicking the MAT.* ***Bottom:*** **Concentric Spheres** ***model.*** *Without* neighborhood enrichment*, the two disconnected spheres will be simplified independently.*

its inside entirely and ignores the inner concave sphere. A proper sphere-mesh approximation of such a thin shell can again be obtained using the *neighborhood enrichment* strategy (green boxes in Fig. 7) which allows to collapse vertices which are not explicitly linked by an edge but close enough (green edges in the figure) and on opposite sides of the shell. This results in topological changes in the mesh, eventually leading to double sided triangles – similar to the medial axis. Such collapses are likely to be favored at the early stages of the simplification, because the "Radius Bound" heuristic prevents the apparition of large spheres approximating a low curvature surface with small directional width. Collapses of vertices that are "facing" each other are unlikely to happen, since the SQEM accounts intrinsically for the normal orientation; we also prevent linking them during the *neighborhood enrichment* based on their normals. Lastly, we prevent explicit triangle inversion.

Finally, a large number of spheres are required to approximate such shapes (20-100 for the *Bell*, 15-100 for the *Concentric Spheres*), because the sphere-mesh representation captures shapes with few connected convex components better.

4.3 Comparisons

Mesh decimation We compare our method to QSlim which implements a mesh simplification algorithm based on the *quadric error metric* [Garland and Heckbert 1997] (QEM), and also offers a multi-resolution approximation of the input in the form of coarser triangle meshes. Fig. 8 illustrates our claim: when a sufficient number of primitives is allowed, traditional triangle meshes simplifications approximate faithfully enough the input 3D object. However, as the number of primitives diminishes, meaningful parts completely vanish. Our sphere-mesh approximation captures similarly well the objects with a high number of primitives, but degrades gracefully to volumetric structures even when drastically decimated, preserving the main parts of the objects, which is a desirable behavior for applications requiring high level shape abstractions (e. g., shape modeling, see Sec. 5).

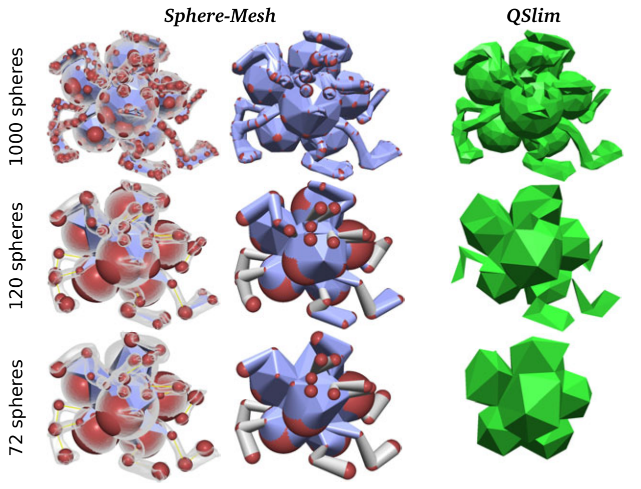


Figure 8: ***Comparison to polygonal simplification:*** *evolution of the approximation with a decreasing number of spheres (resp. vertices) for the sphere-mesh (resp. simplified mesh). From left to right: sphere mesh approximation with semi-transparent original surface, interpolated sphere-mesh geometry and QSlim (Garland and Heckbert 97) polygonal simplification in green.*

In Tab. 1, we report the approximation error of QEM simplifications, for the same number of primitives as with our SQEM method. Beyond the visual assessment, this experimental study shows that the approximation quality of our sphere-meshes clearly outperforms the one of decimated polygon meshes, both for Hausdorff and mean distances, for most examples. However, in the case of incomplete inputs (*Half bear* model), the mean distance from the approximation to the input surface (M21) is significantly higher using a sphere-mesh: indeed, the sphere-mesh geometry interpolation tends to "fill" holed regions which, depending on the application, may be a benefit or a drawback.

Medial axis transform The medial axis transform (MAT) is not designed to represent shapes with the same number of primitives as sphere-meshes, which become volumetric structures only at very coarse simplification levels (see Fig. 10). The extraction parameters also differ: while we allow the user to tune the final number of primitives, typical MAT extraction techniques [Amenta et al. 2001] propose to tune the reconstruction error rather than the number of polar spheres.

Nevertheless, sphere-meshes and the MAT share geometric and structural similarities: First, the MAT is defined as well in terms of sphere interpolation over a non-manifold structure composed of triangles and edges; second, the MAT can offer as well a multi-resolution description of the input shape, through a filtering process. For instance, Tam and Heidrich [2003] suggest to remove *entire manifold sheets* of the medial axis iteratively, based on the volume each one carries in the final reconstruction. Attali and Montanver [1996] propose to filter medial spheres based on the angle formed by their two closest boundary points w.r.t. their center; whereas Chazal and Lieutier [2005] use the circumradius of the two closest boundary points instead. Alternatively, Miklos et al. [2010] compute a filtered medial axis as the medial axis of a set of *scaled* spheres.

Beside these similarities, there are at least two main differences between our approach and MAT (see Fig. 10): First, sphere-meshes degenerate from surface to volumetric structures progressively and parts of the sphere-mesh geometry can remain homeomorphic to

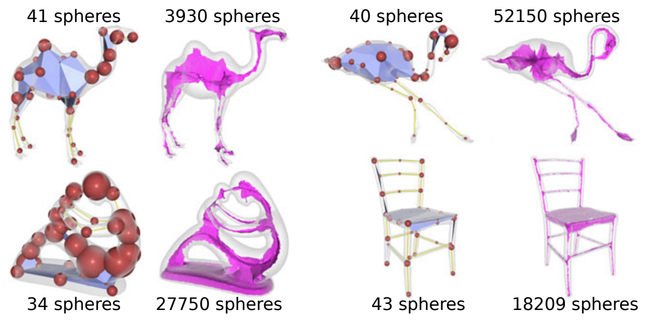


Figure 10: ***Visual comparison to the medial axis transform*** *(pink). Sphere-meshes obtained with $\sigma = 0$. Medial axes extracted with Powercrust [Amenta et al. 2001].*

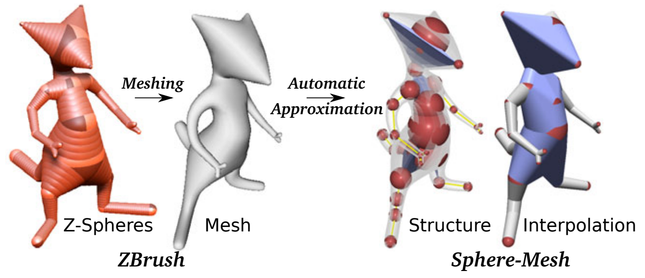


Figure 11: ***Comparison with sphere skeletons:*** *On the left, a sphere skeleton constructed with ZBrush and its associated mesh surface (i.e., ZBrush skin). On the right, a sphere-mesh (30 spheres) with edges and triangles, automatically computed from this surface.*

the input surface (see bodies of *Flamengo* and *Camel* in Fig. 10 for instance). Moreover, this transition from surface to volumetric structures arises in an adaptive manner (see the body of the *Flamengo* sphere-mesh compared to its legs in Fig. 10). In contrast, the MAT is a purely volumetric structure. Second, although a filtering process can also provide a MAT-based multi-resolution description of the input shape, this comes as a natural side effect of our approximation technique. In our case, we simplify the input mesh directly, in a progressive and continuous manner, instead of simplifying a precomputed medial structure (that can be even more complex than the input shape itself). Indeed, the MAT is not designed to reach the simplification levels we obtain: existing filtering techniques focus on denoising the medial axis rather than coarsening it.

Sphere skeletons In contrast to sphere-meshes, which target the approximation of an existing high resolution mesh (e.g., scanned object), sphere-skeletons such as Z-Spheres [Pixologic 2001] or B-Meshes [Ji et al. 2010] are designed for interactive shape creation and facilitate the creation of coarse shapes from scratch. However, both representation models can be compared in terms of flexibility and expressiveness. To some extent, the sphere-mesh representation extends Z-Sphere and B-Meshes beyond tubular structures. The presence of polygons (in addition to edges) in the sphere-mesh topological structure helps modeling large flat regions which cannot be captured by tubular components. The *chair* example in Fig. 5 illustrates this notion: while most thin parts are approximated using edges, the seat is captured by a sphere interpolation over polygons. Using a sphere skeleton (e.g., Z-Spheres), this region would either be more complex to model or less accurate in terms of approximation. Indeed, sphere skeletons methods often come with a specific surface extraction method, to pursue the modeling session with displacement painting for instance. Sphere-meshes allow the reversal of the pipeline, by recovering automatically a sphere-based structure from a detailed input surface (see Fig. 11).

5 Application to Deformation Control

Beyond shape approximation, a sphere-mesh degenerates naturally into an internal structure – eventually into a skeleton in tubular regions – which is a convenient metaphor for a number of interactive shape modeling applications. Alternatively to skeletons and cages, a sphere-mesh can for instance be used as an automatic high-level structure for controlling a shape. In this section, we explain how to tie a mesh to its sphere-mesh and use the latter to interactively control the deformation of the former in a multi-resolution fashion.

Overview Given a mesh, its sphere-mesh and a skinning machinery (e. g., linear blends [Lewis et al. 2000]), we compute a skinning of the mesh establishing a mesh/sphere-mesh relationship and provide the user with interactive *control primitives* in the form of spheres, edges and triangles from the sphere-mesh (see Fig. 9). The mesh geometry is updated in real time according to the skinning and the current sphere-mesh layout i.e., translations, rotation and scaling prescribed by the user on the primitives. Moreover, the user can instantly change the editing scale by navigating the sphere-mesh hierarchy (see Fig. 12). After each deformation, the sphere-mesh is updated to fit the deformed geometry. This framework, as well as providing an *automatic* multi-resolution control structure (while most deformation cages are constructed manually for instance), combines interesting properties of several alternative methods: (i) as with skeleton-based systems, elongated parts (e.g., arms, legs) can be bent by simply setting rotations on selected spheres; (ii) as with cages [Lipman et al. 2008], the volume of a region can be smoothly controlled, using spheres radii; (iii) as

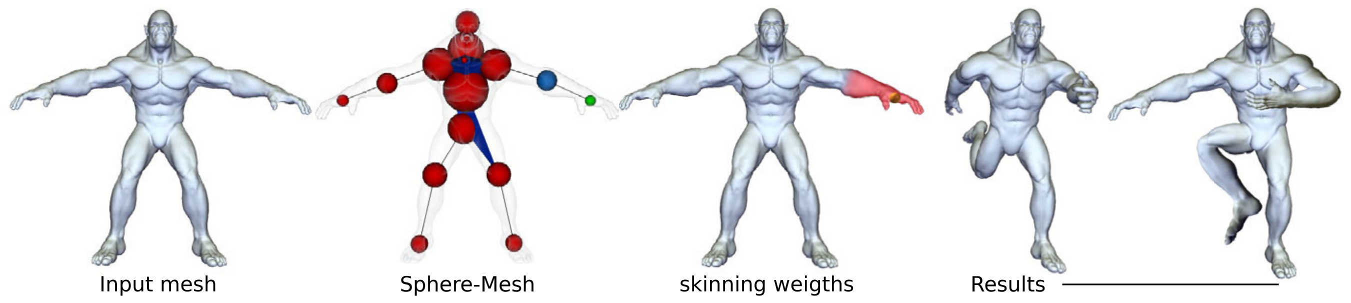


Figure 9: ***Sphere-mesh-based deformation:*** *starting from a surface mesh (left), a sphere-mesh is automatically computed (middle left) and the input mesh is skinned to its elements (middle). The user can then manipulate the sphere-mesh to smoothly deform the input (right).*

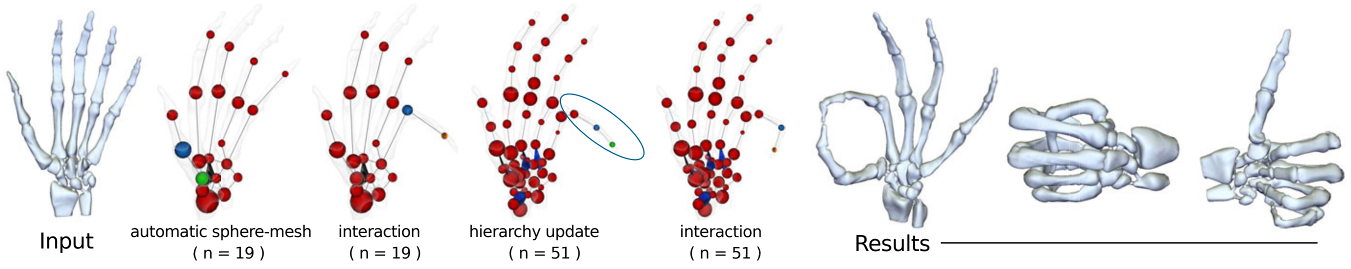


Figure 12: ***Sphere-mesh based deformation****: a multi-resolution editing session, where the sphere-mesh acts as an automatic multi-resolution control structure, refined on-demand to match the desired deformation scale.*

with multi-resolution mesh editing [Zorin et al. 1997], the user can quickly go from large scale deformation to tuning fine details. The good behavior of sphere-meshes at extreme simplification levels is a key feature here.

Skinning To illustrate the use of sphere-meshes as control structures, we implemented a deformation machinery based on the work of Baran and Popović [2007]. More precisely, we define the weights w_i^j (weight of the vertex i w.r.t. the control primitive j) as the solution to the following system: $(\Delta + H) \cdot w^j = H \cdot p^j$ where Δ is the discrete surface Laplacian; w^j is the vector of weights w_i^j for all vertices i (of size n); $H = \lambda I_n$ is a diagonal matrix with λ being a value describing the diffusion of the weights over the mesh (the smaller λ, the bigger the diffusion is performed); and p^j is a vector with $p_i^j = 1$, if j is the closest control primitive to vertex i, 0 otherwise. We limit the search of the closest primitive to the ones adjacent to the sphere to which the vertex corresponds in the clustering of the original mesh. The result of this linear system is a set of weights w_i^j that sum up to 1 for all vertices i [Baran and Popović 2007]. The left hand-side matrix of the linear system $\Delta + H$ is sparse and independent of j, we can therefore factorize the system once and solve it iteratively over each primitive j to obtain the weights of the whole input mesh w.r.t. j. As a result, the user can easily define smooth deformations on the mesh by applying rigid transformations on the primitives of its sphere-mesh approximation (see Fig. 9).

Multi-resolution freeform deformation To turn the previous framework into a multi-resolution one, we record, at each edge-collapse of the initial sphere-mesh approximation (see Sec. 3), the positions and radii of the two collapsed vertices and of the resulting one, as well as the set of edges/triangles that were created/deleted. We structure these events in a binary tree so that the user can navigate through the so-defined hierarchy to gather the desired control structure intuitively (i. e., level-of-detail sphere-mesh).

At deformation time, the user simply applies rigid transformations to the individual primitives of the sphere-mesh at a chosen level of detail and gets a real-time feedback of the induced smooth deformation on the original high-resolution mesh (see Fig 12). Since the left hand side of the linear system $(\Delta + H)$ does not depend on the control structure, it does not need to be re-factored, and the update of the whole structure as well as the computation of new weights can be performed in real-time. Each time the user switches the current editing level of detail, the spheres in the hierarchy are updated bottom-up, computing new SQEM $\mathbf{Q}_i$ at each level of the hierarchy. This update step is necessary to ensure that the entire sphere-mesh hierarchy fits the deformed geometry of the mesh, and not the initial one. In practice, we start by computing the new SQEM for each leaf (i.e., mesh vertex) of the tree based on the current geometry of the mesh. For all recorded collapses (i.e., internal tree node) $uv \rightarrow w$, the sphere $\mathbf{s}_w$ of the parent w is obtained by minimizing children quadrics, i. e., $\mathbf{Q}_w = \mathbf{Q}_u + \mathbf{Q}_v$, and $\mathbf{s}_w = argmin_{\mathbf{s}} \mathbf{Q}_w(\mathbf{s})$. This tree update is linear in the number of input vertices and is performed in a few milliseconds on models made of several hundreds of thousands vertices, as only the geometry of the spheres requires an update, while the topology of the initial sphere-mesh remains unchanged.

Although the skinning method of Baran and Popović works well in our experiments, any alternative skinning scheme may be used with our structure, thus providing a multi-resolution control interface for all linear blend skinning (LBS) techniques.

6 Discussion

6.1 Limitations & Future work

Representation Currently, a sphere-mesh represents the surface as a base complex (vertices, triangles, ...) and a spatially-varying thickness (spheres radius), but the actual surface geometry is expressed as the linear interpolation of the spheres over the complex and is not directly transformable into e. g., a triangle mesh. "Closing the loop" and extracting a high quality mesh from the sphere-mesh boundary is an interesting direction for future work.

Incomplete data and open surfaces As shown in the result section, incomplete surfaces with large holes often see their holes filled by the sphere-mesh geometry. Indeed, there is no explicit modeling of boundaries in the sphere-mesh representation and defining a restriction of the interpolated geometry to an open 2-manifold is left as future work.

Shape approximation Currently, our definition of sphere-meshes does not allow to enforce, in a simple manner, that spheres remain strictly inside the shape or that the resulting mesh preserves the input's topology i.e., topological errors can occur at extreme simplification levels. Furthermore, sphere-meshes represent shapes with few connected convex components better, and models featuring large concave parts still require a large number of spheres to be modeled properly.

Optimization Our approximation algorithm worked well on all the examples we tried but does not, unfortunately, provide the global minimizer. Indeed, our approximation strategy is limited by the fact that we fit spheres to regions, but do not optimize for the interpolation between the spheres, which is our major research direction for future work. In some cases, the interpolated sphere-mesh geometry can be a poor approximation of the input surface even if the spheres fit their respective regions correctly. Bottom-up algorithms (e. g., [Garland and Heckbert 1997]) typically do not optimize for the connectivity at each step, which is not optimal and could benefit, for instance, from intermediate try-and-test steps. However, this would require efficiently computing the input/sphere-mesh distance for each try, dramatically increasing the computational complexity of the algorithm.The partitioning is also computed by minimizing the one way $\mathcal{L}_2$ distance from the input mesh to its simplified geometry, and the results could be further improved by minimizing the Hausdorff distance instead. Lastly, we use a bottom-up reduction of the input mesh connectivity as the backbone of our approximation algorithm. Alternative mechanisms, such as variational or statistical sphere distributions may be derived to exploit our SQEM differently.

6.2 Conclusion

We proposed a shape approximation algorithm based on a sphere interpolation over a mesh substructure, for capturing complex shapes with a reduced set of connected spheres. The main technical contribution is the SQEM whose minimization finds spheres representing best the tangent spaces of a set of polygons. We also proposed an algorithm for efficiently computing a sphere-mesh approximating a triangle mesh, with a natural multi-resolution structure and an explicit control on its feature sensitivity. We analyzed its performance on a diverse set of inputs, showing that a sphere-mesh approximation performs in general better than mesh decimation. Lastly, we proposed an application of sphere-meshes to shape modeling by using them as automatic multi-resolution control structures for interactive freeform deformation. Beyond the possible technical improvements discussed earlier, we believe that sphere-meshes can be further developed for shape processing and analysis methods, including reconstruction from point clouds, progressive compression, shape recognition and visualization scenarios.

Acknowledgements This work has been partially funded by the European Commission under contract FP7-287723 REVERIE and FP7-323567 Harvest4D, by the ANR iSpace&Time project and by the Chaire MODIM of Telecom ParisTech. We thank the anonymous reviewers for their remarks and suggestions.

References

ALLIEZ, P., DE VERDIÈRE, E. C., DEVILLERS, O., AND ISENBURG, M. 2003. In *Shape Modeling International, 2003*, IEEE, 49–58.

AMENTA, N., AND BERN, M. 1999. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry 22*, 4, 481–504.

AMENTA, N., CHOI, S., AND KOLLURI, R. 2001. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, ACM, 249–266.

ATTALI, D., AND MONTANVERT, A. 1996. Modeling noise for a better simplification of skeletons. In *Image Processing, 1996. Proceedings., International Conference on*, vol. 3, IEEE, 13–16.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. In *ACM Transactions on Graphics (TOG)*, vol. 26, ACM, 72.

BEN-CHEN, M., AND GOTSMAN, C. 2008. Characterizing shape using conformal factors. In *Eurographics workshop on 3D object retrieval*, 1–8.

BLUM, H. 1967. A Transformation for Extracting New Descriptors of Shape. In *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed. MIT Press, Cambridge, 362–380.

BRADSHAW, G., AND O'SULLIVAN, C. 2002. Sphere-tree construction using dynamic medial axis approximation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 33–40.

BRADSHAW, G., AND O'SULLIVAN, C. 2004. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics (TOG) 23*, 1, 1–26.

CHAZAL, F., AND LIEUTIER, A. 2005. The λ-medial axis. *Graphical Models 67*, 4, 304–331.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. In *ACM Transactions on Graphics (TOG)*, vol. 23, ACM, 905–914.

DE FLORIANI, L., MAGILLO, P., PUPPO, E., AND SOBRERO, D. 2004. A multi-resolution topological representation for non-manifold meshes. *Computer-Aided Design 36*, 2, 141–159.

DEY, T., AND ZHAO, W. 2004. Approximate medial axis as a voronoi subcomplex. *Computer-Aided Design 36*, 2, 195–202.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 209–216.

GÄRTNER, B., AND HERRMANN, T. 2001. Computing the width of a point set in 3-space.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. In *ACM SIGGRAPH*, 19–26.

JAMES, D. L., AND PAI, D. K. 2004. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (TOG) 23*, 3, 393–398.

JI, Z., LIU, L., AND WANG, Y. 2010. B-mesh: A modeling system for base meshes of 3d articulated shapes. In *Computer Graphics Forum*, vol. 29, Wiley Online Library, 2169–2177.

LEWIS, J., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 165–172.

LINDSTROM, P. 2000. Out-of-core simplification of large polygonal models. In *ACM SIGGRAPH*, 259–262.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Transactions on Graphics (TOG) 27*, 3, 78:1–78:10.

LU, L., CHOI, Y., WANG, W., AND KIM, M. 2007. Variational 3d shape segmentation for bounding volume computation. In *Computer Graphics Forum*, vol. 26, Wiley Online Library, 329–338.

MIKLOS, B., GIESEN, J., AND PAULY, M. 2010. Discrete scale axis representations for 3d geometry. *ACM Transactions on Graphics (TOG) 29*, 4, 101.

PIXOLOGIC, 2001. Zbrush.

ROSSIGNAC, J., AND BORREL, P. 1993. Multi-resolution 3d approximation for rendering complex scenes. *Modeling in Computer Graphics*, 455–465.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH*, 343–352.

SCHAEFER, S., AND WARREN, J. 2003. Adaptive vertex clustering using octrees. In *Proceedings of SIAM Geometric Design and Computing*, 491–500.

STOLPNER, S., KRY, P., AND SIDDIQI, K. 2012. Medial spheres for shape approximation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 6, 1234–1240.

SUD, A., FOSKEY, M., AND MANOCHA, D. 2007. Homotopy-preserving medial axis simplification. *International Journal of Computational Geometry & Applications 17*, 05, 423–451.

TAM, R., AND HEIDRICH, W. 2003. Shape simplification based on the medial axis transform. In *Visualization, 2003. VIS 2003. IEEE*, IEEE, 481–488.

TURK, G. 1992. Re-tiling polygonal surfaces. *Computer Graphics (SIGGRAPH 92) 26*, 2 (July), 55–64.

WANG, R., ZHOU, K., SNYDER, J., LIU, X., BAO, H., PENG, Q., AND GUO, B. 2006. Variational sphere set approximation for solid objects. *The Visual Computer 22*, 9, 612–621.

WU, J., AND KOBBELT, L. 2005. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum 24*, 3, 277–284.

YAN, D.-M., LIU, Y., AND WANG, W. 2006. Quadric surface extraction by variational shape approximation. In *Geometric Modeling and Processing*, vol. 4077 of *Lecture Notes in Computer Science*. 73–86.

YAN, D.-M., LÉVY, B., LIU, Y., SUN, F., AND WANG, W. 2009. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Proc. Symposium on Geometry Processing*, 1445–1454.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *ACM SIGGRAPH '97*, SIGGRAPH '97, 259–268.