

Watertight skin meshes of three-dimensional R -shapes: exact distance, visualisation and mass properties

Vincent Richefeu

Laboratoire 3SR, Université Grenoble Alpes, CNRS, Grenoble INP, 38000 Grenoble, France

vincent.richefeu@univ-grenoble-alpes.fr

October 2026 · Code: <https://github.com/richefeu/rockable>

Abstract

In *Rockable*, a grain is an R -shape (a sphero-polyhedron): the Minkowski sum of a skeleton (vertices, edges and faces) and a ball of radius R . Only the skeleton and the radius are stored, so the surface of the grain is never built, and the viewers draw overlapping spheres, tubes and plates instead. This drawing fails with transparency and cannot be exported. This note documents the tools added to *Rockable* to build the surface explicitly. They rely on classical results of solid modelling and computer graphics, recalled with their references: the signed distance to an R -shape is the signed distance to its skeleton minus R , exact everywhere, with an analytic gradient; marching cubes on this function, followed by a Newton projection of each vertex onto the exact surface, gives a watertight triangle mesh whose only error is the chordal error set by one tolerance; a coplanar merge removes the extra triangles that marching cubes leaves on flat faces; and the divergence theorem applied to the mesh gives the volume, centre of mass and inertia tensor at second order in the grid step. For each step, the note gives the corresponding classes, functions, options and files of the code (*ShapeSDF*, *buildShapeMesh*, *simplifyCoplanar*, the *shape2mesh* tool and the *.rmsh* file read by the viewers), the validation tests shipped with the code, and the known limits.

Keywords: sphero-polyhedra; discrete element method; signed distance function; offset surface; marching cubes; mass properties; visualisation.

In short. `shape2mesh shapes.txt -rmsh -s` writes `shapes.rmsh` next to the shape file; see then draws every particle with its watertight skin instead of the overlapping primitives. `shape2mesh shapes.txt -f ply -e 1e-3` exports one mesh per shape, with exact normals, for Blender, ParaView or MeshLab. Section 8 gives the workflow, and Table 1 maps each section of this note to the code.

1 Introduction

The discrete-element method (DEM) models a granular medium as a set of rigid grains and follows their motion under contact forces and body forces. Grain shape controls packing, dilatancy, force transmission and strength, so its description is a central part of a DEM model, and many descriptions coexist, each with its own balance between geometric accuracy and contact cost [1]: clumps of glued spheres [2], polyhedra with specific contact schemes [3], superquadrics [4], potential particles [5], and level sets sampled on a grid, which can be read directly from tomographic images of real sand [6, 7].

Rockable [8] uses sphero-polyhedra, called R -shapes in the code: a base polyhedron rounded by a sphere of fixed radius R . This description, introduced in DEM by Pournin and Liebling [9, 10] and developed by Alonso-Marroquín, Galindo-Torres and co-workers [11–13], keeps the angular look of polyhedra while giving a well-defined normal everywhere, and it reduces contact detection to distance queries between skeleton features, the basic queries of collision detection [14–16]. The reference article of *Rockable* [8] describes the contact model, the time integration and the parallelisation. The code has been used for rock avalanches and boulder impacts [17–20], the compression of brittle hollow particles [21], and the flow, elasticity and fracture of polyhedral grains [22–24], and some of its features were ported to ExaDEM [25].

This note is about visualisation, where the Minkowski-sum structure of an R -shape had not been used in *Rockable*. An R -shape is stored only as a skeleton and a radius, so its surface is never built explicitly. The viewers (see, *shapeSurvey*) approximate it with a set of overlapping primitives: one sphere per vertex,

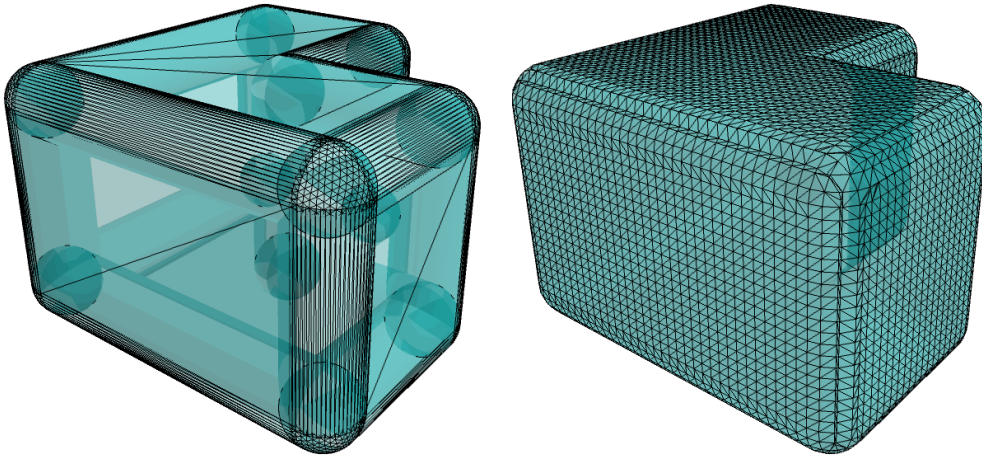


Figure 1: The same R -shape (an L-prism) shown semi-transparently in the Rockable viewer, from one camera. *Left:* the overlapping primitives used for display: a sphere per vertex, a tube per edge and two offset polygons per face. When they are opaque, a depth buffer hides their interiors and they look like a solid. With transparency, every internal sphere, tube and face becomes visible, and the surfaces that coincide are blended several times. *Right:* the single watertight skin mesh described in this note, which is one consistent translucent solid. Both panels are rendered by the same viewer.

one tube per edge, and two offset polygons per face. This set is not a single closed surface but a group of overlapping solids. In an opaque image a depth buffer hides this; with transparency the separate primitives become visible (Figures 1 and 2), and the union can neither be lit with shadows nor exported to a standard mesh format.

What is missing is an explicit, watertight surface. The Minkowski sum gives it directly: the surface of an R -shape is the offset surface of its skeleton at distance R , a classical object of solid modelling [26], and its signed distance is the signed distance to the skeleton minus R . None of the ingredients used below is new; they are recalled in Section 2 with their references. The purpose of this note is to document how they are combined in Rockable, which choices were made and why, how the result was validated, and where its limits are.

The note is organised as follows. Section 2 reviews the related work. Section 3 recalls the R -shape model of Rockable. Section 4 gives the signed distance function and its sign rules (class `ShapeSDF`), Section 5 the mesher (`buildShapeMesh`), Section 6 the coplanar merge (`simplifyCoplanar`), and Section 7 the mass properties by the divergence theorem. Section 8 describes the tools and files seen by the user, Section 9 the validation, and Section 10 stress tests on difficult skeletons, with the cases where the mesher fails.

2 Related work

Offset surfaces and Minkowski sums The solid obtained by moving a ball of radius R over a set is its *offset*, or Minkowski sum with the ball. Rossignac and Requicha [26] gave the mathematical properties of solid offsetting and noted, among others, that offsetting rounds convex features and keeps concave ones sharp, which is the behaviour described in Section 3. Exact Minkowski sums of polyhedra are combinatorially expensive, which led to approximate, volumetric constructions [27]. Closest to the present tools, Pavić and Kobbelt [28] compute the offset of a triangle mesh as the union of spheres, cylinders and prisms, by evaluating a distance on a volumetric grid and extracting its iso-surface; their unsigned distance tolerates non-manifold input, an issue also met below. For convex bodies, Steiner’s formula gives the volume of the offset as a polynomial in R [29], which provides closed-form reference volumes for convex R -shapes.

Distance fields Signed distance fields are a standard representation of shape in graphics; the survey of Jones et al. [30] covers their computation and use, and adaptive sampling [31] reduces their storage. Computing the sign of the distance to a triangle mesh is the delicate part. The angle-weighted pseudonormal of Bærentzen and Aanæs [32] gives an exact sign for a closed manifold; ray casting with a parity count [33, 34] tolerates

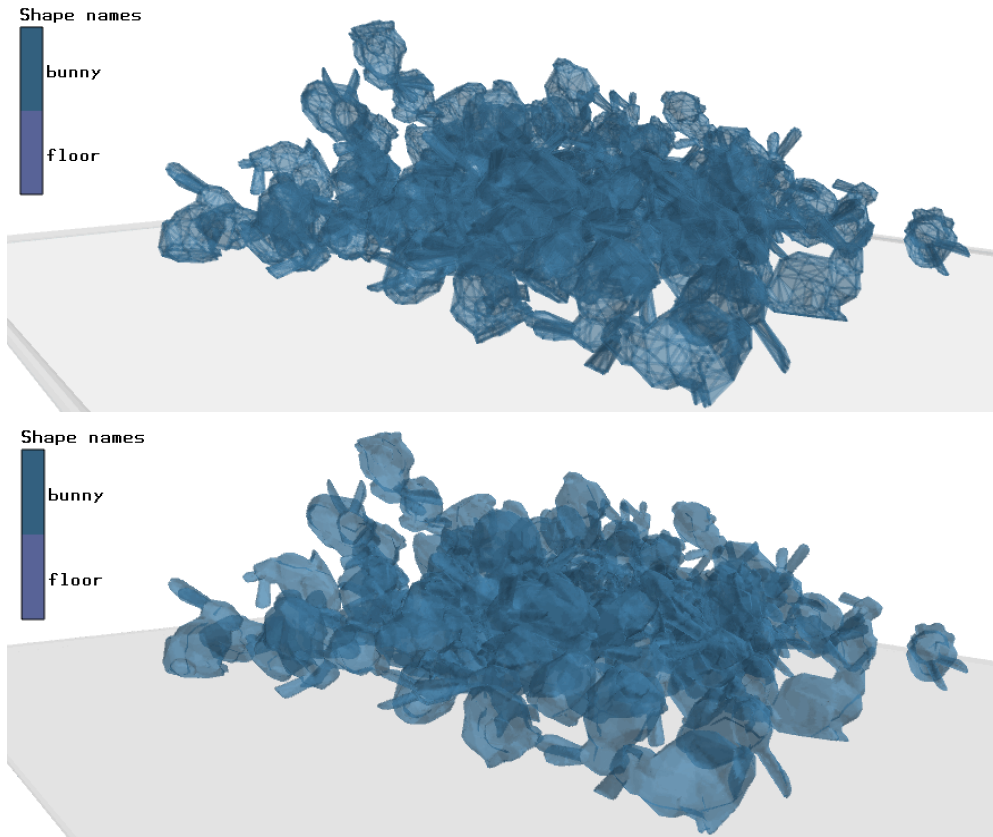


Figure 2: About one hundred Stanford bunnies falling to form a heap, shown semi-transparently in see. Top: the overlapping-primitive drawing; the seams and the internal facets of the primitives fill the image. Bottom: the same scene drawn with the skin meshes read from the `.rmsh` companion file (Section 8).

small defects; the generalized winding number [35] is robust to open and self-intersecting meshes. An R -shape is a case where the distance is *exact* and cheap, because it reduces to point-primitive queries on the skeleton [15, 16], accelerated by a bounding-volume hierarchy [36]. Such a function can also be rendered without any mesh by sphere tracing [37]; a mesh was preferred here because it is reused for export and for the mass properties.

Polygonization of implicit surfaces Marching cubes [38] and its many variants [39], including the treatment of ambiguous cases [40, 41], and earlier polygonizers of implicit surfaces [42], extract a triangle mesh from a sampled scalar field. Projecting the vertices onto the exact iso-surface with the gradient is a common refinement when the function is known analytically. Marching cubes rounds sharp creases; the extended marching cubes of Kobbelt et al. [43] and dual contouring [44] recover them from Hermite data (positions and normals), which the present mesher already stores. A general reference on these operations is the book of Botsch et al. [45].

Mesh simplification Merging the coplanar triangles produced by an iso-surface extractor and re-triangulating the merged polygons was proposed by Hinker and Hansen [46]; superfaces [47] grow nearly planar regions under a bounded error, and vertex decimation [48] removes vertices and re-triangulates the hole. More general methods, such as edge collapse with a quadric error metric [49], trade a controlled error for a target size. The re-triangulation uses ear clipping [33, 50], whose robust implementation on degenerate polygons is discussed by Held [51].

Mass properties The volume, first and second moments of a polyhedron follow from the divergence theorem as sums over its faces [52, 53]. For a triangle mesh, the sums reduce to closed-form tetrahedron integrals [54, 55]. The alternative used so far in Rockable is quasi-Monte-Carlo integration with a Sobol

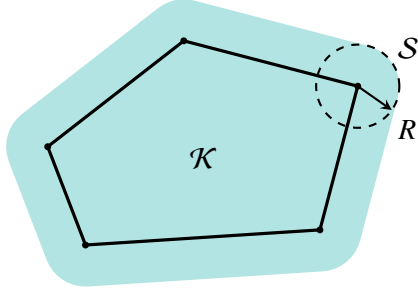


Figure 3: Construction of a sphero-polygon (2D). The skeleton $\mathcal{K}$ (bold) is dilated by a disk of radius R (dashed) into the solid $\mathcal{S}$ (shaded).

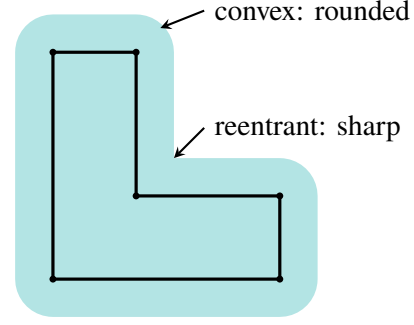


Figure 4: Convex versus reentrant features (2D). Convex corners are rounded with radius R ; at the reentrant corner the dilation fills the notch and leaves a sharp crease.

sequence [56, 57]; its convergence is slowed by the discontinuity of the integrand at the surface [58], an effect measured in Section 7.

What is specific here The tools documented below are a direct assembly of these methods. What the R -shape model brings is that the distance is exact everywhere and its smallest radius of curvature is known (R), so that a single tolerance fixes the grid step, the offset faces are exact planes that can be merged without error, and the mesh depends only on the shape, so it is computed once and shared by all the particles that use it.

3 Sphero-polyhedral shapes in Rockable

An R -shape is defined by a *skeleton* and a rounding radius. In a Rockable shape file, each shape is a block that gives its name, its radius R , and the skeleton in the body frame: `nv` vertices, `ne` edges (pairs of vertex indices) and `nf` faces (planar polygons); the keyword `isSurface` declares that the faces form an open surface rather than the boundary of a solid. In memory, this is the class `Shape` (`src/Core/Shape.hpp`). The R -shape is the set

$$\mathcal{S} = \{ \mathbf{x} : \zeta(\mathbf{x}) \text{dist}(\mathbf{x}, \mathcal{K}) \leq R \}, \quad (1)$$

where the sign $\zeta(\mathbf{x})$ is -1 when $\mathbf{x}$ lies in a volume enclosed by the faces of the skeleton and $+1$ everywhere else. Every point within a distance R of the skeleton belongs to the shape, and so does every point of the volume enclosed by its faces. Geometrically, $\mathcal{S}$ is the Minkowski sum of the filled skeleton and the ball $\mathcal{B}(R)$. A skeleton that encloses no volume (a point, a segment, an open plate) has $\zeta \equiv +1$, and the shape is a ball, a capsule or a slab.

The figures of this section are drawn in two dimensions, where the skeleton has only vertices and edges and the R -shape is a *sphero-polygon* [11] (Figure 3). The 3D case adds planar faces, whose offset gives flat surface patches.

The distance from a point $\mathbf{x}$ to the skeleton is the smallest distance to any primitive,

$$\text{dist}(\mathbf{x}, \mathcal{K}) = \min_{k \in \mathcal{K}} \text{dist}(\mathbf{x}, k). \quad (2)$$

The distance to a vertex is a point distance, to an edge a segment distance, and to a face the distance to a planar polygon: the perpendicular distance if the foot of the perpendicular lies inside the polygon, the distance to its boundary otherwise [16].

The boundary of a convex R -shape is made of three kinds of patch: each face gives a flat patch, parallel to the face and moved out by R ; each convex edge a cylindrical patch of radius R over the outer dihedral angle; each convex vertex a spherical patch of radius R . For such a shape, Steiner's formula [29] gives the volume in closed form,

$$V(\mathcal{S}) = V(\mathcal{K}) + A(\mathcal{K}) R + \frac{R^2}{2} \sum_e \ell_e (\pi - \theta_e) + \frac{4}{3} \pi R^3, \quad (3)$$

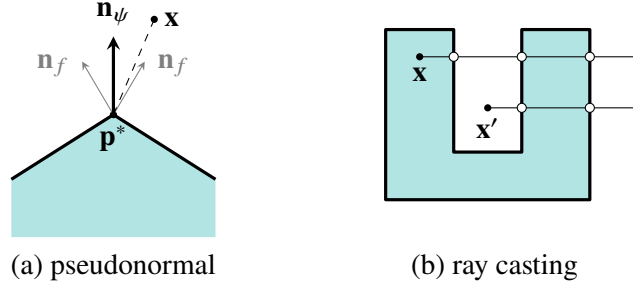


Figure 5: Determining the sign (2D). (a) For a clean closed surface, the sign of $\mathbf{x}$ is the sign of $(\mathbf{x} - \mathbf{p}^*) \cdot \mathbf{n}_\psi$, with $\mathbf{n}_\psi$ the angle-weighted pseudonormal at the closest point [32]. (b) For an imperfect but nearly closed surface, a ray from an inside point ($\mathbf{x}$) crosses the boundary an odd number of times, and from an outside point ($\mathbf{x}'$, in the concavity) an even number of times.

with $A(\mathcal{K})$ the area of the faces, ℓ_e the length of edge e and θ_e its interior dihedral angle. Only convex parts are rounded: at a *reentrant* edge or vertex, the dilation fills the notch and the surface keeps a sharp crease (Figure 4) [26], and Equation (3) no longer holds. This is where a numerical method is needed, and the crease also matters for meshing (Section 5).

Finally, the dimension of the skeleton sets the type of R -shape: a single vertex gives a sphere, vertices and edges with no face give a union of capsules, faces that enclose a volume give the usual sphero-polyhedron, and faces that form an open surface give a slab of thickness $2R$. All these cases come from the same Equation (1); only the sign ζ (Section 4.1) distinguishes a filled interior from a shell.

4 Signed distance function (ShapeSDF)

With Equation (1), an R -shape is an implicit solid. Its signed distance function (SDF) is

$$f(\mathbf{x}) = s(\mathbf{x}) - R, \quad s(\mathbf{x}) = \zeta(\mathbf{x}) \text{dist}(\mathbf{x}, \mathcal{K}), \quad (4)$$

where s is the signed distance to the skeleton. Unlike the overlapping primitives used for rendering, Equation (4) is exact everywhere in space, and $\|\nabla f\| = 1$ wherever f is differentiable. The surface of the R -shape is the zero level set $\{f = 0\}$. Since s is the distance to a fixed set, its gradient points from the closest skeleton point $\mathbf{p}^*(\mathbf{x})$ towards $\mathbf{x}$,

$$\nabla f(\mathbf{x}) = \zeta(\mathbf{x}) \frac{\mathbf{x} - \mathbf{p}^*(\mathbf{x})}{\|\mathbf{x} - \mathbf{p}^*(\mathbf{x})\|}. \quad (5)$$

The exact outward normal at a surface point is thus given by the same closest-point query that gives the distance. The gradient is undefined only on the skeleton itself, which lies at a distance R inside the surface when $R > 0$. Since f is a true distance, the Newton step

$$\mathbf{x} \leftarrow \mathbf{x} - f(\mathbf{x}) \nabla f(\mathbf{x}) \quad (6)$$

moves a point almost onto the surface in one iteration; it is used to project the mesh vertices.

In the code The class ShapeSDF (src/Core/ShapeSDF.hpp) is built once from a Shape. Its methods `closestPoint(x)` (distance, closest point, and type and index of the primitive), `value(x)` (Equation 4), `gradient(x)` (Equation 5) and `inside(x)` answer the queries. The closest-point query is the minimum of Equation (2); computed directly it costs $O(|\mathcal{K}|)$, so it goes through an axis-aligned bounding-box tree built by recursive median split [15, 36]. Two trees are built: one over all primitives (closest point) and one over the faces only (sign).

4.1 Determining the sign

The magnitude of the distance is easy to compute, but its sign is not, because a skeleton read from a file is not always a clean closed surface. The constructor of ShapeSDF classifies the faces and chooses one of three rules (Figure 5).

Algorithm 1 Skin meshing of an R -shape (`buildShapeMesh`)

```

1: build  $f$  and its bounding-volume hierarchy from  $(\mathcal{K}, R)$  ▷ ShapeSDF
2:  $h \leftarrow \sqrt{8} \epsilon R$ ; if not solid:  $h \leftarrow \min(h, R/2)$ ; clamp to maxCells
3: sample  $f$  on a grid of step  $h$  over the dilated AABB ▷ OpenMP
4: for each grid cell do
5:   marching cubes  $\rightarrow$  triangles with vertices on grid edges
6:   for each new edge vertex  $\mathbf{p}$  (one per grid edge) do
7:     repeat newtonIters times:  $\mathbf{p} \leftarrow \mathbf{p} - f(\mathbf{p}) \nabla f(\mathbf{p})$ 
8:     store the normal  $\nabla f(\mathbf{p})$  and the generating primitive
9:   end for
10: end for
11: return the watertight triangle mesh

```

Closed manifold When the faces form a closed, orientable two-manifold, the sign is read from the *angle-weighted pseudonormal* at the closest point [32]: the face normal inside a face, the sum of the two adjacent face normals on an edge, and at a vertex $\mathbf{v}$ the sum of the incident face normals weighted by their angle α_f at $\mathbf{v}$,

$$\mathbf{n}_\psi = \sum_{f \ni \mathbf{v}} \alpha_f \mathbf{n}_f, \quad \zeta(\mathbf{x}) = \text{sign}((\mathbf{x} - \mathbf{p}^*) \cdot \mathbf{n}_\psi). \quad (7)$$

The faces of a shape file are not guaranteed to be consistently oriented. They are therefore oriented by a walk through the face-adjacency graph (a face is flipped when it disagrees with a neighbour on the direction of their shared edge), and the global orientation is fixed by requiring that the enclosed volume,

$$V = \frac{1}{6} \sum_f \sum_i \mathbf{v}_0 \cdot (\mathbf{v}_i \times \mathbf{v}_{i+1}), \quad (8)$$

computed on the fan triangulation $(\mathbf{v}_0, \mathbf{v}_i, \mathbf{v}_{i+1})$ of each face, be positive. This volume of the skeleton is available as `ShapeSDF::skeletonVolume()`, and `isClosedVolume()` reports that this rule is used.

Imperfect face soup Meshes imported from other formats (for instance with the converter `stl2shape`) are often not clean manifolds: a few duplicated edges, T -junctions or a small hole, while still clearly bounding a solid. For such faces the sign is obtained by *ray casting*: it is negative when a ray from $\mathbf{x}$ along $+x$ crosses the faces an odd number of times [33], each crossing being a ray–triangle intersection [34]. This parity rule holds for any concave solid, including one with internal cavities, and it is the point-in-polyhedron test of `Shape::inside()` used for contact detection [8], so the two predicates agree. Ray casting is chosen when the fraction of *boundary edges* (edges used by a single face) is small, below 10%: a nearly closed STL has few, a real open sheet is made almost only of them. The generalized winding number [35] would need no threshold; ray casting was kept for consistency with `Shape::inside()`.

Open surface When the skeleton has no face (a sphere, a capsule), or when its faces are declared as a surface (`isSurface`), there is no interior and $\zeta \equiv +1$. The zero level set of $f = \text{dist}(\mathbf{x}, \mathcal{K}) - R$ is then the closed surface at distance R from the skeleton: a ball, a capsule or a slab of thickness $2R$. The method `ShapeSDF::isSolid()` is false in this case, which the mesher uses to resolve the thickness (Section 5).

5 Surface meshing (`buildShapeMesh`)

The surface mesh is the zero level set of Equation (4), extracted with marching cubes [38] on a regular sampling of f , each vertex being then moved onto the exact surface with Equation (6) (Algorithm 1). The function is `buildShapeMesh(shape, options)` (`src/Core/ShapeMesh.cpp`); it returns a `ShapeMesh` holding the positions $\mathbf{P}$, the exact normals $\mathbf{N}$, the generating primitive of each vertex (`primType`, `primIndex`), the triangles `tri`, and the sag, volume and area of the mesh.

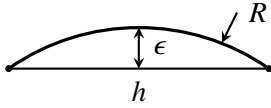


Figure 6: Resolution control (2D). Across a rounded feature of radius R , a mesh edge of length h (chord) leaves a sag $\epsilon \approx h^2/(8R)$ from the true surface (arc). A target sag ϵ therefore sets the grid step $h \approx \sqrt{8\epsilon R}$.

Sampling and extraction f is evaluated at the nodes of a regular grid covering the bounding box of the shape enlarged by R , with `marginCells` (two) cells of padding. Marching cubes creates the triangles given by the sign pattern of the eight corners of each cell, with the lookup table of the original algorithm. The ambiguous configurations of this table [39–41] are not specially treated: every mesh produced in the tests was checked to be watertight (Section 9), and the only topological failures observed (Section 10) come from features narrower than the grid step, not from ambiguous cells. Each vertex is placed on the crossed cell edge by linear interpolation and receives a key from this grid edge, so it is created once even though the edge is shared by up to four cells. This makes the mesh watertight by construction: each mesh edge belongs to exactly two triangles.

Projection and exact normals The interpolated vertex is accurate where f is linear along the cell edge, not near rounded features. Newton steps, Equation (6), move it onto $\{f = 0\}$. Since f is a true distance, each step is a full projection and convergence is quadratic, so a fixed number of steps (`newtonIters`, two by default) is enough to reach rounding error, with no stopping test to tune. For each vertex, the exact normal ∇f and the generating skeleton primitive (vertex, edge or face) are stored; they are used for shading, colouring, debugging and simplification. After projection, the only remaining error is the chordal error between vertices.

Resolution and tolerance A single tolerance controls this error: the target *chordal error* ϵ , or *sag* (from *sagitta*), the largest gap between a mesh edge and the true surface (Figure 6). It is the option `epsilon` of `ShapeMeshOptions`, in the length unit of the shape file; when it is zero, $\epsilon = \text{epsilonRel} \times R$ with `epsilonRel` = 0.01. Across a rounded feature of radius R , a chord of length h gives $\epsilon \approx h^2/(8R)$, hence $h \approx \sqrt{8\epsilon R}$. Since R is the smallest radius of curvature of an R -shape, this step resolves every rounded edge and corner with the same sag, and flat faces need no finer step. The mesh is an inscribed polyhedron, so its volume converges to that of the smooth solid as $O(h^2)$, that is, linearly in ϵ .

Mesh size and cost The surface has about one triangle per area h^2 , so the number of triangles scales as $1/(\epsilon R)$. The computing time, however, is dominated by the $O((L/h)^3)$ grid nodes for a shape of size L , while only $O((L/h)^2)$ cells produce triangles. Hence the need for fast queries and for a limit on the grid size.

Thin features and budget A shape that is not a filled solid (`isSolid()` false: sphere, capsule, plate, open surface) has a size of order R or is a slab of thickness $2R$; its step is limited to $h \leq R/2$ so that the grid does not step over it. A filled solid needs no such limit. In addition, `maxCells` (2×10^7 grid nodes by default) protects against a large thin shape, such as a wall: when the requested step would exceed it, h is increased to fit and a message is printed.

Acceleration Both queries (closest point, and ray where needed) use the bounding-volume hierarchies of `ShapeSDF`, which reduce the search of Equation (2) from $O(|\mathcal{K}|)$ to about $O(\log |\mathcal{K}|)$. The grid sampling runs on several threads with OpenMP [59] when the code is compiled with it.

One mesh per shape The mesh lives in the body frame and depends only on the skeleton and the radius, not on the position or orientation of a particle. Since `Shape::homothety()` scales both the vertices and R , two homothetic shapes are similar and one mesh plus a scale factor describes both. A sample of thousands of grains built from a few shapes needs only a few meshes, which is what makes an explicit surface affordable for a whole scene.

Sharp reentrant edges The Minkowski sum keeps reentrant edges sharp (Section 3), but marching cubes on a uniform grid rounds them over a length of about h . This small visual effect is accepted. The positions and normals already stored are the Hermite data used by extended marching cubes [43] or dual contouring [44], which would recover the crease.

6 Coplanar simplification (`simplifyCoplanar`)

Marching cubes cuts a flat region into many small coplanar triangles. On an R -shape, the flat regions are the offset faces, which are exact planes, so their triangulation can be made coarser without moving the surface. The function `simplifyCoplanar(mesh, options)` (`src/Core/ShapeMeshSimplify.cpp`) does this, in the spirit of the coplanar merge of Hinker and Hansen [46] and of superfaces [47]: it removes the interior vertices of planar regions and keeps their boundary unchanged. A general decimation [48, 49] would also coarsen these regions, but it moves vertices and balances an error against a size; on an exactly planar region the merge removes triangles with no geometric error.

Region growing A region starts from a seed triangle and adds a neighbour when its normal agrees with the seed normal within `angleToleranceDeg` (0.25° by default) and each of its vertices lies within a distance tolerance of the seed plane. The seed plane is fixed for the whole region, so a chain of slightly tilted triangles cannot drift away from it. On the offset faces both tests pass up to rounding, while curved patches are never merged because neighbouring normals across a rounded edge differ by much more than the angle tolerance.

The distance tolerance follows the mesh This tolerance bounds how much the surface can move when an interior vertex is removed. It is `planeToleranceSag` times the sag of the mesh (one twentieth by default), so the error of the merge is always a small fraction of the mesh error and decreases with it. A tolerance tied to the bounding box, a natural first choice that was the first implementation, stays fixed while the mesh error decreases and ends up being the largest error (Section 9). `planeToleranceRel` is used only for a mesh that carries no sag.

Re-triangulation The boundary edges of a region, connected end to end, form an oriented loop, which is triangulated in the region plane by ear clipping [33, 50]. Every boundary vertex must be kept: the neighbouring curved regions are not modified and keep every vertex of the shared boundary, and removing one would create a T -junction and break watertightness. A flat face with n boundary vertices then goes from $O(n^2)$ grid triangles to $n - 2$ triangles. Keeping every boundary vertex makes the loop hard to triangulate, because marching-cubes boundaries have long collinear runs, a known difficulty of ear clipping on degenerate polygons [51]. Three measures make it robust: the point-in-triangle test is *strict*, so a vertex lying on an edge of a candidate ear does not block it; among the valid ears the largest is clipped first, so collinear points become the base of well-shaped triangles; and every clipped triangle is checked to be empty. If no safe ear is left, or if the boundary is not one simple loop (a face with a hole), the region keeps its original triangles. Without these measures, the concave face of an L-shaped prism (Figure 4) was closed by a triangle across its notch.

Properties The merge does not move or add any vertex and keeps every boundary edge, so the mesh stays a closed two-manifold with the same Euler characteristic, and a concave face is handled correctly. On an exactly planar region it preserves the volume exactly; elsewhere, a thin strip of the neighbouring rounded part may pass as flat, and the change is bounded by the distance tolerance. The saving depends on how flat the surface is (Figure 7): large for a rounded polyhedron, none for a sphere.

7 Mass properties by the divergence theorem

A DEM code needs the volume, centre of mass and inertia tensor of each grain. In Rockable, `Shape::massProperties()` estimates them by quasi-Monte-Carlo integration over the bounding box: the

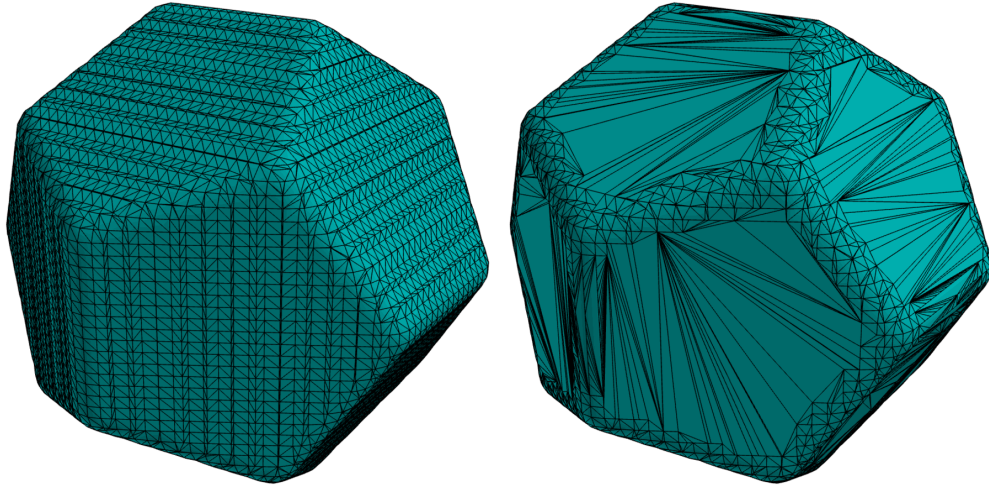


Figure 7: Coplanar simplification of a rounded polyhedron, drawn by the Rockable viewer. Left: the marching-cubes mesh (15 500 triangles). Right: after `simplifyCoplanar` (5 840 triangles); the flat offset faces are re-triangulated from their boundary, the rounded edges and corners keep their resolution, and the mesh stays watertight.

points come from a Sobol sequence [56, 57], their number is set by the keyword `MCnstep` of the shape file (10^4 by default, clamped to $[10^3, 10^7]$), and each point is tested with `Shape::inside()`. The results are written back to the shape file (`volume`, `I/m`, `preCompDone`); `shapeSurvey -P` runs this computation without a window. The estimate is cheap to write but its error decreases slowly and irregularly with the number of samples, and the inertia tensor remains noisy. The watertight mesh gives a deterministic alternative.

With the divergence theorem [52, 53], any volume integral over the solid becomes a sum over the triangles. With $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ the vertices of triangle t , $d_t = \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$ and $\mathbf{s} = \mathbf{a} + \mathbf{b} + \mathbf{c}$,

$$V = \frac{1}{6} \sum_t d_t, \quad V\mathbf{C}_g = \frac{1}{24} \sum_t d_t \mathbf{s}, \quad \int_S \mathbf{xx}^T dV = \frac{1}{120} \sum_t d_t (\mathbf{aa}^T + \mathbf{bb}^T + \mathbf{cc}^T + \mathbf{ss}^T), \quad (9)$$

which are the exact integrals over the tetrahedra joining the origin to each triangle [54, 55]. The inertia tensor about $\mathbf{C}_g$ follows from the second-moment matrix with the parallel-axis theorem. These values are exact for the meshed polyhedron and converge to those of the smooth solid as $O(h^2)$, linearly in ϵ .

State of the code In the library, the volume and the area are computed for every mesh by `ShapeMesh::computeVolume()` and `computeArea()`, and stored in the fields `volume` and `area`. The centre of mass and the inertia of Equation (9) are, at the time of writing, only implemented in the benchmark program used for Figure 8; `Shape::massProperties()` still uses the Sobol estimate. Moving these sums into the library, as an option of the pre-computation, is the natural next step.

Comparison Figure 8 compares the two methods on a capsule, the concave L-prism and the Stanford bunny, all with $R = 2$ mm, as the relative error versus computing time (Apple M4 laptop, 10 cores, GCC 16 with optimisation and OpenMP). A single Sobol run gives one fixed error that does not decrease monotonically with N ; to measure the *typical* error, the sequence is randomised with a Cranley–Patterson shift [60], and the mean over independent replicates is plotted with their spread. The capsule has a closed-form reference; for the L-prism and the bunny the reference is a Richardson extrapolation [61] of the two finest meshes, $V^* \approx V_f + (V_f - V_c)/((h_c/h_f)^2 - 1)$, done also for each component of the centre of mass and inertia.

The Monte-Carlo error decreases roughly as $N^{-2/3}$. A heuristic argument explains this: the integrand is an indicator function, so the error comes from the cells of side $N^{-1/3}$ crossed by the surface, about $N^{2/3}$ of them, each contributing an error of order $1/N$ with independent signs after the random shift, hence $\sqrt{N^{2/3}}/N = N^{-2/3}$. Rigorous rates for randomised quasi-Monte-Carlo on discontinuous integrands are given by He and Wang [58]. One more decade of accuracy thus costs about three decades of samples, whereas the divergence-theorem error decreases as the inverse of the number of triangles. On the bunny, where each

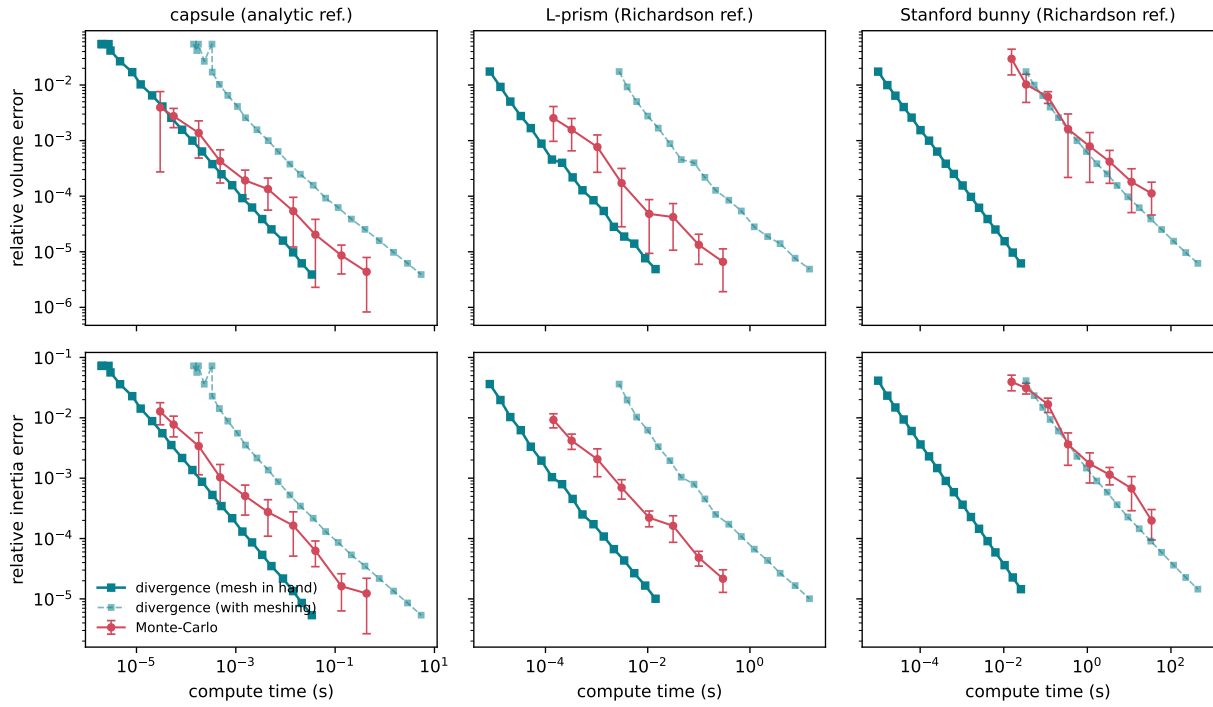


Figure 8: Mass properties by quasi-Monte-Carlo (the Sobol estimator of `Shape::massProperties()`) versus the divergence theorem on the skin mesh, for three R -shapes at $R = 2$ mm: relative errors of the volume (top) and of the inertia tensor (bottom, Frobenius norm) versus computing time. Monte-Carlo (red): mean over randomised-Sobol replicates, with their spread. Divergence theorem: solid line with the mesh already built, dashed line including the mesh build. Reference: analytic for the capsule, Richardson extrapolation otherwise; points within three times the uncertainty of the reference (1.2×10^{-6} for the L-prism, 3.3×10^{-7} for the bunny) are not plotted. The reentrant crease of the L-prism adds an error term not cancelled by the pure h^2 extrapolation.

inclusion test casts a ray against hundreds of faces, the best Monte-Carlo point reaches 1.1×10^{-4} in 34 s, while the triangle sum reaches it in 1.6 ms once the mesh exists, and in 9 s including the mesh build.

For the inertia tensor (error measured with the Frobenius norm), the mesh values keep the second order (measured order 2.0 on the three shapes) and the error is 1.4 to 2.3 times the volume error at the finest mesh, whereas the Monte-Carlo inertia error is two to three times its volume error, since the second moments weight each sample by a squared distance. At the finest meshes (four to ten million triangles, a few tens of milliseconds per sum), the relative volume errors are 3.9×10^{-6} (capsule), 4.9×10^{-6} (L-prism) and 6.2×10^{-6} (bunny), and the centre of mass is within 3×10^{-9} , 2×10^{-7} and 6×10^{-7} of the reference, in units of the shape size.

Refinement as a diagnostic The convergence is also a test of the *input*. As downloaded, the bunny skeleton has two vertices 7.4×10^{-5} m apart (3.7% of R) that were never merged. This pair creates a degenerate face and two dangling edges, so the faces are not a closed manifold and the sign falls back to ray casting; a ray leaving through the degenerate face counts one crossing too many, so a region of fixed size gets the wrong sign. Figure 9 shows that the two series agree down to a relative error of about 10^{-4} , after which the imported skeleton stops converging while the repaired one keeps decreasing. Such a defect is invisible in a viewer and in a coarse mass computation. It is revealed by meshing at two resolutions and comparing the volumes, and the choice of the sign rule already flags it: a closed-looking shape that falls back to ray casting has faces that are not a clean manifold and deserves a check.

Oriented bounding box The mesh could also serve the oriented bounding box used for neighbour detection. Rockable fits its orientation on the skeleton vertices only (option `fib0bbOption`) and enlarges the box by R ; a skeleton with few vertices (a capsule has two) gives little information, whereas the mesh vertices sample

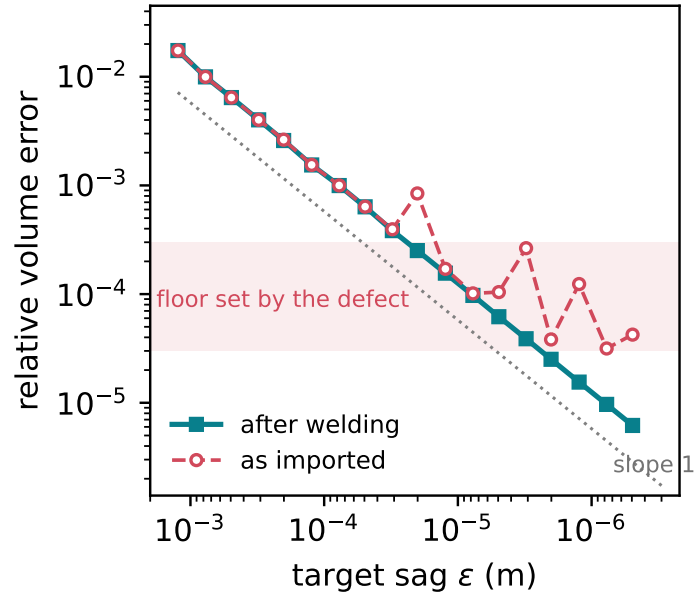


Figure 9: Effect of one duplicate vertex: relative error of the skin-mesh volume of the Stanford bunny versus the target sag, for the file as imported (circles) and after merging its duplicate vertex (squares). Below about 10^{-4} , the imported skeleton stops converging (shaded band), while the repaired one keeps decreasing at order one in the sag (dotted line).

the real surface evenly. This is left for future work.

8 Tools and files

The tools of Sections 4–7 belong to the C++ core library of Rockable (target `Rockable_core`), distributed under the CeCILL-B licence, with no external dependency beyond the optional OpenMP already used by the DEM code. All functions work in the body frame and do not modify the shape. Table 1 maps the sections of this note to the code.

Command-line mesher `shape2mesh` applies these functions to a shape file:

```
shape2mesh shapes.txt -f ply -e 1e-3 -s      # one PLY per shape, sag 1e-3, merged
shape2mesh shapes.txt --rmsh -s              # all shapes -> shapes.rmsh, for the viewer
```

The options are `-f obj|ply` (format), `-e` (sag, in the length unit of the shape file; default 1% of R), `-s [deg]` (coplanar merge, with an optional angle tolerance), `-r/--rmsh` (companion file) and `-o` (output directory). OBJ and PLY files carry the exact normals as vertex normals and can be opened in Blender, MeshLab or ParaView. A file with one shape gives `<stem>.<ext>`, a file with several shapes `<stem>_<shapeName>.<ext>`. In `-rmsh` mode without `-e`, each shape gets its own sag, 0.4% of the diagonal of its bounding box, so that a large wall and a small grain in the same file both get a suitable mesh.

Companion mesh file A `.rmsh` file stores the skin mesh of each shape as one block named after the shape (positions, exact normals, triangles). When `see` opens a configuration, it looks for a file with the base name of the shape file and the extension `.rmsh`; the mesh of each shape found there is compiled once into an OpenGL display list and drawn for every particle of this shape with its position, orientation and homothety applied. Shapes without a block, and soft particles, are drawn with the overlapping primitives. The simulation files are not modified, and the meshing cost is paid once, before visualisation.

Table 1: Where each part of this note lives in the code.

Section	Code	File
3 R -shape	<code>Shape, Shape::inside()</code>	<code>src/Core/Shape.hpp</code>
4 SDF and sign	<code>ShapeSDF: closestPoint, value, gradient, inside, isClosedVolume, isSolid, skeletonVolume</code>	<code>src/Core/ShapeSDF.hpp/.cpp</code>
5 Meshing	<code>buildShapeMesh, ShapeMeshOptions (epsilon, epsilonRel, newtonIters, marginCells, maxCells), ShapeMesh</code>	<code>src/Core/ShapeMesh.hpp/.cpp</code>
6 Merge	<code>simplifyCoplanar, SimplifyOptions (angleToleranceDeg, planeToleranceSag, planeToleranceRel)</code>	<code>src/Core/ShapeMeshSimplify.cpp</code>
7 Mass	<code>ShapeMesh::computeVolume, computeArea; Monte-Carlo: Shape::massProperties, keyword MCnstep</code>	<code>src/Core/ShapeMesh.cpp, src/Core/Shape.cpp</code>
8 Export	<code>writeOBJ, writePLY, writeRmsh, loadRmsh</code>	<code>src/Core/ShapeMesh.cpp</code>
8 Mesher	<code>shape2mesh</code>	<code>prepro/converters/shape2mesh/</code>
8 Viewers	<code>see (reads <stem>.rmsh), shapeSurvey -skin</code>	<code>src/Apps/see.cpp, prepro/shapesurvey/</code>
9 Tests	<code>test_ShapeSDF, test_ShapeMesh (ctest -R Shape)</code>	<code>test/ShapeSDF/</code>

Shape viewer `shapeSurvey` draws a skin with `-skin file.rmsh`, and has a screenshot mode (`-o file.png`, with `-elev`, `-azim`, `-A` for the opacity, `-wire`, `-radius` to override R) that renders a single frame and exits. All renderings of this note were produced this way.

Typical workflow

1. Build or import the shapes (`stl2shape` for an STL), check them in `shapeSurvey`, and pre-compute them once with `shapeSurvey shapes.txt -P shapes.txt`. A shape still marked `preCompDone` is otherwise pre-computed by `shape2mesh` itself at each run, with the Monte-Carlo cost of Section 7, because the mesh is built in the body frame (mass centre and principal axes).
2. Run `shape2mesh shapes.txt -rmsh -s` next to the shape file.
3. Open the configuration in `see`: the console reports “Loaded n skin mesh(es) from `shapes.rmsh`”.
4. Re-run step 2 whenever a shape changes: the `.rmsh` file is a cache and is not updated automatically.

Choosing the sag The default sags are meant for display. For export or for mass properties, a sag of $10^{-3}R$ to $10^{-2}R$ is typical; halving it roughly doubles the number of triangles. If `shape2mesh` reports that the grid budget was reached, the shape is large compared with R (a wall): increase the sag for this shape, or raise `maxCells` in a program linked to the library.

9 Validation

The two test programs of `test/ShapeSDF` (`test_ShapeSDF` and `test_ShapeMesh`, registered in `CTest` as `ShapeSDF` and `ShapeMesh`) check the properties below, on shapes whose distance function is known in closed form, against the Monte-Carlo volume of `Shape::massProperties()`, and on real shapes, including a model imported from an STL file (Figure 10).

Distance and normals A sphere, a capsule, a rounded box and a plate have hand-written SDFs. Over 2×10^4 random points, `ShapeSDF::value` equals the closed form exactly for the first three and within 2.2×10^{-16} for the plate; $|\|\nabla f\| - 1| \leq 3.3 \times 10^{-16}$ on all four, and the gradient agrees with a centred finite difference to 3×10^{-10} , the accuracy of the finite difference. On closed skeletons, the sign agrees with `Shape::inside()` at every sampled point.

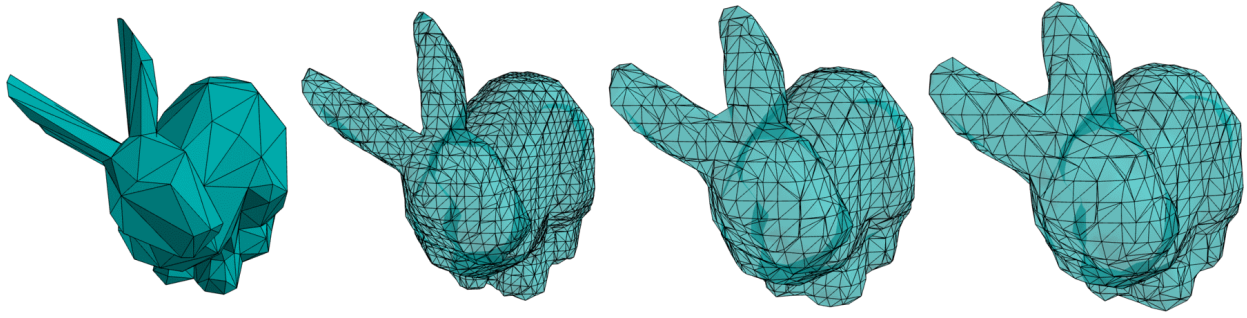


Figure 10: An STL imported as an R -shape (`stl2shape`), drawn by the viewer with the mesh wireframe. The triangles become the skeleton, to which a radius is given. From left to right, the Stanford bunny (about 100 mm long) with $R = 0, 2, 4$ and 5 mm; the panels with $R > 0$ show the skin mesh.

Watertightness and surface accuracy Every mesh is watertight (each edge belongs to exactly two triangles), with Euler characteristic $\chi = 2$ on the sphere, capsule, rounded box, concave L and a rounded polyhedron. After the Newton projection, $\max |f| \approx 5.6 \times 10^{-17}$ at the vertices of the polyhedron, and the stored normals equal the analytic gradient.

Volume convergence Over three grids with h halved at each level, the observed order is 1.96 (sphere), 2.04 (capsule) and 1.99 (rounded box), and 2.39 for the concave L, whose rounded reentrant crease adds an error that decreases faster. The limit is the true volume, as confirmed against the analytic capsule and the Monte-Carlo estimate. This check also revealed that the volume stored in one of the sample shape files did not match its own skeleton and radius.

Coplanar merge After `simplifyCoplanar` the mesh is still watertight with $\chi = 2$, including the concave L. The number of triangles goes from 15 500 to 5 840 (−62%) on the rounded polyhedron and from 13 716 to 6 828 (−50%) on the L; a sphere is unchanged. With the tolerance tied to the sag, the relative change in volume stays between 3×10^{-7} and 1×10^{-6} on the polyhedron for sags from 4×10^{-3} down to 3×10^{-4} , while the saving grows from 52% to 69%. With a tolerance tied to the bounding box, the change grows from 4×10^{-6} to 2×10^{-5} over the same range, and would end up larger than the discretisation error.

Performance For a rock in the shape of a hand imported from an STL file with about 2800 faces, the direct $O(|\mathcal{K}|)$ search at every grid node is impractical, while the hierarchy makes it routine: with OpenMP, a set of four shapes including this model is meshed in about 16 s, and the 309-face bunny in about 0.1 s.

10 Stress tests on very complex shapes

The skeletons accepted by Rockable are not limited to rounded polyhedra: faces may or may not enclose a volume, edges may have no face, a shape may combine the three kinds of primitive, and through-openings or knots are allowed. The mesher was tested on shapes built for this purpose, each with one kind of complexity, and every mesh was checked for watertightness and topology: number of connected components and Euler characteristic χ of each (vertices minus edges plus triangles). A closed surface with no hole has $\chi = 2$, and each through-hole (handle) reduces χ by two: a torus has $\chi = 0$, and $\chi = -8$ means genus 5.

Figure 11 shows these shapes. The *frame* keeps its through-opening after dilation ($\chi = 0$). The *cage* is a union of capsules joined at the eight corners, a genus-5 surface, and the mesh gives exactly $\chi = -8$. The *bowl* gives one watertight shell with a rounded rim ($\chi = 2$). The three slabs of the *book* are joined along the spine into a single watertight surface. The *hybrid* gives the two expected shells (the box with its tube, and the isolated ball), and the *trefoil* a knotted torus with $\chi = 0$.

Thin openings The dilation reduces a hole of width d to $d - 2R$ (Figure 12). For $d = 4R$ the channel is $2R$ wide and the genus is kept; for $d = 1.5R$ the dilation fills the hole, which is the true Minkowski geometry.

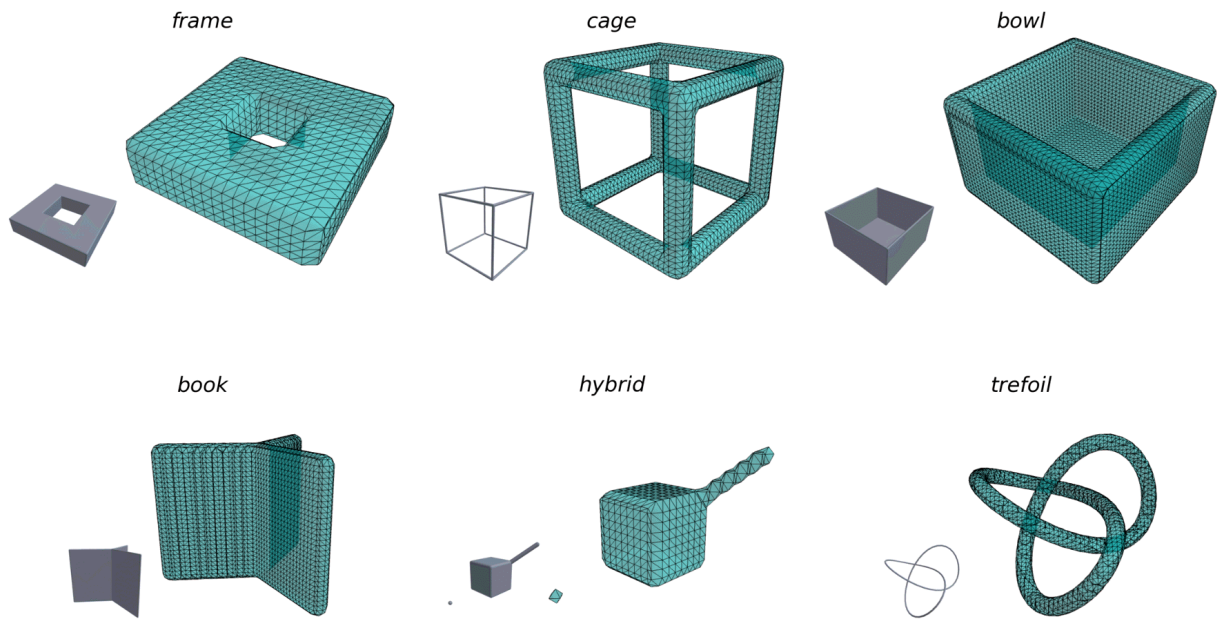


Figure 11: Six complex R -shapes and their skin meshes, drawn semi-transparently with the wireframe; next to each, in grey, its skeleton drawn with a small radius. Top: a *frame* (genus-1 solid); a *cage*, the twelve edges of a cube with no face (genus 5); a *bowl*, a box without its lid declared as a surface (slab of thickness $2R$). Bottom: a *book*, with one spine edge shared by three sheets (non-manifold); a *hybrid*, a filled box with a dangling edge and an isolated vertex (two shells); a *trefoil*, seventy-two capsules forming a knotted torus. Every mesh is watertight with the expected components and Euler characteristic.

For $d = 2.5R$ the channel exists but is only $R/2$ wide, smaller than the default step: no grid point falls inside it, marching cubes closes it, and the mesh is still perfectly watertight, so nothing looks broken although the genus is wrong. A finer sag gives the hole back.

Thin details on a solid The *hybrid* is a filled solid, so the limit $h \leq R/2$ of Section 5 does not apply and its two details of size R are sampled with a step chosen for the box: the isolated ball is reduced to a few triangles and the tube is irregular (Figure 13), with a correct topology. A finer sag solves it.

Lessons Every observed failure is topological; the mesh never has a crack, because shared grid edges always give a watertight surface. Both failures have the same cause, a feature (solid or empty) narrower than the grid step, and the same remedy, a smaller sag. Thin solids are protected automatically, thin openings are left to the user; a step limit based on the narrowest opening of the skeleton would be the natural extension. Finally, the topology check is useful beyond testing: the shape readers do not validate their input, and a shape generator bug that produced an edge pointing to a non-existent vertex was read without warning but immediately detected by the component and Euler-characteristic check of its skin. This check is a cheap test for any imported shape.

11 Summary and perspectives

Because an R -shape is a Minkowski sum, its signed distance is the signed distance to its skeleton minus the rounding radius, an exact function with an analytic gradient. Rockable uses it, through ShapeSDF, to build

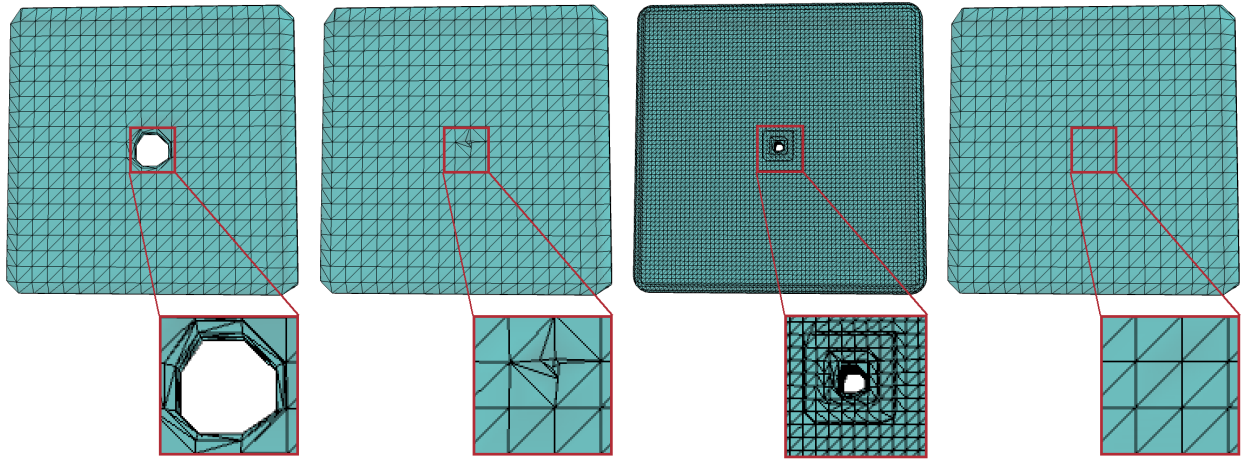


Figure 12: A plate with a square through-hole of side d , seen from above, with the central region magnified (white seen through means the channel is open). From left to right: $d = 4R$, the hole is kept (genus 1); $d = 2.5R$ with the default sag, the remaining channel of width $R/2$ is smaller than the grid step and is closed, a failure invisible in a watertight mesh; the same with a finer sag, the hole is open again; $d = 1.5R < 2R$, the dilation itself fills the hole, and genus 0 is correct.

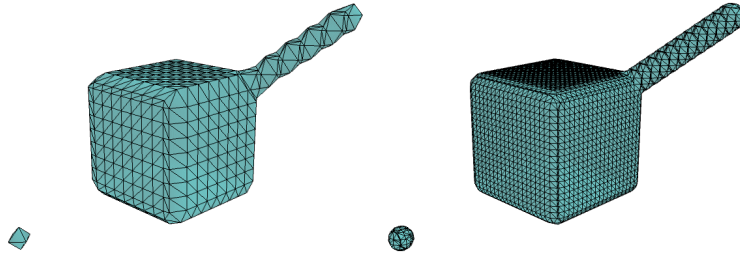


Figure 13: The hybrid at the default sag (left) and at a finer one (right). The shape is a filled solid, so the limit $h \leq R/2$ does not apply, and the isolated ball and the tube are coarse at the default sag, with the correct topology.

a watertight skin mesh by marching cubes and Newton projection (`buildShapeMesh`), controlled by a single sag tolerance and accelerated by a bounding-volume hierarchy; to remove the unneeded triangles of the flat faces (`simplifyCoplanar`); and to obtain the mass properties by the divergence theorem, deterministically and at second order, orders of magnitude faster than the Sobol estimate for a given accuracy once the mesh exists. The sign rule handles clean manifolds, imported meshes with defects, and open surfaces. The meshes are written by `shape2mesh`, either as OBJ/PLY files or as a `.rmsh` companion read by `see` and `shapeSurvey`.

The main purpose is visualisation: an explicit watertight surface can be shaded, made transparent, lit with shadows and exported, which the overlapping primitives cannot. Possible extensions are: computing the centre of mass and inertia from the mesh inside `Shape::massProperties()`; recovering sharp reentrant edges with extended marching cubes or dual contouring [43, 44] from the data already stored; storing several levels of detail per shape for the instanced rendering of large samples; extending the step limit to thin openings; and using Steiner’s formula, Equation (3), as an exact reference for convex shapes. The same steps (marching cubes, projection onto the zero level set, moments by the divergence theorem) also apply to level-set grains [6, 7], with a sampled rather than exact distance.

Note on the preparation The geometric tools described here, the companion mesh file, the viewer support and the benchmark scripts were written with the help of an AI coding assistant (Anthropic’s Claude, through Claude Code), which was also used to draft and edit this note. The Rockable framework, its shape model and its numerical methods predate this work. The author defined the goals, reviewed every change and checked every reported number; several defects found during this review (faces drawn inside a solid, a merge that closed a reentrant notch, a tolerance that did not decrease with the mesh) are described in the text with

their correction. Every image is a rendering by the Rockable viewers or a plot of computed data.

References

- [1] Gang Lu, Jonathan R. Third, and Christoph R. Müller. Discrete element models for non-spherical particle systems: From theoretical developments to applications. *Chemical Engineering Science*, 127:425–465, 2015. doi: 10.1016/j.ces.2014.11.050.
- [2] John F. Favier, Mohammad H. Abbaspour-Fard, Martin Kremmer, and Abdul O. Raji. Shape representation of axi-symmetrical, non-spherical particles in discrete element simulation using multi-element model particles. *Engineering Computations*, 16(4):467–480, 1999. doi: 10.1108/02644409910271894.
- [3] Peter A. Cundall. Formulation of a three-dimensional distinct element model—Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, 25(3):107–116, 1988. doi: 10.1016/0148-9062(88)92293-X.
- [4] Alexander Podlozhnyuk, Stefan Pirker, and Christoph Kloss. Efficient implementation of superquadric particles in Discrete Element Method within an open-source framework. *Computational Particle Mechanics*, 4(1):101–118, 2017. doi: 10.1007/s40571-016-0131-6.
- [5] Guy T. Houlsby. Potential particles: a method for modelling non-circular particles in DEM. *Computers and Geotechnics*, 36(6):953–959, 2009. doi: 10.1016/j.compgeo.2009.03.001.
- [6] Reid Kawamoto, Edward Andò, Gioacchino Viggiani, and José E. Andrade. Level set discrete element method for three-dimensional computations with triaxial case study. *Journal of the Mechanics and Physics of Solids*, 91: 1–13, 2016. doi: 10.1016/j.jmps.2016.02.021.
- [7] Reid Kawamoto, Edward Andò, Gioacchino Viggiani, and José E. Andrade. All you need is shape: predicting shear banding in sand with LS-DEM. *Journal of the Mechanics and Physics of Solids*, 111:375–392, 2018. doi: 10.1016/j.jmps.2017.10.003.
- [8] Vincent Richefeu, Gaël Combe, Lhassan Amarsid, Raphaël Prat, Jean-Mathieu Vanson, Saïed Nezamabadi, Patrick Mutabaruka, Jean-Yves Delenne, and Farhang Radjai. Advanced strategies for discrete simulations with three-dimensional R -shapes in the Rockable framework. *Computer Physics Communications*, 320:109997, 2026. doi: 10.1016/j.cpc.2025.109997.
- [9] Lionel Pournin and Thomas M. Liebling. A generalization of distinct element method to tridimensional particles with complex shapes. In Ramón García-Rojo, Hans J. Herrmann, and Sean McNamara, editors, *Powders and Grains 2005*, volume 2, pages 1375–1378, Leiden, 2005. A.A. Balkema.
- [10] Lionel Pournin and Thomas M. Liebling. From spheres to spheropolyhedra: generalized distinct element methodology and algorithm analysis. In William Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 347–363. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-76796-1_16.
- [11] Fernando Alonso-Marroquín. Spheropolygons: A new method to simulate conservative and dissipative interactions between 2D complex-shaped rigid bodies. *EPL (Europhysics Letters)*, 83(1):14001, 2008. doi: 10.1209/0295-5075/83/14001.
- [12] Fernando Alonso-Marroquín and Yucang Wang. An efficient algorithm for granular dynamics simulations with complex-shaped objects. *Granular Matter*, 11(5):317–329, 2009. doi: 10.1007/s10035-009-0139-1.
- [13] Sergio A. Galindo-Torres and Dorival M. Pedroso. Molecular dynamics simulations of complex-shaped particles using Voronoi-based spheropolyhedra. *Physical Review E*, 81(6):061303, 2010. doi: 10.1103/PhysRevE.81.061303.
- [14] Elmer G. Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988. doi: 10.1109/56.2083.
- [15] Gino van den Bergen. *Collision Detection in Interactive 3D Environments*. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, San Francisco, 2003. ISBN 978-1-55860-801-6.

- [16] Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, San Francisco, 2005. ISBN 978-1-55860-732-3.
- [17] Vincent Richefeu, Guilhem Mollon, Dominique Daudon, and Pascal Villard. Dissipative contacts and realistic block shapes for modeling rock avalanches. *Engineering Geology*, 149–150:78–92, 2012. doi: 10.1016/j.enggeo.2012.07.021.
- [18] Guilhem Mollon, Vincent Richefeu, Pascal Villard, and Dominique Daudon. Numerical simulation of rock avalanches: influence of a local dissipative contact model on the collective behavior of granular flows. *Journal of Geophysical Research: Earth Surface*, 117(F2):F02036, 2012. doi: 10.1029/2011JF002202.
- [19] Guilhem Mollon, Vincent Richefeu, Pascal Villard, and Dominique Daudon. Discrete modelling of rock avalanches: sensitivity to block and slope geometries. *Granular Matter*, 17(5):645–666, 2015. doi: 10.1007/s10035-015-0586-9.
- [20] B. Garcia, V. Richefeu, J. Baroth, D. Daudon, and P. Villard. Collision of shaped boulders with sand substrate investigated by experimental, stochastic, and discrete approaches. *Journal of Geophysical Research: Earth Surface*, 125(11):e2019JF005500, 2020. doi: 10.1029/2019JF005500.
- [21] M. Stasiak, G. Combe, V. Richefeu, G. Armand, and J. Zghondi. High compression of granular assemblies of brittle hollow tubular particles: investigation through a 3D discrete element model. *Computational Particle Mechanics*, 9(4):825–842, 2022. doi: 10.1007/s40571-021-00449-3.
- [22] Duc Chung Vu, Lhassan Amarsid, Jean-Yves Delenne, Vincent Richefeu, and Farhang Radjaï. Rheology and scaling behavior of polyhedral particle flows in rotating drums. *Powder Technology*, 434:119338, 2024. doi: 10.1016/j.powtec.2023.119338.
- [23] Duc Chung Vu, Lhassan Amarsid, Jean-Yves Delenne, Vincent Richefeu, and Farhang Radjaï. Macro-elasticity of granular materials composed of polyhedral particles. *Granular Matter*, 26(1):6, 2024. doi: 10.1007/s10035-023-01382-3.
- [24] Duc Chung Vu, Lhassan Amarsid, Jean-Yves Delenne, Vincent Richefeu, and Farhang Radjaï. Particle fracture regimes from impact simulations. *Physical Review E*, 109(4):044907, 2024. doi: 10.1103/PhysRevE.109.044907.
- [25] Raphaël Prat, Thierry Carrard, Lhassan Amarsid, Vincent Richefeu, Carlo-elia Doncecchi, Paul Lafourcade, Guillaume Latu, and Jean-Mathieu Vanson. ExaDEM: a HPC application based on exaNBody targeting scalable DEM simulations with complex particle shapes. *Journal of Open Source Software*, 10(106):7484, 2025. doi: 10.21105/joss.07484.
- [26] Jaroslaw R. Rossignac and Aristides A. G. Requicha. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, 3(2):129–148, 1986. doi: 10.1016/0167-8396(86)90017-8.
- [27] Gokul Varadhan and Dinesh Manocha. Accurate Minkowski sum approximation of polyhedral models. *Graphical Models*, 68(4):343–355, 2006. doi: 10.1016/j.gmod.2005.11.003.
- [28] Darko Pavić and Leif Kobbelt. High-resolution volumetric computation of offset surfaces with feature preservation. *Computer Graphics Forum (Proc. Eurographics)*, 27(2):165–174, 2008. doi: 10.1111/j.1467-8659.2008.01113.x.
- [29] Rolf Schneider. *Convex Bodies: The Brunn–Minkowski Theory*, volume 151 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2 edition, 2014. doi: 10.1017/CBO9781139003858.
- [30] Mark W. Jones, J. Andreas Bærentzen, and Milos Sramek. 3D distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, 2006. doi: 10.1109/TVCG.2006.56.
- [31] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pages 249–254. ACM Press, 2000. doi: 10.1145/344779.344899.
- [32] J. Andreas Bærentzen and Henrik Aanæs. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):243–253, 2005. doi: 10.1109/TVCG.2005.49.

- [33] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, Cambridge, 2 edition, 1998. ISBN 978-0-521-64976-6.
- [34] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997. doi: 10.1080/10867651.1997.10487468.
- [35] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics*, 32(4):33:1–33:12, 2013. doi: 10.1145/2461912.2461916.
- [36] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997. doi: 10.1080/10867651.1997.10487480.
- [37] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996. doi: 10.1007/s003710050084.
- [38] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. doi: 10.1145/37402.37422.
- [39] Timothy S. Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5): 854–879, 2006. doi: 10.1016/j.cag.2006.07.021.
- [40] Gregory M. Nielson and Bernd Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd Conference on Visualization (VIS '91)*, pages 83–91. IEEE Computer Society Press, 1991. doi: 10.1109/VISUAL.1991.175782.
- [41] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003. doi: 10.1080/10867651.2003.10487582.
- [42] Jules Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988. doi: 10.1016/0167-8396(88)90013-1.
- [43] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanerke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pages 57–66. ACM Press, 2001. doi: 10.1145/383259.383265.
- [44] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of Hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002. doi: 10.1145/566654.566586.
- [45] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. A K Peters / CRC Press, Natick, MA, 2010. ISBN 978-1-56881-426-1.
- [46] Paul Hinker and Charles Hansen. Geometric optimization. In Gregory M. Nielson and R. Daniel Bergeron, editors, *Proceedings of IEEE Visualization '93*, pages 189–195. IEEE Computer Society Press, 1993. doi: 10.1109/VISUAL.1993.398868.
- [47] Alan D. Kalvin and Russell H. Taylor. Superfaces: polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, 1996. doi: 10.1109/38.491187.
- [48] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *ACM SIGGRAPH Computer Graphics*, 26(2):65–70, 1992. doi: 10.1145/142920.134010.
- [49] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pages 209–216. ACM Press, 1997. doi: 10.1145/258734.258849.
- [50] Gary H. Meisters. Polygons have ears. *The American Mathematical Monthly*, 82(6):648–651, 1975. doi: 10.1080/00029890.1975.11993898.
- [51] Martin Held. FIST: Fast industrial-strength triangulation of polygons. *Algorithmica*, 30(4):563–596, 2001. doi: 10.1007/s00453-001-0028-4.
- [52] Sheue-ling Lien and James T. Kajiya. A symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra. *IEEE Computer Graphics and Applications*, 4(10):35–42, 1984. doi: 10.1109/MCG.1984.6429334.

- [53] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2): 31–50, 1996. doi: 10.1080/10867651.1996.10487458.
- [54] Fulvio Tonon. Explicit exact formulas for the 3-D tetrahedron inertia tensor in terms of its vertex coordinates. *Journal of Mathematics and Statistics*, 1(1):8–11, 2005. doi: 10.3844/jmssp.2005.8.11.
- [55] Michael Kallay. Computing the moment of inertia of a solid defined by a triangle mesh. *Journal of Graphics Tools*, 11(2):51–57, 2006. doi: 10.1080/2151237X.2006.10129220.
- [56] Ilya M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. doi: 10.1016/0041-5553(67)90144-9.
- [57] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1992. doi: 10.1137/1.9781611970081.
- [58] Zhijian He and Xiaoqun Wang. On the convergence rate of randomized quasi-Monte Carlo for discontinuous functions. *SIAM Journal on Numerical Analysis*, 53(5):2488–2503, 2015. doi: 10.1137/15M1007963.
- [59] Leonardo Dagum and Ramesh Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55, 1998. doi: 10.1109/99.660313.
- [60] R. Cranley and T. N. L. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13(6):904–914, 1976. doi: 10.1137/0713071.
- [61] Lewis F. Richardson and J. Arthur Gaunt. The deferred approach to the limit. *Philosophical Transactions of the Royal Society of London A*, 226:299–361, 1927. doi: 10.1098/rsta.1927.0008.