PR-Cage: Progressive Feasibility Relaxation for Tight Bounding Cage Generation

HUIBIAO WEN, Shandong University, China and University of Health and Rehabilitation Sciences, China
KAIKAI QIN, Hangzhou Dianzi University, China
XINXIN SU, Shandong Huayun Technology Co., Ltd., China
JINGCHENG MEI, HoteamSoft, China
SHUANGMIN CHEN, Qingdao University of Science and Technology, China and Shandong Key Laboratory of Deep Sea Equipment Intelligent Networking, China
CHONGYANG DENG, Hangzhou Dianzi University, China
CHANGHE TU, Shandong University, China
SHIQING XIN*, Shandong University, China
WENPING WANG, Texas A&M University, USA

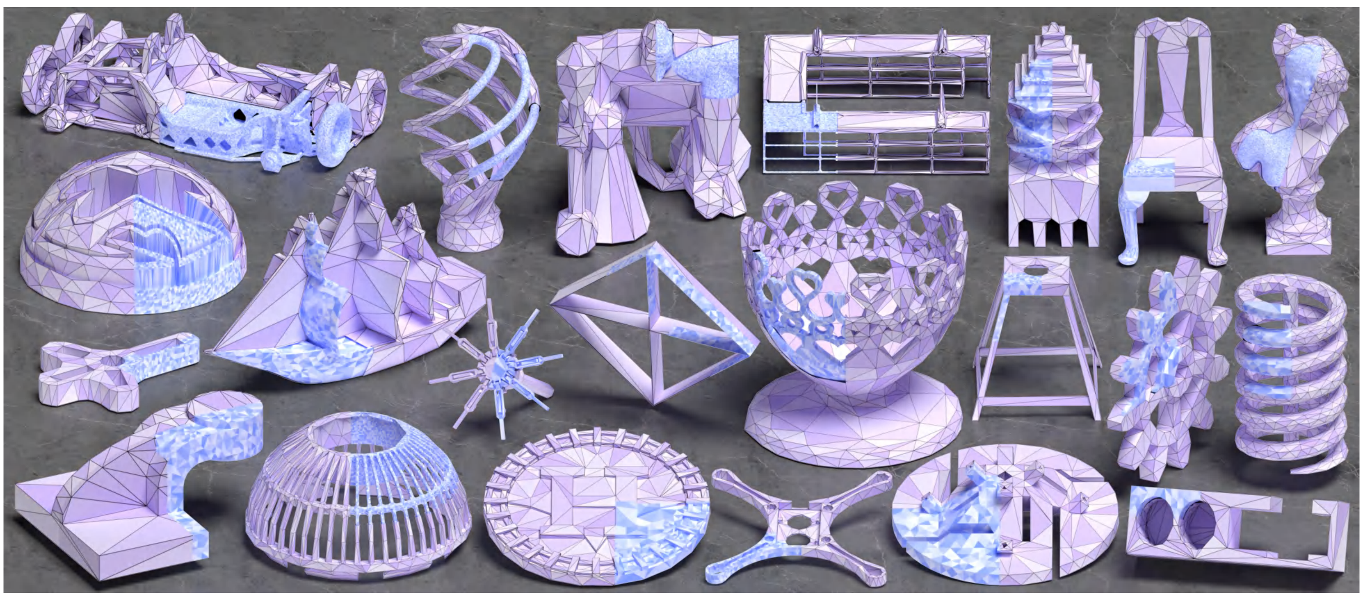


Fig. 1. A gallery of cages generated by PR-Cage on representative models from the Thingi10K [Zhou and Jacobson 2016] and ABC [Koch et al. 2019] datasets, encompassing diverse geometric shapes. Despite the presence of thin plates, tubular structures, and high topological complexity, PR-Cage consistently achieves an optimal balance between simplicity and tightness by progressively relaxing the thickness parameter via a staircase schedule.

Cages are fundamental structures in computer graphics, serving as versatile proxies for a wide range of applications. A high-quality cage must balance two competing objectives: minimizing the face count to ensure simplicity, and maximizing tightness to maintain high geometric fidelity to the input mesh. In this paper, we propose PR-Cage, a nested optimization framework for automated cage generation.

For the outer control layer, we introduce a thickness parameter τ that defines a feasibility region; the evolving cage is guided by the τ-offset surface. We observe that an optimal balance between simplicity and tightness

*Corresponding author: Shiqing Xin.

Authors' Contact Information: Huibiao Wen, Shandong University, China and University of Health and Rehabilitation Sciences, China, ericvein@163.com; Kaikai Qin, Hangzhou Dianzi University, China, qin-kaikai@hdu.edu.cn; Xinxin Su, Shandong Huayun Technology Co., Ltd., China, suxinxin@crowncad.com; Jingcheng Mei, HoteamSoft, China, mjc@hoteamsoft.com; Shuangmin Chen, Qingdao University of Science and Technology, China and Shandong Key Laboratory of Deep Sea Equipment Intelligent Networking, China, csmqq@163.com; Chongyang Deng, Hangzhou Dianzi University, China, dcy@hdu.edu.cn; Changhe Tu, Shandong University, China, chtu@sdu.edu.cn; Shiqing Xin, Shandong University, China, xinshiqing@sdu.edu.cn; Wenping Wang, Texas A&M University, USA, wenping@tamu.edu.

ACM 1557-7368/2026/8-ART149
https://doi.org/10.1145/3811300

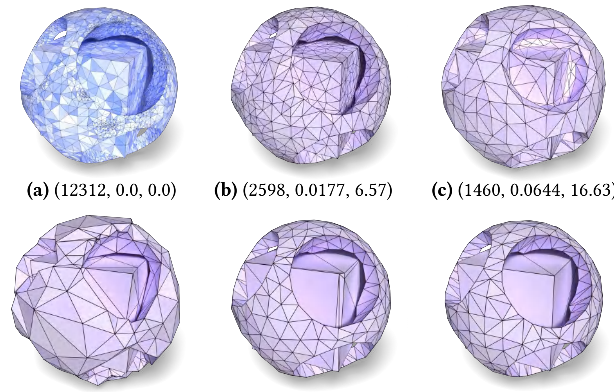
is achievable by progressively relaxing the parameter τ via a staircase schedule. For the inner iterations, we extend the traditional Quadric Error Metric (QEM) framework by incorporating rigorous linear inequality constraints to suppress triangle degeneration and prevent normal flips.

Our algorithm relies exclusively on the atomic operations of edge collapses and edge flips, resulting in high computational efficiency and robustness. Comparative experiments on public datasets demonstrate that PR-Cage consistently outperforms existing methods, achieving extreme simplification while maintaining high adherence to the underlying geometry; see the teaser figure. Due to these favorable properties, we demonstrate the utility of our method in several downstream applications, such as contact simulation and deformation, where PR-Cage exhibits significant advantages in both quality and performance.

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh simplification**.

Additional Key Words and Phrases: Cage Generation, Mesh Simplification, Progressive Relaxation, Geometric Optimization

ACM Reference Format:
Huibiao Wen, Kaikai Qin, Xinxin Su, Jingcheng Mei, Shuangmin Chen, Chongyang Deng, Changhe Tu, Shiqing Xin, and Wenping Wang. 2026. PR-Cage: Progressive Feasibility Relaxation for Tight Bounding Cage Generation. *ACM Trans. Graph.* 45, 4, Article 149 (August 2026), 16 pages. https://doi.org/10.1145/



(a) (12312, 0.0, 0.0) **(b)** (2598, 0.0177, 6.57) **(c)** (1460, 0.0644, 16.63)

(d) (928, 0.0634, 31.66) **(e)** (928, 0.0095, 5.36) **(f)** (928, 0.00535, 2.69)

Fig. 2. Comparison of cage generation methods applied to the input model **(a)**. **(b–f)** Results from Nested Cage [Sacht et al. 2015], Broxy [Calderon and Boubekeur 2017], BPShell [Jiang et al. 2020], Robust Cage [Guo et al. 2024], and PR-Cage (Ours). Metrics listed are: (triangle count, bidirectional Hausdorff distance, and average normal deviation). Our method achieves the tightest enclosure (lowest Hausdorff distance) and superior shape preservation (lowest normal deviation) while maintaining the lowest triangle count among all comparisons.

1 Introduction

In geometry processing and 3D modeling, **cages** serve as versatile, low-complexity proxies for high-resolution surfaces across a diverse range of applications. First, they provide an intuitive control structure for space deformation and animation [Ju et al. 2005; Tong et al. 2025; Yifan et al. 2020], preserving local surface details while enabling global shape editing. Second, cages ensure computational tractability to facilitate physically based simulation and conservative collision handling [Calderon and Boubekeur 2017; Stuart et al. 2013], particularly in scenarios where processing original dense meshes would be prohibitively expensive. Finally, cages have emerged as a vital representation for the neural parameterization of unstructured meshes [Edelstein et al. 2025] and for multiresolution geometry abstraction [Sacht et al. 2015], providing efficient level-of-detail control in complex scenes.

For these applications, a "high-quality" cage must satisfy a demanding combination of geometric and topological properties [Guo et al. 2024; Qiu et al. 2024; Ströter et al. 2024]: simplicity, strict enclosure, manifoldness, watertightness, and the absence of self-intersections. Practically, an ideal cage should: (1) possess a low triangle count to ensure the proxy is significantly cheaper to process than the original mesh; (2) be free of self-intersections; (3) inherit manifold topology conformal to the input geometry; (4) rigorously enclose the input geometry while maximizing shape adherence; and (5) maintain reasonable triangle quality to ensure numerical stability.

While many cage generation algorithms have been proposed [Ben-Chen et al. 2009; Calderon and Boubekeur 2017; Deng et al. 2011; Guo et al. 2024; Płocharski et al. 2024; Qiu et al. 2024; Xian et al. 2015, 2009], existing pipelines typically address only a subset of these requirements. As shown in Fig. 2, no current method simultaneously satisfies all aforementioned criteria (see the survey in [Ströter et al. 2024]). Particularly, minimizing face count to ensure simplicity while maximizing tightness to maintain high geometric fidelity remains a significant challenge.

To address these gaps, we propose PR-Cage, a nested optimization framework for automated cage generation. Our key insight is that an optimal balance between simplicity and tightness is achievable by optimizing the cage under a τ-based feasible region, defined with respect to the input surface and an outward τ-offset bound, and by progressively relaxing the thickness parameter τ via a staircase schedule. For a fixed τ, we extend the traditional Quadric Error Metric (QEM) framework by incorporating rigorous linear inequality constraints to suppress triangle degeneration and prevent normal flips. This constrained optimization is solved efficiently using an active-set method. As demonstrated in Fig. 2, our method produces cages with substantially lower geometric complexity and markedly higher shape fidelity than representative volumetric or tetrahedral approaches. Notably, our algorithm adaptively allocates more triangles to high-curvature regions while maintaining extreme simplicity in flat areas.

In summary, our contributions are threefold:

(1) We introduce PR-Cage, a nested optimization framework that generates high-fidelity cages by progressively relaxing a thickness-based feasibility constraint via a staircase schedule.
(2) We propose a constrained QEM formulation that employs linear inequality constraints to ensure topological integrity and prevent geometric artifacts, relying exclusively on efficient atomic operations such as edge collapses and flips.
(3) We demonstrate the practical utility of PR-Cage in contact simulation and deformation, where it exhibits significant advantages in both geometric quality and computational performance.

2 Related Work

In this section, we review representative methods for cage generation from polygonal meshes. Since our objectives partially overlap with those of surface wrapping and bounding volume generation, we also discuss relevant literature in these areas, highlighting their connections to and distinctions from cage-based representations.

2.1 Cage Generation

Existing methods for automatic cage construction on polygonal meshes can be broadly grouped into three families: (1) offset- or surface-based simplification methods; (2) volumetric or voxelization-based cage extraction; and (3) structure-guided constructions used in deformation frameworks. Below, we briefly review these categories and highlight the limitations that motivate our approach.

Offset and Surface-Based Cage Construction. Early automatic cage generation techniques largely built upon mesh simplification and offset-surface manipulation. The progressive hull of [Sander et al. 2000], derived from Hoppe's progressive meshes [Hoppe 1996], enforces convex-hull containment during edge collapses, yielding enclosing but coarse sequential envelopes that struggle with non-convex or topologically complex shapes. Offset-based methods refine this idea by operating on isosurfaces of distance fields [Deng et al. 2011; Qiu et al. 2024; Sacht et al. 2015; Shen et al. 2004; Xian et al. 2015], improving robustness but introducing a strong dependence on grid resolution and often requiring explicit self-intersection handling [Karras 2012]. Closely related wrapping and shell constructions [Dai et al. 2024; Jiang et al. 2020] provide conservative enclosures but may incur noticeable geometric distortion.

Volumetric and Voxelization-Based Cage Extraction. Volumetric discretizations provide another major class of cage generation techniques, where the cage is extracted as the boundary of a volumetric representation rather than being directly simplified from the input surface. One line of work embeds the input mesh into volumetric tessellations [Guo et al. 2024; Stuart et al. 2013], such as tetrahedral meshes. Guo et al. [2024], for example, construct cages by embedding the surface inside a quality tetrahedral mesh and simplifying the outer boundary under explicit approximation bounds. While this strategy provides strong control over approximation error and avoids self-intersections, the resulting cage density is largely dictated by the volumetric mesh, often leading to overly dense proxies even for simple shapes.

Voxelization-based methods [Faraj et al. 2012; Nesme et al. 2009; Xian et al. 2009, 2012] form a closely related family. Broxy [Calderon and Boubekeur 2017] builds heterogeneous voxel grids via conservative GPU-parallel voxelization, applies morphological operators to regularize the occupied region, and extracts a quad-dominant cage. These methods are robust by construction, yet their behavior is strictly governed by voxel resolution: thin features may disappear, fine-scale adaptivity is limited, and obtaining cages that are both tight and sparse often requires manual tuning.

Structure-Guided Cage Construction. A separate line of work leverages high-level structural priors such as skeletons or templates. These methods stem primarily from deformation and rigging pipelines, where the goal is to obtain a controllable, low-dimensional structure aligned with semantic features rather than a tight enclosure of arbitrary geometry. Skeleton-based methods [Casti et al. 2019; Chen and Feng 2014; Yang et al. 2013] generate cages by stitching cross-sections along curve skeletons, but they depend on robust skeleton extraction and typically require volumetric meshing. Template-driven strategies [Ju et al. 2008] reuse pre-designed cages for specific shape categories, enabling consistent rigging but restricting applicability to shapes outside the template library. Interactive systems provide direct user control [Calderon and Boubekeur 2017; Le and Deng 2017], but these demand substantial manual effort and are not intended for fully automatic construction on arbitrary inputs.

2.2 Wrap Surfaces

Classical shrink-wrapping techniques build an initial watertight wrapper around the input and iteratively shrink it toward the surface. Early work [Kobbelt et al. 1999] and later variants based on octree labeling [Huang et al. 2018; Lee et al. 2010] generate wrapper surfaces from volumetric cell complexes, which are then pulled toward the geometry via vertex projection or Laplacian smoothing [Huang et al. 2020; Juretić and Putz 2012; Martineau et al. 2016]. These methods are widely used for mesh repair but remain highly resolution-dependent.

More recent methods pursue wrapping surfaces with formal guarantees. Delaunay-based constructions with offsets can produce orientable, watertight outer hulls that enclose even noisy or non-manifold inputs [Dai et al. 2024; Portaneri et al. 2022]. While wrap surfaces are conceptually close to cages, they are primarily designed as robust outer hulls for reconstruction or repair, prioritizing conservativeness and robustness over the extreme low complexity and surface adaptivity required for cage-based applications.

2.3 Bounding Volumes

Bounding volumes approximate complex geometry using simple convex primitives (e.g., spheres, boxes, k-DOPs) to ensure efficiency in collision detection and visibility queries [Ericson 2005; Kay and Kajiya 1986; Schneider and Everly 2003]. Bounding Volume Hierarchies (BVHs) organize these primitives into tree structures; for ray tracing, axis-aligned bounding boxes (AABBs) are the de facto standard due to their intersection performance [Meister et al. 2021].

Despite the shared concept of enclosure, BVHs differ fundamentally from cages. Designed for query acceleration, BVHs prioritize hierarchy traversal speed over geometric fidelity. They comprise collections of loose primitives rather than a single simplified, watertight mesh. Although recent neural bounding methods [Liu et al. 2024] have moved beyond analytic primitives, they do not align with the general requirements of cages.

3 Methodology

3.1 Overview

Given a closed, oriented, manifold triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ that is free of self-intersections, the objective of this work is to construct a lower-complexity proxy surface C that serves as a bounding

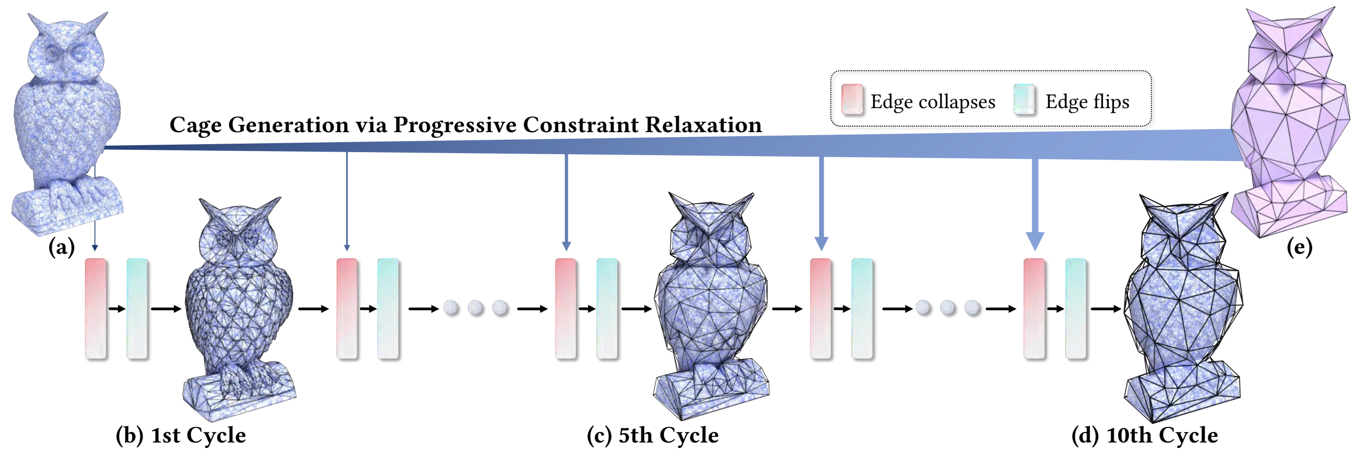


Fig. 3. Overview of our pipeline. Given an input mesh (**a**), we perform constrained edge collapses and edge flips such that the simplified geometry strictly encloses the input surface while its outward deviation is controlled by the thickness parameter τ (**b–d**). PR-Cage operates by incrementally relaxing this thickness parameter toward $\tau_{\max}$ following a staircase schedule, ultimately yielding a low-complexity and highly adhering cage (**e**).

cage for $\mathcal{M}$. We achieve this through a greedy sequence of constrained edge collapses while rigorously satisfying fundamental geometric and topological requirements: *strict enclosure*, *manifoldness*, *watertightness*, *topological equivalence* to the input geometry, and the *absence of self-intersections*.

In addition to these basic requirements, we need to specifically address two critical challenges in automated cage generation:

(1) **High-Fidelity Simplification:** We aim to maximize shape adherence even at extreme simplification levels, ensuring the cage automatically captures the prominent features of the input geometry.
(2) **Non-degeneracy Regularization:** We need to suppress triangle degeneration and normal flips, ensuring numerical stability throughout the progressive simplification process.

To resolve these challenges, our PR-Cage framework employs a nested optimization strategy. The outer layer governs the progressive relaxation of a thickness-based feasibility region. Within this constraint, the inner layer executes constrained edge collapses and edge flips to control geometric deviation while strictly enclosing the input surface. In practice, the final cage is controlled by two complementary factors: the thickness parameter τ, which regulates geometric adherence, and an optional stopping criterion, which can be used to terminate the optimization once a desired face budget is reached. An overview of the PR-Cage pipeline is illustrated in Fig. 3.

3.2 τ-Constrained Edge Collapses and Edge Flips

Constrained Edge Collapses. We adopt the classical Quadric Error Metric (QEM) [Garland and Heckbert 1997] to guide the edge collapse process. For each triangle $f \in \mathcal{F}$ with unit outward normal $\mathbf{n}_f$ and centroid $\mathbf{c}_f$, the squared distance from a point $\mathbf{p}$ to the supporting plane of f is given by $((\mathbf{p} - \mathbf{c}_f) \cdot \mathbf{n}_f)^2$. This distance can be expressed as a quadratic form:

$$Q_f(\mathbf{p}) = \mathbf{p}^\top \mathbf{A}_f \mathbf{p} + 2\mathbf{b}_f^\top \mathbf{p} + d_f. \tag{1}$$

When an edge e is contracted to a new vertex $\mathbf{v}$, the cumulative squared error is defined as the sum of the quadrics associated with all triangles incident to the endpoints of e. Following the original QEM framework, the optimal position for $\mathbf{v}$ is sought by solving the following unconstrained minimization problem:

$$\mathbf{v} = \arg\min_{\mathbf{p}_e \in \mathbb{R}^3} \left(\mathbf{p}_e^\top \mathbf{A}_e \mathbf{p}_e + 2\mathbf{b}_e^\top \mathbf{p}_e + d_e \right). \tag{2}$$

Each of the $\mathbf{A}_e$, $\mathbf{b}_e$, d_e terms are the sums of the respective terms over the triangle neighbors of e. However, the solution to this unconstrained optimization cannot guarantee that the resulting mesh maintains the strict enclosure property. Furthermore, it may lead to geometric or topological artifacts, such as self-intersections or normal flips, which are unacceptable for high-quality cage generation.

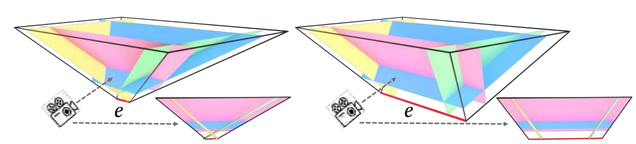


Fig. 4. τ-enclosure for a concave edge feature. **Left**: When the edge e is short relative to τ, the feasible region for the target contraction point is non-empty. **Right**: When the edge e is long relative to τ, the intersection of slab constraints results in an empty feasible region, thereby preventing the edge contraction.

τ-Enclosure for Feature Preservation. Let $\mathcal{M}_\tau$ denote the outward offset surface at a distance τ from the input mesh $\mathcal{M}$ (noting that $\mathcal{M}_\tau$ does not need to be explicitly computed). We require that any decimation operator—whether an edge collapse or an edge flip—keeps the updated local geometry within a τ-controlled band between $\mathcal{M}$ and $\mathcal{M}_\tau$. To achieve this, for each triangle f incident to the endpoints of e, we enforce the following constraint on the new vertex position $\mathbf{p}_e$:

$$0 \le \mathbf{n}_f^\top (\mathbf{p}_e - \mathbf{c}_f) \le \tau, \tag{3}$$

where each incident face defines a local *slab* of thickness τ along its normal direction $\mathbf{n}_f$.

As illustrated in Fig. 4, this formulation requires the target point $\mathbf{p}_e$ to lie within the intersection of all such slabs. When the edge e is short relative to τ (left), the intersection of these linear inequalities is non-empty, permitting the collapse. Conversely, for a long edge e (right), the opposing slab constraints result in an empty feasible region, effectively preventing the contraction. This geometric behavior holds for both convex and concave regions. Consequently, these constraints not only ensure geometric adherence (fidelity) but also naturally facilitate the preservation of dominant features by limiting the simplification of large-scale structures that would otherwise deviate too far from the original shell.

Critically, Eq. (3) only bounds the deviation relative to the current local configuration. Because a progressive edge contraction process may accumulate error over successive steps, the target point $\mathbf{p}_e$ might eventually drift beyond the τ-offset distance from the original input $\mathcal{M}$. To address this, we reformulate the constraint to account for the current distance from the input surface, $d(\mathbf{c}_f, \mathcal{M})$, yielding a more conservative update rule:

$$0 \leq \mathbf{n}_f^\top(\mathbf{p}_e - \mathbf{c}_f) \leq \tau - d(\mathbf{c}_f, \mathcal{M}), \tag{4}$$

where $d(\mathbf{c}_f, \mathcal{M})$ is the distance from the face centroid $\mathbf{c}_f$ to the input mesh $\mathcal{M}$. We note that the τ-constraint is enforced only during candidate vertex updates via per-face linearized inequality constraints. As such, it provides effective control of outward deviation in practice, but does not constitute a strict guarantee that the evolving cage remains entirely inside the exact τ-offset of the input surface.

As illustrated in Fig. 5, the parameter τ acts as the primary mechanism for balancing geometric fidelity against simplification. When τ is small, the feasible region is restricted to a narrow band, so maintaining strict enclosure typically requires a higher face count. As τ increases, the feasible region expands, giving the optimizer greater spatial freedom to perform more aggressive decimation.

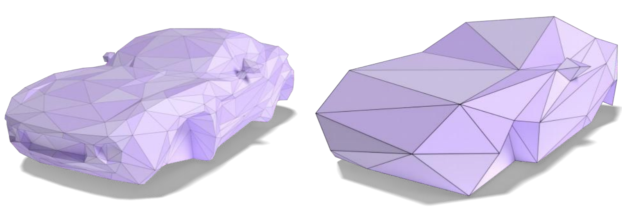


Fig. 5. Influence of the τ-enclosure constraint on cage complexity. **Left:** Result under a restrictive τ, maintaining high geometric adherence to the original surface. **Right:** Result with an expanded τ, facilitating aggressive decimation and a significantly reduced face count.

Following standard requirements in the literature [Guo et al. 2024; Sacht et al. 2015], a valid cage should ideally not touch the original surface to ensure a strictly positive enclosure. Consequently, we initially displace all vertices of $\mathcal{M}$ outward along their respective normals by a safety margin $\sigma = 10^{-5}l$, where l denotes the diagonal length of the bounding box of $\mathcal{M}$. If this displacement induces self-intersections, we iteratively halve σ until the resulting mesh remains manifold and intersection-free.

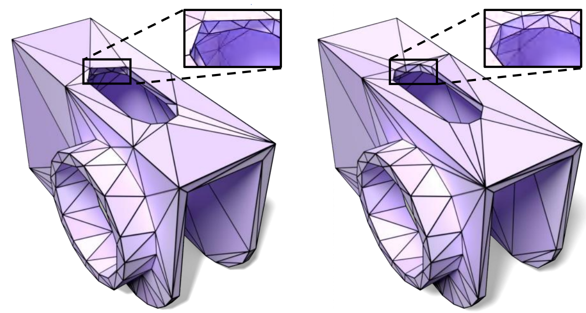
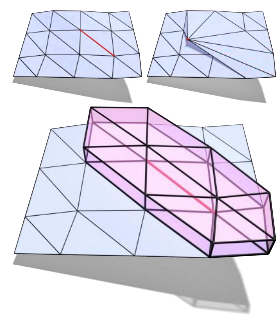


Fig. 6. Effect of orientation constraints. **Left**: Foldovers occurring without constraints. **Right**: Foldovers are effectively eliminated by our orientation constraints.

Suppressing Foldovers. Despite the τ-enclosure constraint, the target vertex may still drift within the feasibility region between $\mathcal{M}$ and $\mathcal{M}_\tau$ in a way that causes local triangle inversions or "foldovers," as illustrated in the inset figure (top).



To suppress foldovers, we require that the normal of each updated face remains fundamentally aligned with its original orientation. Suppose $f = \triangle(\mathbf{v}_1, \mathbf{v}_2, \mathbf{a})$ is a triangle incident to endpoint $\mathbf{a}$ of the edge $e = (\mathbf{a}, \mathbf{b})$ being contracted. After the collapse to $\mathbf{p}_e$, the new triangle is $f' = \triangle(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}_e)$. We enforce the following orientation constraint:

$$\big((\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{p}_e - \mathbf{v}_1)\big) \cdot \mathbf{n}_f \geq \epsilon, \tag{5}$$

where $\epsilon > 0$ is a small threshold to prevent degeneracy. This defines a half-space constraint for the target vertex $\mathbf{p}_e$ (see the pink prism in the inset figure). As shown in Fig. 6, neglecting these orientation constraints can result in near-180° dihedral angles (left), whereas their inclusion (right) effectively prevents foldovers and yields a clean, high-quality cage. Note that Eq. (5) only enforces local orientation consistency in the updated one-ring; global self-intersection avoidance is handled separately by explicit collision tests.

QP Solver. Both the enclosure constraints (Eq. (4)) and the orientation constraints (Eq. (5)) are linear with respect to the target point $\mathbf{p}_e$. Consequently, each edge collapse is formulated as a small Quadratic Programming (QP) problem involving only three variables and a limited number of linear inequalities (typically around 20). We solve this efficiently using an active-set method [Goldfarb and Idnani 1983].

Edge-flip Stage. Following the decimation phase, we perform a refinement stage using edge flips to enhance the cage's geometric quality while maintaining a constant face count. For a candidate edge e shared by two triangles, we consider flipping it to the alternative diagonal e' if and only if the following criteria are simultaneously satisfied:

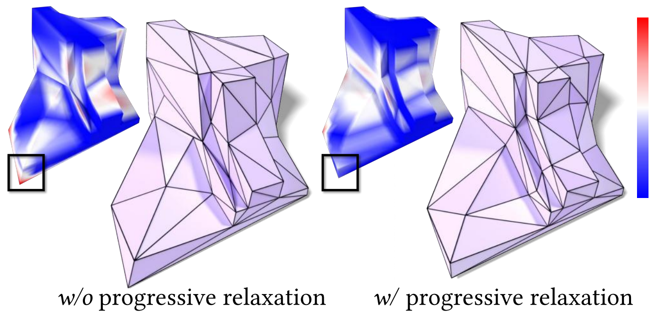


Fig. 7. Effect of progressive thickness relaxation. **Left:** Result of directly setting $\tau = \tau_{\max}$. **Right:** By relaxing τ toward $\tau_{\max}$ via a staircase schedule, our method ensures high fidelity and preserves prominent shape features. Insets visualize the Euclidean distance error (Blue: 1.3×10^{-6}; Red: 3.99×10^{-2}).

(1) **Quality Improvement:** The flip must increase the minimum interior angle within the local quadrilateral footprint.
(2) **Feasibility Maintenance:** The flipped edge e' must lie entirely within the volumetric union of the local slabs defined by the two incident faces, thereby satisfying the τ-enclosure requirements of Eq. (4). Since fine-scale features were already simplified or anchored during the preceding decimation, this localized adjustment enhances mesh regularity without altering the structural essence of the cage.

Because an edge flip only modifies the immediate neighborhood, these checks are performed efficiently. We apply these flips iteratively in an arbitrary order until local optimality is reached. This process ensures that the overall mesh regularity and shape adherence are optimized while preserving the established enclosure relationship.

Self-intersection Handling. In addition to the local orientation consistency enforced by Eq. (5), we explicitly prevent self-intersections using an operation-level validity check. Since an edge collapse or edge flip affects only a small local patch of the current cage, while leaving the remainder of the mesh unchanged, it suffices to test whether the updated patch introduces any new intersections. For each candidate operation, we first construct the tentative updated patch induced by the proposed vertex position or connectivity change. We then reject the operation if the updated patch contains any degenerate or intersecting triangle configuration. Next, we test the updated patch against the rest of the cage for triangle-triangle intersections, excluding adjacent triangle pairs. These queries are accelerated using an AABB tree maintained over the current cage, following the implementation in `libigl` [Jacobson et al. 2018]. If any new intersection is detected, the candidate operation is discarded; otherwise, it is accepted. After each accepted update, only the bounding volumes associated with the affected triangles are updated incrementally. Since the initial offset cage is constructed to be self-intersection-free, and each accepted operation is required not to introduce new intersections, the evolving cage remains self-intersection-free throughout the simplification and edge-flip process. These collision checks are also related in spirit to earlier self-intersection-free simplification methods, such as Gumhold et al. [2003].

3.3 Progressive Thickness Relaxation

Rather than setting the thickness τ directly to its maximum value $\tau_{\max}$, we perform relaxation following a staircase schedule. Consequently, the cage generation process spans multiple optimization cycles. As illustrated in Fig. 7, the color-coded Euclidean distance insets demonstrate that this progressive strategy not only maintains high geometric fidelity but also prevents the premature collapse of prominent shape features.

In our implementation, we define the thickness relaxation via a monotonically increasing sequence with uniform steps:

$$0 < \beta_0 < \beta_1 < \cdots < \beta_K = 1. \tag{6}$$

In the k-th cycle, the τ-enclosure constraint for each face f is updated as:

$$0 \le \mathbf{n}_f^\top(\mathbf{p}_e - \mathbf{c}_f) \le \beta_k \tau_{\max} - d(\mathbf{c}_f, \mathcal{M}), \tag{7}$$

where $d(\mathbf{c}_f, \mathcal{M})$ represents the distance from the face centroid $\mathbf{c}_f$ to the input mesh $\mathcal{M}$. This distance is efficiently evaluated using FCPW [Sawhney 2021], an acceleration library for high-performance closest-point and distance queries. A related alternative is to determine an offset parameter through search rather than a prescribed schedule, as in the simple nested cages via binary search of Sellán et al. [2023]. In contrast, we adopt a fixed monotone relaxation schedule as a simple outer control for constrained cage simplification.

Fig. 8 visualizes the results across multiple cycles. We observe that the face count $|F|$ drops steeply during the initial cycles and subsequently levels off, indicating that the majority of mesh decimation occurs early in the process. Similarly, the Hausdorff distance d_H increases during the initial cycles before reaching a stable plateau. Notably, the resulting cage adaptively allocates larger triangles to flat regions, effectively redistributing the degrees of freedom to curvature-heavy areas for improved adherence and feature preservation. Crucially, this behavior is achieved fully automatically without the need for explicit feature labeling.

3.4 Variants of Cages

Topology-Simplified Cages. Since our framework leverages the QEM-based simplification paradigm, the resulting cage naturally inherits the topology of the input mesh. For applications where exact topological fidelity is not a requirement, we instead first construct a topology-simplified proxy surface $\mathcal{M}'$ using the off-the-shelf alpha-wrap method [Portaneri et al. 2022]. Specifically, we employ a relatively large wrapping scale α to eliminate tunnels and thin handles below a target threshold, combined with a small offset parameter $\varsigma = l/3000$ to ensure that $\mathcal{M}'$ remains geometrically close to $\mathcal{M}$. This proxy $\mathcal{M}'$ is then used as the input for our standard pipeline, producing a cage that tightly encloses $\mathcal{M}$ but with significantly reduced topological complexity, as demonstrated in Fig. 9.

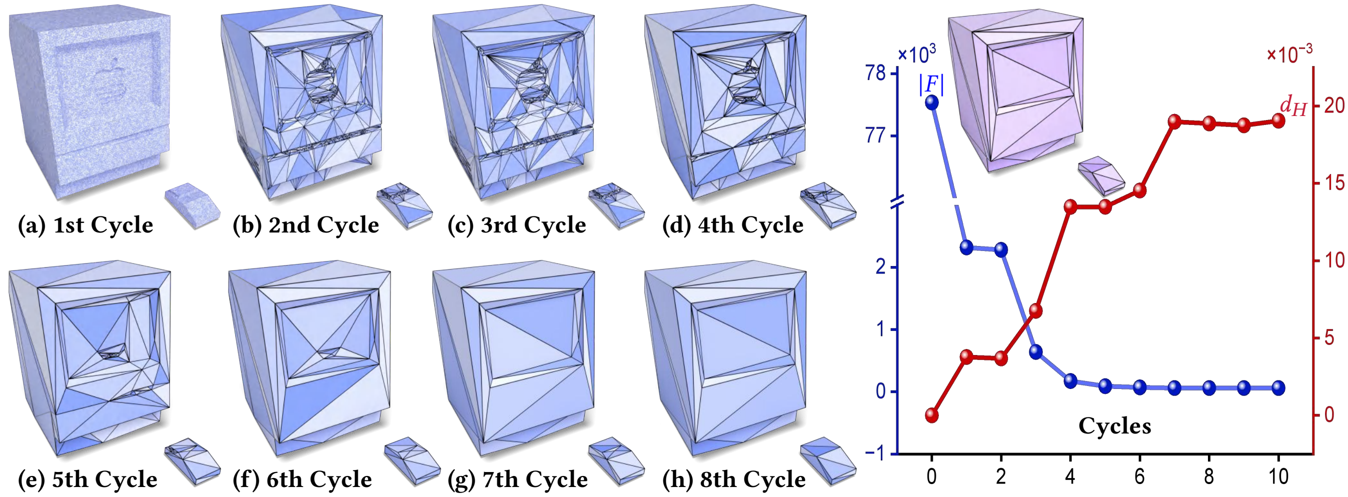


Fig. 8. Visualizing the progressive thickness relaxation process. The rightmost plot illustrates the evolution of the face count $|F|$ and the Hausdorff distance d_H between the cage and the original mesh across optimization cycles.

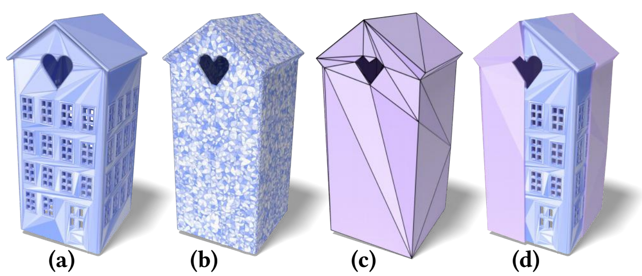
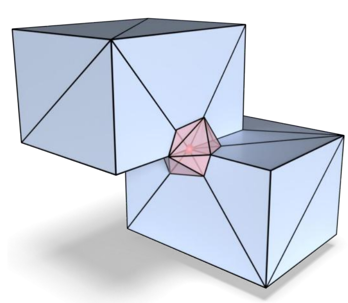


Fig. 9. Topology-simplified cage generation. **(a)** Original model $\mathcal{M}$ with complex genus. **(b)** Topology-simplified proxy surface $\mathcal{M}'$ obtained via alpha-wrap, which closes small openings and thin handles. **(c)** Topology-simplified cage produced by our method. **(d)** Sectional view illustrating that the final cage rigorously encloses $\mathcal{M}$ while maintaining tight geometric adherence.



Handling Singularities. In certain scenarios, the normals of triangles incident to a vertex $\mathbf{v}$ may vary drastically, potentially becoming nearly antiparallel. In such cases, any displacement of $\mathbf{v}$ (while keeping neighbors fixed) leads to penetration into the original surface. To introduce more geometric freedom to $\mathbf{v}$, we perform a local refinement, following the general spirit of BPShell [Jiang et al. 2020]

Specifically, we define a small ε-radius sphere centered at $\mathbf{v}$ and traverse its incident edges in counter-clockwise order to identify a ring of intersection points between the sphere and the mesh edges. Using a stereographic projection (assuming the projection pole lies within the surface), we map these intersection points onto a plane and compute their Delaunay triangulation. This process replaces the immediate neighborhood of $\mathbf{v}$ with a small, well-behaved local cap. This modified local geometry effectively wraps the original singularity and facilitates the successful execution of our cage generation algorithm.

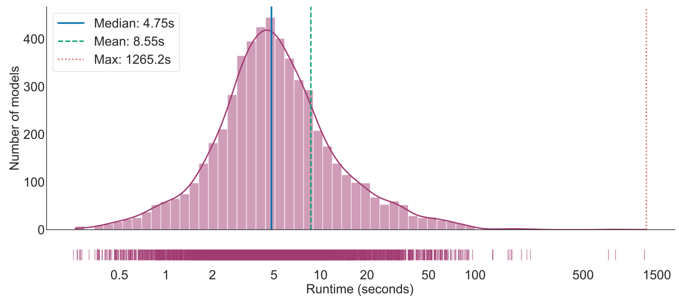
4 Experiments

4.1 Experimental Setting

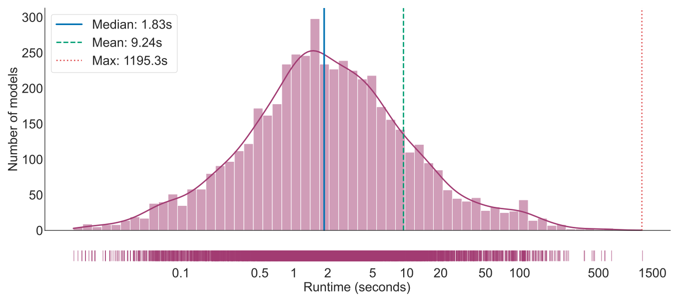
We implemented our algorithm in C++ and evaluated it on a desktop computer equipped with an Intel Core i9-13900K CPU and 64 GB of RAM. Our implementation relies on Eigen [Guennebaud et al. 2010] for linear algebra routines and Libigl [Jacobson et al. 2018] for fundamental geometry processing tasks.

We evaluate our cage generation framework on a large-scale benchmark consisting of 5,046 models from Thingi10K [Zhou and Jacobson 2016] and 5,508 models from ABC [Koch et al. 2019]. The selection criteria ensure all inputs are manifold, self-intersection-free, consistently oriented, and watertight. Thingi10K comprises a diverse collection of real-world meshes with a wide range of geometric and topological complexity, whereas ABC focuses on CAD-style models characterized by sharp features, thin structures, and precise geometric detail.

For the experimental setup, we set the outer offset distance $\tau_{\max} = l/40$ for Thingi10K and $\tau_{\max} = l/80$ for ABC. The parameter $\tau_{\max}$ determines the thickness of the admissible outer band and thus controls the trade-off between simplification freedom and geometric tightness. In general, $\tau_{\max}$ should be chosen according to the geometric proximity and feature density of the target shape family: models with thin structures, narrow gaps, or sharp CAD-like details typically require a smaller value to avoid excessive outward deviation, whereas smoother shapes with fewer close-by features



(a) Runtime of PR-Cage on ABC dataset.



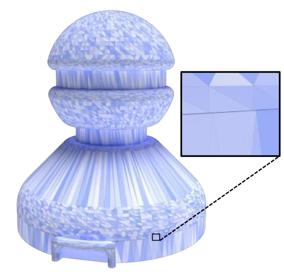
(b) Runtime of PR-Cage on Thingi10K dataset.

Fig. 10. Runtime statistics of PR-Cage on large-scale benchmarks. We report the distribution of per-model runtimes of PR-Cage on the ABC dataset (top) and Thingi10K dataset (bottom). Histograms show the number of models falling into each runtime bin, overlaid with kernel density estimates to visualize the overall distribution. Vertical lines indicate the median (solid blue), mean (dashed green), and maximum (dotted red) runtimes.

can tolerate a larger value to enable more aggressive simplification. The specific choices $l/40$ and $l/80$ were selected empirically as stable dataset-level defaults. We use a relatively larger value for Thingi10K, whose shapes are generally less dominated by sharp CAD-like detail, and a smaller value for ABC, whose models more often contain thin structures, narrow gaps, and sharp features that call for a tighter outer bound. Unless otherwise stated, we fix the maximum number of progressive relaxation cycles to $K = 10$ and use $\beta_k \in \{0.1, 0.2, \ldots, 1.0\}$, yielding a linear increase of the effective thickness bound from an initially conservative stage to the final target value. To ensure a consistent evaluation across different scales, all input meshes are uniformly normalized to fit within a $[-0.5, 0.5]^3$ bounding box.

We report runtime (seconds), relative Hausdorff distance $\overline{d_H}$, and average angular normal deviation $\overline{\theta}$ (degrees), together with a success rate. The cage generation is counted as *successful* only if it satisfies all validity conditions: (i) self-intersection free, (ii) manifold, (iii) strictly enclosing the input (no intersection with the input surface), and (iv) topologically consistent with the input (Euler characteristic mismatch is treated as failure).

4.2 Validation



Across the full benchmark of 10,554 models, PR-Cage successfully generated valid cages for all but 11 cases, indicating strong robustness in practice. The observed failures fall into two categories. The first involves near-degenerate input triangles, which lead to unstable local normal estimation and consequently invalidate the corresponding linear feasibility constraints during simplification. The second arises from models with extremely narrow gaps or nearly touching surface sheets, where the feasible outer region becomes too restricted to support further collapses without violating enclosure, as illustrated in the inset figure.

As illustrated in Fig. 1, the gallery of 22 complex models demonstrates that PR-Cage consistently yields compact, strictly-enclosing cages, even for geometries featuring thin structures, high genus, or intricate details. Fig. 10 further reports the per-model runtime statistics on the ABC and Thingi10K datasets. Notably, PR-Cage exhibits high efficiency: 92% of models from Thingi10K and 93% of models from ABC are processed within 20 seconds. Only a small fraction, 1.1% and 3.4% respectively, require more than one minute, underscoring the method's scalability for large-scale benchmarks. Beyond these aggregate statistics, we observe that the longest-running cases are not determined by input size alone. Besides larger triangle counts, narrow gaps, thin structures, and poor triangulations with many elongated triangles can all increase the cost of validity checks and edge-flip polishing.

Quantitative analysis in Fig. 11 attests to our method's ability to maintain low geometric distortion. For the ABC dataset, the median relative Hausdorff distance is as low as 1.38%, with 75% of the models achieving an error below 2.00%. Even on the topologically diverse Thingi10K dataset, the median error remains consistently low at 2.27%. The prominent clustering of the distribution curves in the low-error regions indicates that our method effectively suppresses surface drifting during cage generation. Furthermore, the average normal deviation centers around 5.36° for ABC and 7.79° for Thingi10K. Notably, 77.5% of the generated cages contain fewer than 500 faces, achieving an orders-of-magnitude reduction in complexity. Despite this significant simplification, our method preserves the structural integrity required for demanding cage-based applications.

4.3 Comparison

We compare our method with four representative state-of-the-art approaches for constructing enclosing proxies/cages: Nested Cage [Sacht et al. 2015], Broxy [Calderon and Boubekeur 2017], BPShell [Jiang et al. 2020] and Robust Cage [Guo et al. 2024]. Our evaluation targets the trade-off between *strict validity* (enclosure, manifoldness, and intersection-freeness) and geometric fidelity under comparable triangle budgets. Broxy relies on grid regularization for robustness, which may sacrifice topological consistency during parameter tuning. BPShell imposes bijectivity within an offset prism, often resulting in geometric distortion and fidelity loss. Robust Cage shares

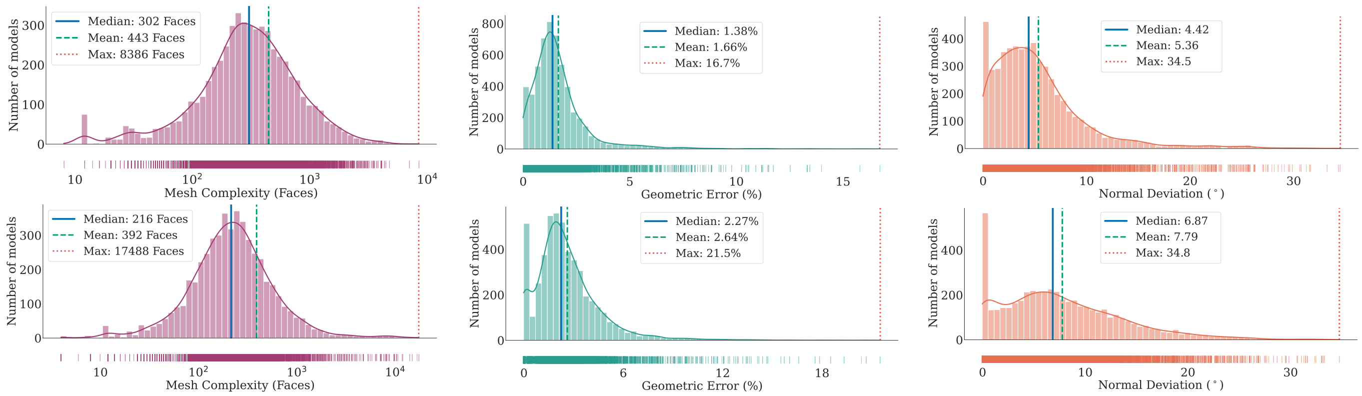


Fig. 11. Quantitative evaluation on large-scale datasets. Statistical distributions of results for the ABC (top row) and Thingi10K (bottom row) benchmarks. **From left to right**: mesh complexity (face count), geometric fidelity (relative Hausdorff distance), and feature preservation (average normal deviation). Our method consistently achieves low geometric and orientation errors, while maintaining low mesh complexity.

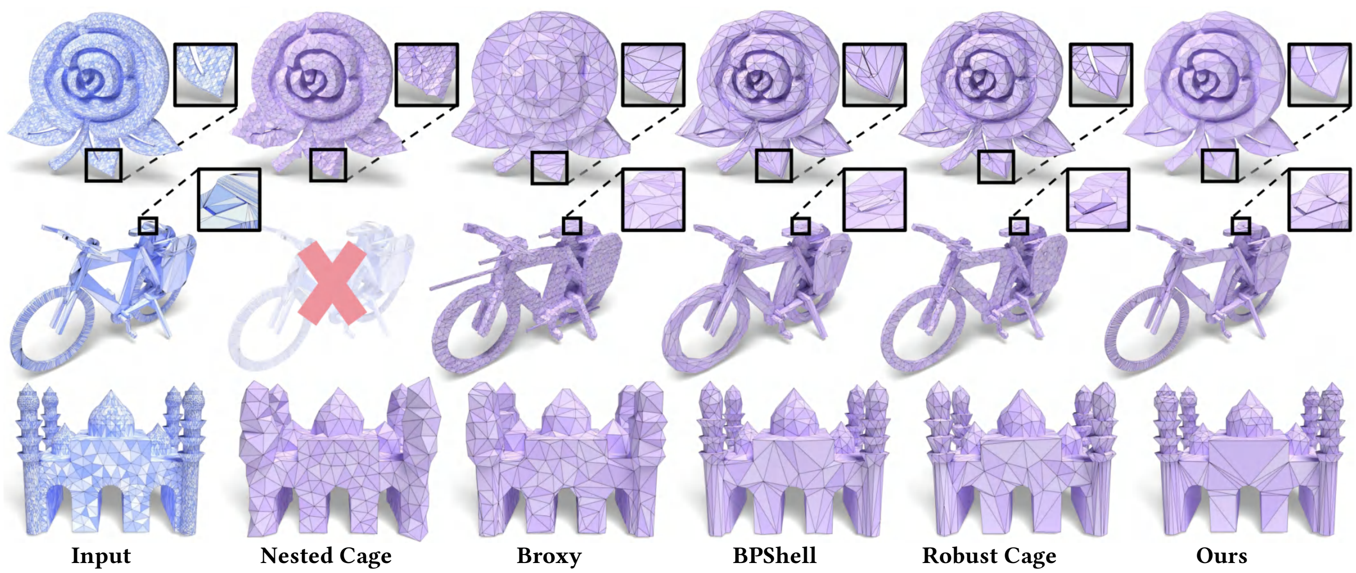


Fig. 12. Comparison of cages generated by state-of-the-art methods on challenging models from Thingi10K. While competing methods lose geometric detail due to uniform triangle allocation, our method adaptively concentrates tessellation on high-curvature regions. This strategy ensures that our generated cages exhibit the highest fidelity to the original shapes, consistently outperforming other methods in capturing the structural essence of the input models.

our goal of strict validity, but its volumetric pipeline is computationally expensive; furthermore, its conservative updates at low face counts often lead to corner rounding. While Nested Cage utilizes SDF-driven optimization, it is susceptible to initialization issues and frequently violates validity constraints on complex models with narrow gaps or ultra-low budgets.

Several baselines require manual parameter tuning and become computationally prohibitive on large inputs. We therefore conduct the evaluation on a representative set of 100 models, 50 from Thingi10K and 50 from ABC, covering a broad range of geometries, including high-genus shapes, thin structures, and feature-rich CAD models. All methods are evaluated on this identical subset. Since BPShell prioritizes a bijective mapping under a geometric tolerance rather than a prescribed face count, we use the complexity of its output (thickness tolerance 0.01) as a reference budget. For methods requiring manual tuning (e.g., Broxy), we adjust parameters to match this reference count as closely as possible. To ensure a fair evaluation, we tested multiple configurations for Nested Cages (varying objective functions and both decimation modes) and selected the highest-quality results for comparison.

Crucially, for each model individually, we set the final evaluation budget to the *minimum* face count achieved by any competing method, and run our method until it reaches this minimum. This protocol guarantees that our method is never advantaged by a higher triangle budget. It is noted that outputs violating any condition are counted as failures in the success rate; when a mesh is produced,

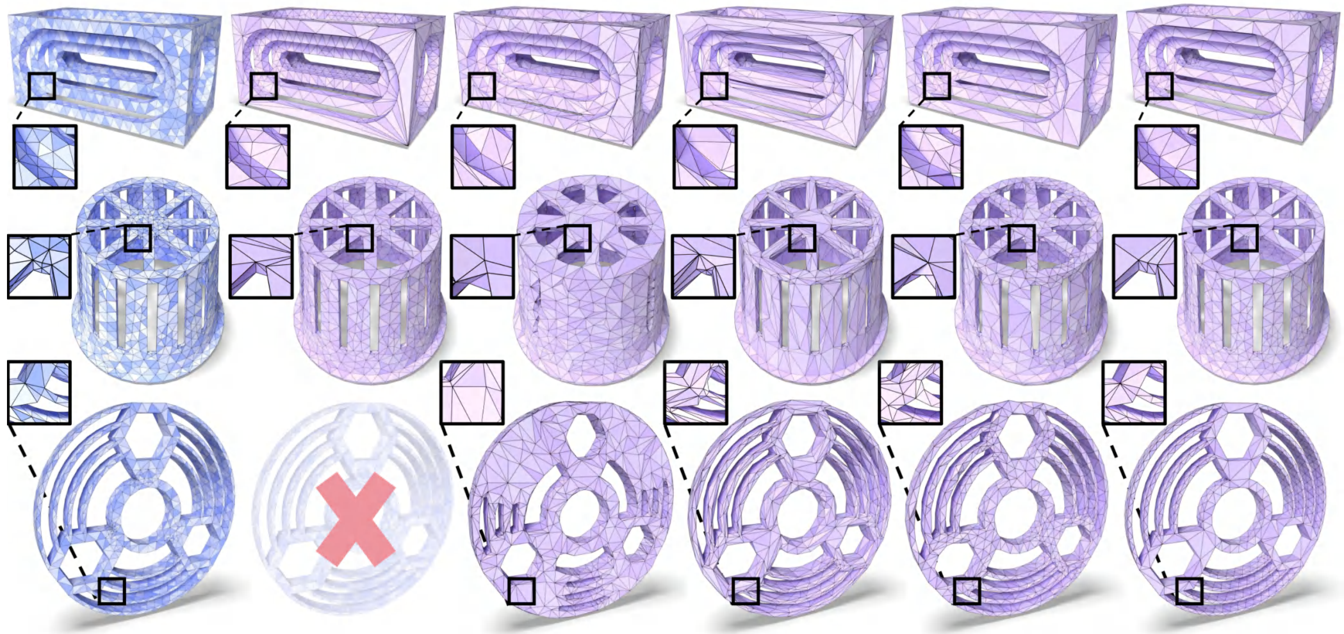


Fig. 13. Comparison of cages generated by state-of-the-art methods on challenging models from ABC.

we additionally report its geometric errors for completeness, while runs producing no output are excluded from metric averaging.

Table 1. Quantitative comparison on Thingi10K and ABC datasets. We report the success rate, average runtime (seconds), mean relative Hausdorff distance ($\overline{d_H}$), and average normal deviation ($\overline{\theta}$). The **best** scores in each column are highlighted in bold.

	Thingi10K Dataset				ABC Dataset			
Method	Success ↑	Time ↓	$\overline{d_H}$ ↓	$\overline{\theta}$ ↓	Success ↑	Time ↓	$\overline{d_H}$ ↓	$\overline{\theta}$ ↓
Nested Cage	28%	610.75	0.01729	10.58	54%	234.48	0.01883	5.91
Broxy	20%	97.08	0.03813	23.01	48%	92.50	0.03514	16.73
BPShell	96%	2611.63	0.01249	9.08	98%	745.01	0.01399	12.46
Robust Cage	94%	2152.43	0.01194	11.55	98%	201.76	0.01262	9.16
Ours	**100%**	**43.02**	**0.01064**	**4.84**	**100%**	**14.81**	**0.00925**	**2.33**

Quantitative results in Table 1 and visual comparisons in Fig. 12 and Fig. 13 highlight the effectiveness of our method. Even under a comparable or smaller triangle budget, PR-Cage consistently achieves the highest shape fidelity by adaptively allocating more triangles to high-curvature regions while maintaining extreme simplicity in flat areas. This emergent property ensures superior geometric consistency and minimal distortion ($\overline{d_H}$, $\overline{\theta}$) over state-of-the-art methods. Regarding Broxy in Fig. 12, attaining an ultra-low triangle budget generally requires a correspondingly coarse grid resolution. Even when the closing operation is disabled, such coarse voxelization often induces undesired topological merging, causing distinct structures—such as individual components of the bicycle—to become artificially connected

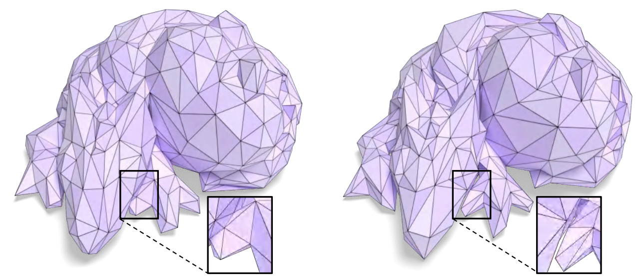


Fig. 14. Comparison of cage generation between the offset-based baseline and our method. **Left:** The offset-based approach fails to preserve topological consistency, introducing an unintended genus change (see zoom-in). **Right:** PR-Cage produces a topology-preserving cage.

Comparison with offset-based approaches. A straightforward baseline for generating an enveloping cage is to first construct an outer offset surface of the input mesh and subsequently apply QEM-based simplification. During the simplification process, edge collapse candidates are discarded if they trigger intersections with the input mesh. However, as illustrated in Fig. 14, this strategy is prone to topological inconsistencies. The primary issue stems from the fact that the offset operation itself does not inherently preserve the topology of the original mesh. In contrast, by performing simplification directly on the input mesh, we ensure that each simplification step is both safe and topologically sound.

Notably, some recent advances, such as Topological Offset [Zint et al. 2025], attempt to address these topological challenges. While this approach can theoretically guarantee a topology-preserving

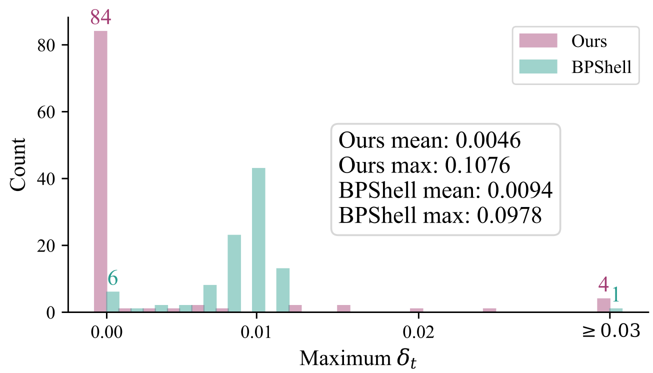


Fig. 15. Comparison with BPShell on thickness-bound violation over the 100-model comparison subset. The histogram shows the distribution of the per-model maximum violation values. The numbers above selected bars indicate the number of models in those bins, and the inset reports the mean and worst-case values for each method.

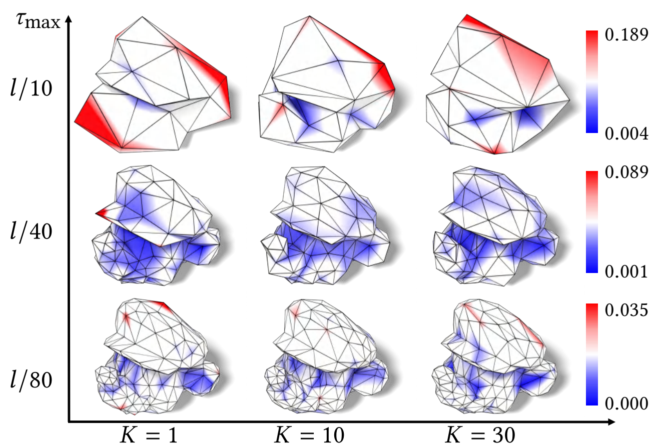


Fig. 16. Signed distance visualization for an ablation over $\tau_{\max}$ and the number of relaxation cycles K. Rows correspond to $\tau_{\max} \in \{l/10, \ l/40, \ l/80\}$ and columns correspond to $K \in \{1, \ 10, \ 30\}$. Each panel shows the resulting cage; vertex colors encode the signed distance to the input surface (blue to red, with per-row color bars).

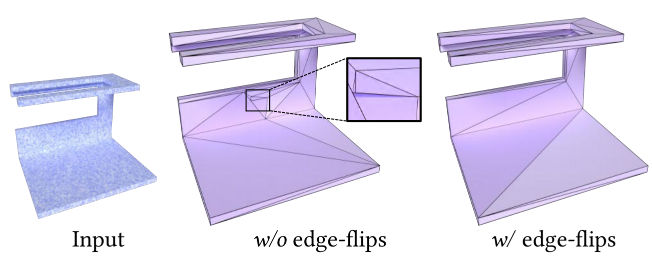


Fig. 17. Effect of edge-flips. The inset highlights a low-quality local triangulation produced without the edge-flip stage.

offset by embedding the input mesh into a tetrahedral mesh and performing operations within the volumetric domain, it introduces significant computational overhead.

Comparison with BPShell on Thickness-bound Violation. To quantify how much the cage exceeds the prescribed thickness bound in practice, we compare our method with BPShell on the same 100-model comparison subset. For each output mesh, we sample 100K points on the cage and measure the violation

$$\delta_t(x) = \max\big(d(x, \mathcal{M}) - t, \ 0\big). \tag{8}$$

Here, $t = \tau_{\max} = l/200$ for our method. For BPShell, however, the thickness parameter specifies the distance between the inner and outer shell surfaces, rather than the outward deviation from the input surface to a single enclosing cage. To make the comparison consistent with our one-sided thickness measure, we therefore use $t = l/200$, corresponding to half of the BPShell thickness $l/100$. For each model, we report the maximum violation over all sampled points. As shown in Fig. 15, our results are much more concentrated near zero, with 84 models falling into the first bin, compared with 6 for BPShell, and a lower mean per-model maximum violation. At the same time, both methods exhibit a small number of outliers, and our method has a slightly larger worst-case value. Overall, although our formulation does not provide a strict continuous thickness guarantee, it yields smaller thickness-bound violation on average than BPShell.

4.4 Ablation Study

Sensitivity to $\tau_{\max}$ and Relaxation Cycles. We study how the outer relaxation schedule interacts with the maximum offset bound by varying $\tau_{\max}$ and the number of relaxation cycles K while keeping all other settings fixed. Enforcing the fully relaxed bound in a single cycle (K=1) often yields localized regions with substantially larger outward deviation, suggesting that too much freedom is introduced before the coarse structure is stabilized. Increasing K mitigates this effect: early conservative cycles anchor the global configuration, while later relaxations gradually allocate the remaining slack, resulting in a tighter and more uniform deviation field. When $\tau_{\max}$ is small, the optimization quickly becomes feasibility-limited, and further increasing K has only a minor impact (Fig. 16).

Effect of edge-flips. We examine the effect of the edge-flip stage on a representative example in Fig. 17. Without edge-flips, the progressive collapse process may accumulate many elongated triangles in local regions. In our framework, this is problematic because the subsequent constrained QP updates are built from local face normals and face-based geometric surrogates. Once the local triangulation becomes excessively skewed, these quantities become less stable numerically, which can degrade later updates and eventually lead to visibly disordered local triangulations. The edge-flip stage alleviates this issue by regularly improving local triangle quality during the simplification process.

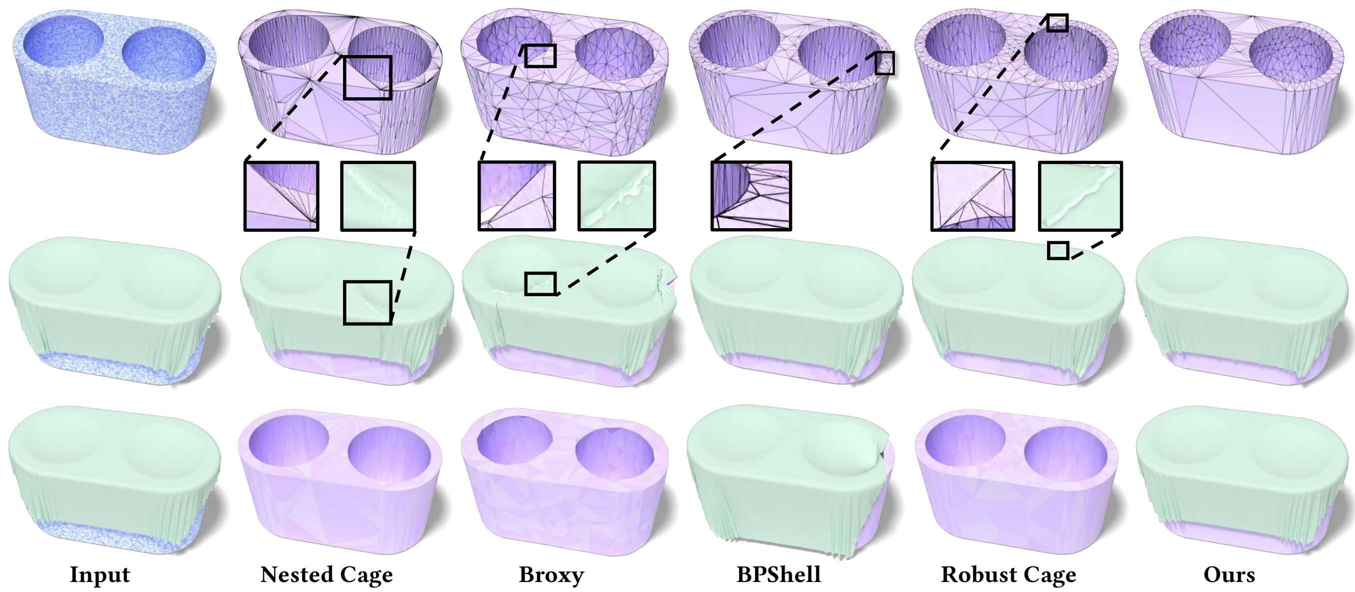


Fig. 18. Cloth simulation results using different collision proxies. **Top:** The original mesh and cages generated by various methods. **Middle:** An intermediate frame (Frame 30) where competing proxies exhibit sliding and local collapse (highlighted). **Bottom:** The quasi-static configuration at Frame 120. At this stage, the cloth remains in its initial position only for the original mesh and PR-Cage, while it has prematurely slid off for the other proxies.

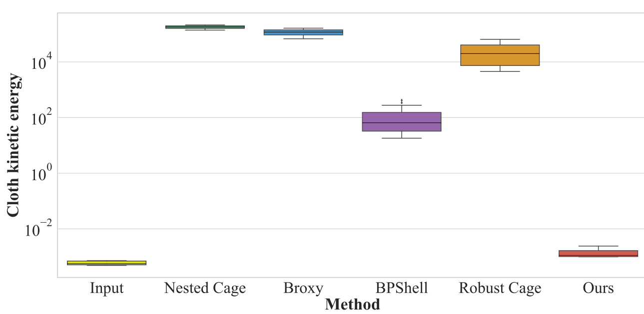


Fig. 19. Quantitative stability analysis of cloth simulations. Box plots show per-frame cloth kinetic energy measured over Frames 100–120 (log scale), corresponding to the regime illustrated in Fig. 18. PR-Cage exhibits lower kinetic energy compared to the alternative proxies, with values approaching the lower-bound baseline provided by the original mesh.

5 Applications

5.1 High-Fidelity Physics Proxies for Contact-Rich Cloth Simulation

We evaluate our cages as collision proxies for contact-rich cloth simulation using the cloth simulator in Blender [Blender Foundation 2024] , aiming to substantially reduce contact complexity while retaining the geometric support needed for stable interactions at sharp features. In these settings, inadequate support near corners and feature transitions often causes local collapse or premature sliding, even when a proxy appears visually acceptable. Many existing cages are derived from volumetric or distance-based representations; although robust, they tend to smooth sharp transitions. This "rounding" produces inconsistent contact normals near feature boundaries, undermining cloth–rigid support contacts and accumulating errors over time.

In contrast, PR-Cage yields strictly enclosing, tightly fitted, self-intersection-free cages with low triangle counts. By aggressively simplifying planar regions while keeping feature-critical vertices nearly fixed, our cages preserve faithful geometric support precisely where contact is most sensitive. In our pipeline, collisions are resolved against this coarse cage while cloth dynamics are simulated at high resolution. Since the cage strictly encloses the input, cloth–proxy interactions remain conservative with respect to the underlying surface, preventing penetration relative to the original model.

As shown in Fig. 18 (middle row), competing proxies become unstable as early as Frame 30, where insufficient support at sharp features triggers sliding. By Frame 120 (bottom row), only our cage and the original mesh allow the cloth to settle into a quasi-static configuration. This is corroborated by the low and stable kinetic energy over Frames 100–120 (Fig. 19). Our method maintains consistently low kinetic energy, outperforming alternatives by orders of magnitude, which exhibit large fluctuations indicative of persistent instability. While the original mesh provides a lower-bound reference for stability, PR-Cage achieves comparable behavior at a fraction of the computational cost. Although slow drift may eventually induce gradual sliding over very long horizons, our results show that PR-Cage provides the structural fidelity required for high-quality short-term contact modeling.

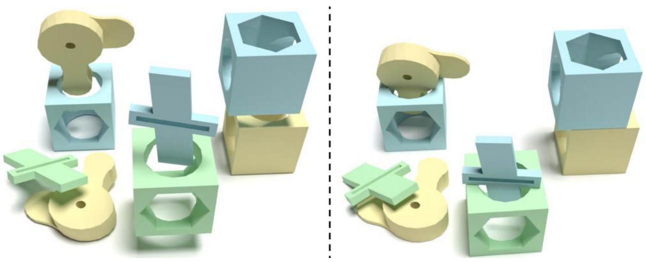
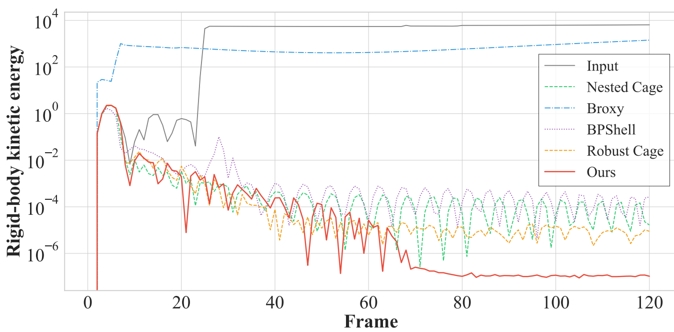
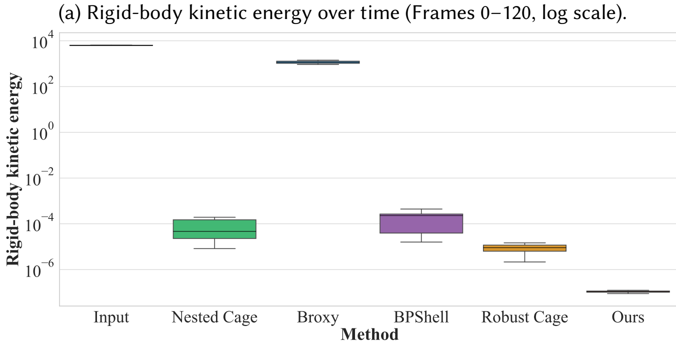


Fig. 20. Contact-rich rigid-body simulation scenario. **Left:** the initial configuration, where multiple interlocking parts form persistent contacts under gravity. **Right:** the quasi-static configuration obtained using our cage-based proxy under identical simulation settings.



(a) Rigid-body kinetic energy over time (Frames 0–120, log scale).



(b) Rigid-body kinetic energy statistics in the quasi-static regime (Frames 100–120).

Fig. 21. Quantitative stability analysis of rigid-body contact simulations. **(a)** Per-frame rigid-body kinetic energy over Frames 1–120 (log scale), revealing the accumulation of micro-jitter under persistent contact. **(b)** Box plots reporting per-frame rigid-body kinetic energy measured over Frames 100–120, corresponding to the quasi-static regime. Lower values indicate more stable contact.

5.2 Stable Contact Proxies for Rigid-Body Simulation

Beyond deformable cloth interactions, we also evaluate PR-Cage in contact-rich rigid-body simulations in Blender [Blender Foundation 2024]. In particular, we use our cages as collision proxies for thin, non-convex mechanical parts in an interlocking stacking scenario (Fig. 20). Such configurations feature persistent contacts, sharp feature interactions, and narrow supporting regions, and expose a common dilemma: accurate mechanical interlocking requires preserving sharp edges and concavities, yet high geometric complexity often destabilizes contact solvers.

Under identical solver settings, simulations driven by the original high-resolution meshes often exhibit instability due to fragmented contact manifolds. Competing cage proxies may alleviate complexity but can still introduce inconsistent contact geometry near sharp features, leading to jitter or artificial sliding in persistent-contact scenarios. In contrast, our strictly enclosing cages provide a compact yet faithful proxy that preserves feature-critical geometry while maintaining a clean and stable contact representation. This yields coherent contact manifolds that reduce numerical jitter without sacrificing physical realism.

We quantify stability via temporal kinetic energy statistics (Fig. 21). Systems simulated with PR-Cage rapidly decay to a near-zero resting state, whereas competing proxies exhibit sustained energy oscillations indicative of persistent contact jitter. These results, together with the supplementary videos, confirm that the improved stability arises from the intrinsic geometric consistency of our proxies rather than solver-specific tuning.

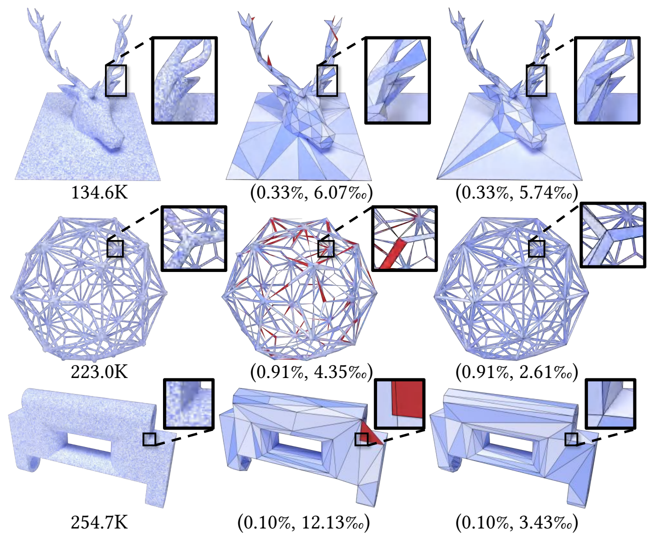


Fig. 22. Comparison of mesh simplification results between classical QEM and our constrained approach. **Left:** Input meshes with their respective triangle counts. **Middle:** Results produced by classical QEM simplification; triangles with inverted orientations are highlighted in red. **Right:** Results produced by our method with identical target face counts. The parentheses denote the remaining face ratio and the relative Hausdorff distance (in ‰) relative to the input.

5.3 Constrained Mesh Simplification

While our primary pipeline restricts decimation between the input and an *outer* τ_{outer}-offset surface to guarantee strict enclosure, the framework naturally generalizes to high-fidelity mesh simplification by expanding the feasible region. Specifically, we redefine the slab constraint in Eq. (7) as a bilateral tolerance zone:

$$\beta_k \tau_{\text{inner}} - d(\mathbf{c}_f, \mathcal{M}) \le \mathbf{n}_f^{\top}(\mathbf{p}_e - \mathbf{c}_f) \le \beta_k \tau_{\text{outer}} - d(\mathbf{c}_f, \mathcal{M}) \quad (9)$$

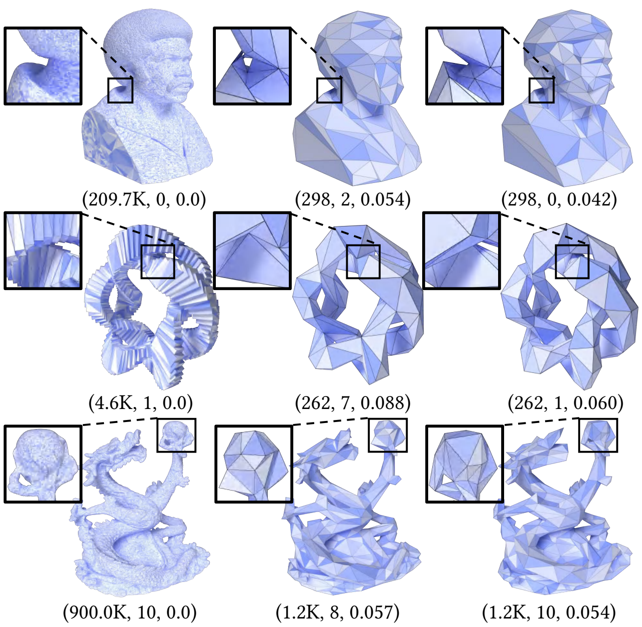


Fig. 23. Comparison of low-poly mesh generation between RoLoPM and our method. **Left:** Input meshes. **Middle:** RoLoPM results showing topological inconsistencies, such as the altered genus in the second row (from 1 to 7). **Right:** Our results preserving input topology and silhouettes. The tuple below each example denotes ($|F|$, genus, silhouette deviation).

where $\tau_{\text{inner}} < 0$, $\tau_{\text{outer}} > 0$, β_k follows Eq. (6), and $d(\mathbf{c}_f, \mathcal{M})$ denotes the signed distance from the face centroid $\mathbf{c}_f$ to the input mesh $\mathcal{M}$. This relaxation permits vertex movement in both inward and outward directions while adhering to a geometric fidelity bound. As demonstrated in Fig. 22, our approach effectively eliminates the surface inversions and topological artifacts (highlighted in red) that frequently occur with unconstrained QEM.

5.4 Low-Poly Mesh Generation

Low-poly modeling shifts the objective from error minimization to stylized, piecewise-planar abstraction while maintaining the visual identity of the input. We achieve this by extending the bilateral band from Sec. 5.3 with an asymmetric slab constraint. Specifically, we first compute a coarse reference wrap $\mathcal{M}'$ using alpha-wrap method (scale $\alpha = l/10$, offset $\varsigma = l/500$) and adapt the constraint as:

$$\beta_k \tau_{\text{inner}} - d(\mathbf{c}_f, \mathcal{M}) \leq \mathbf{n}_f^\top (\mathbf{p}_e - \mathbf{c}_f) \leq \delta_f, \tag{10}$$

where $\delta_f = \min(d(\mathbf{c}_f, \mathcal{M}'), \beta_k \tau_{\text{outer}})$. Setting a tight τ_{inner} anchors convex features and corners, preventing them from receding inward. Simultaneously, the adaptive upper bound δ_f provides the optimizer with the freedom to "fill in" concave regions. As decimation progresses, fine-scale concave details are absorbed into large facets while dominant silhouettes are preserved, resulting in a clean, structural representation. Compared to RoLoPM [Chen et al. 2023], which frequently produce topological artifacts such as altered genus or merged components (Fig. 23, middle), our method strictly preserves the topology consistency. As quantified in Fig. 23, our low-poly meshes maintain topological correctness and lower multi-view silhouette deviation even when reducing the face count by two to three orders of magnitude.

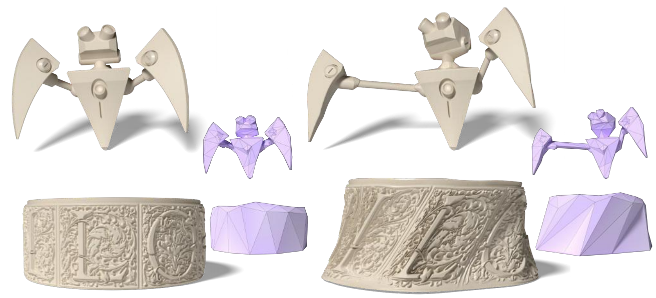


Fig. 24. Application in cage-based deformation. The top row illustrates a mechanical character, and the bottom row displays a high-resolution cylindrical relief. **Left pair**: Input meshes and automatically generated cages (purple) in their rest poses. **Right pair**: Deformed configurations achieved by manipulating the control cages. High-frequency surface details and structural integrity are maintained throughout the deformation process.

5.5 Cage-based Deformation

Cage-based deformation is a fundamental technique where a low-dimensional control structure drives the transformation of a high-resolution mesh. Our automatically generated cages serve as efficient proxies for this pipeline, providing a high-quality alternative to manually authored structures. We evaluate our method by integrating the cages into CageLab [Casti et al. 2018] and binding the input geometry using Mean Value Coordinates (MVC) [Ju et al. 2005]. Fig. 24 demonstrates the versatility of PR-Cage across diverse deformation scenarios. Our method effectively captures the articulated structures of mechanical characters while ensuring smooth deformation transfer for high-resolution reliefs.

6 Limitations

Despite its versatility and robustness, our framework is subject to several limitations that suggest directions for future work:

(1) When geometric features are in extreme proximity (e.g., extremely narrow gaps), the outward vertex displacement required for strict enclosure can induce local self-intersections. In such tightly spaced regions, the feasible space defined by our constraints may become empty, causing the constrained decimation to stall. A potential workaround is to pre-process the input with a watertight wrap surface, though this may come at the cost of losing fine-scale geometric fidelity.
(2) Our method inherits the greedy nature of QEM-based simplification, where edge collapses are performed sequentially based on a priority queue. Because each operation modifies the local error quadrics and the available feasible region for subsequent steps, our framework does not explicitly enforce symmetry. Consequently, the resulting cage may exhibit asymmetric connectivity even when the input model is perfectly symmetric.
(3) The guarantee of strict enclosure in our formulation relies on a well-defined interior–exterior orientation. This necessitates that the input surface be a closed, manifold mesh. For open meshes or

surfaces with boundaries, the absence of a consistent orientation prevents us from formulating a rigorous enclosure relationship, limiting the applicability of our method to watertight geometries.

7 Conclusion

We have presented PR-Cage, a nested optimization framework for automated cage generation that effectively balances geometric simplicity with enclosure tightness. The core of our approach is a progressive feasibility relaxation strategy, which uses a staircase schedule for the thickness parameter to preserve prominent shape features while enabling aggressive simplification. By incorporating linear inequality constraints into a constrained QEM formulation, we ensure strict enclosure and topological integrity through efficient atomic operations.

Extensive evaluations on the ABC and Thingi10K datasets demonstrate that PR-Cage consistently outperforms existing methods in shape fidelity and computational efficiency. We have also showcased its practical utility in diverse downstream applications, including contact-rich physics simulations, high-fidelity mesh simplification, and cage-based deformation. While current limitations include a requirement for closed manifold meshes and challenges with extreme geometric proximity, our framework provides a robust foundation for generating high-quality proxies in complex geometry processing pipelines. Future research will focus on improving triangle quality and enabling more intuitive user editability.

Acknowledgments

The authors thank the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Natural Science Foundation of China (Grants U23A20312, 62272277, and 61872121) and the Natural Science Foundation of Shandong Province (Grant ZR2025MS986).

References

Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. 2009. Spatial Deformation Transfer. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, Eitan Grinspun and Jessica Hodgins (Eds.). ACM SIGGRAPH / Eurographics Association.

Blender Foundation. 2024. *Blender*. https://www.blender.org

Stéphane Calderon and Tamy Boubekeur. 2017. Bounding Proxies for Shape Approximation. *ACM Trans. Graph.* 36, 4, Article 57 (July 2017), 13 pages.

Sara Casti, Fabrizio Corda, Marco Livesu, and Riccardo Scateni. 2018. CageLab: an Interactive Tool for Cage-Based Deformations. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association.

Sara Casti, Marco Livesu, Nicolas Mellado, Nadine Abu Rumman, Riccardo Scateni, Loïc Barthe, and Enrico Puppo. 2019. Skeleton based cage generation guided by harmonic fields. *Computers & Graphics* 81 (2019), 140–151.

Xue Chen and Jieqing Feng. 2014. Adaptive skeleton-driven cages for mesh sequences. *Computer Animation and Virtual Worlds* 25, 3-4 (2014), 445–453.

Zhen Chen, Zherong Pan, Kui Wu, Etienne Vouga, and Xifeng Gao. 2023. Robust Low-Poly Meshing for General 3D Models. *ACM Trans. Graph.* 42, 4, Article 119 (July 2023), 20 pages.

Jiayi Dai, Yiqun Wang, and Dong-Ming Yan. 2024. Feature-preserving shrink wrapping with adaptive alpha. *Computer Aided Geometric Design* 111 (2024), 102321.

Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. 2011. Automatic Cage Building with Quadric Error Metrics. *Journal of Computer Science and Technology* 26, 3 (01 May 2011), 538–547.

Michal Edelstein, Hsueh-Ti Derek Liu, and Mirela Ben-Chen. 2025. CageNet: A Meta-Framework for Learning on Wild Meshes. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 59, 11 pages.

Christer Ericson. 2005. Chapter 10 - GPU-assisted Collision Detection. In *Real-Time Collision Detection*, Christer Ericson (Ed.). Morgan Kaufmann, San Francisco, 413–426.

Noura Faraj, Jean-Marc Thiery, and Tamy Boubekeur. 2012. VoxMorph: 3-scale freeform deformation of large voxel grids. *Computers & Graphics* 36, 5 (2012), 562–568. Shape Modeling International (SMI) Conference 2012.

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–216.

D. Goldfarb and A. Idnani. 1983. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming* 27, 1 (Sep 1983), 1–33.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen: A C++ Template Library for Linear Algebra: Matrices, Vectors, Numerical Solvers, and Related Algorithms. http://eigen.tuxfamily.org.

Stefan Gumhold, Pavel Borodin, and Reinhard Klein. 2003. Intersection free simplification. *International Journal of Shape Modeling* 9, 02 (2003), 155–176.

Jia-Peng Guo, Wen-Xiang Zhang, Chunyang Ye, and Xiao-Ming Fu. 2024. Robust Coarse Cage Construction With Small Approximation Errors. *IEEE Transactions on Visualization and Computer Graphics* 30, 7 (2024), 4234–4245.

Hugues Hoppe. 1996. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 99–108.

Jingwei Huang, Hao Su, and Leonidas J. Guibas. 2018. Robust Watertight Manifold Surface Generation Method for ShapeNet Models. *ArXiv* abs/1802.01698 (2018).

Jingwei Huang, Yichao Zhou, and Leonidas J. Guibas. 2020. ManifoldPlus: A Robust and Scalable Watertight Manifold Surface Generation Method for Triangle Soups. *ArXiv* abs/2005.11621 (2020).

Alec Jacobson, Daniele Panozzo, et al. 2018. Libigl: A Simple C++ Geometry Processing Library. https://libigl.github.io/.

Zhongshi Jiang, Teseo Schneider, Denis Zorin, and Daniele Panozzo. 2020. Bijective projection in a shell. *ACM Trans. Graph.* 39, 6, Article 247 (Nov. 2020), 18 pages.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean Value Coordinates for Closed Triangular Meshes. *ACM Trans. Graph.* 24, 3 (July 2005), 561–566.

Tao Ju, Qian-Yi Zhou, Michiel van de Panne, Daniel Cohen-Or, and Ulrich Neumann. 2008. Reusable skinning templates using cage-based deformations. *ACM Trans. Graph.* 27, 5, Article 122 (Dec. 2008).

Franjo Juretić and Norbert Putz. 2012. A Surface-Wrapping Algorithm with Hole Detection Based on the Heat Diffusion Equation. In *Proceedings of the 20th International Meshing Roundtable*, William Roshan Quadros (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 405–418.

Tero Karras. 2012. Maximizing Parallelism in the Construction of BVHs, Octrees, and K-d Trees. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. Eurographics Association, 33–37.

Timothy L. Kay and James T. Kajiya. 1986. Ray tracing complex scenes. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 269–278.

Leif P. Kobbelt, Jens Vorsatz, and Ulf and Labsik. 1999. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum* 18, 3 (1999), 119–130.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Binh Huy Le and Zhigang Deng. 2017. Interactive cage generation for mesh deformation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (San Francisco, California) *(I3D '17)*. Association for Computing Machinery, New York, NY, USA, Article 3, 9 pages.

Y. K. Lee, Chin K. Lim, Hamid Ghazialam, Harsh Vardhan, and Erling Eklund. 2010. Surface mesh generation for dirty geometries by the Cartesian shrink-wrapping technique. *Eng. with Comput.* 26, 4 (Aug. 2010), 377–390.

Stephanie Wenxin Liu, Michael Fischer, Paul D. Yoo, and Tobias Ritschel. 2024. Neural Bounding. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 98, 10 pages.

David Martineau, Jeremy Gould, and Jacques Papper. 2016. An Integrated Framework for Wrapping and Mesh Generation of Complex Geometries. In *Proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016)*. Institute of Structural Analysis and Antiseismic Research School of Civil Engineering National Technical University of Athens (NTUA) Greece, Crete Island, Greece, 6938–6954.

Daniel Meister, Shinji Ogaki, Carsten Benthin, Michael J. Doyle, Michael Guthe, and Jiří Bittner. 2021. A Survey on Bounding Volume Hierarchies for Ray Tracing. *Computer Graphics Forum* 40, 2 (2021), 683–712.

Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. 2009. Preserving topology and elasticity for embedded deformable models. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH 09)*. Association for Computing Machinery, New York, NY, USA, Article 52, 9 pages.

Aleksander Płocharski, Joanna Porter-Sobieraj, Andrzej Lamecki, Tomasz Herman, and Andrzej Uszakow. 2024. Skeleton based tetrahedralization of surface meshes. *Computer Aided Geometric Design* 111 (2024), 102317.

Cédric Portaneri, Mael Rouxel-Labbé, Michael Hemmer, David Cohen-Steiner, and Pierre Alliez. 2022. Alpha Wrapping with an Offset. *ACM Transactions on Graphics (TOG)* 41, 4, Article 127 (July 2022), 22 pages.

Hao Qiu, Wentao Liao, and Renjie Chen. 2024. High-Quality Cage Generation Based on SDF. *Pacific Graphics Conference Papers and Posters* (2024).

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested cages. *ACM Trans. Graph.* 34, 6, Article 170 (Nov. 2015), 14 pages.

Pedro V. Sander, Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. 2000. Silhouette clipping. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 327–334.

Rohan Sawhney. 2021. *FCPW: Fastest Closest Points in the West*.

Philip J. Schneider and David H. Everly. 2003. CHAPTER 9 - GEOMETRIC PRIMITIVES IN 3D. In *Geometric Tools for Computer Graphics*, PHILIP J. SCHNEIDER and DAVID H. EBERLY (Eds.). Morgan Kaufmann, San Francisco, 325–364.

Silvia Sellán, Jack Luong, Leticia Mattos Da Silva, Aravind Ramakrishnan, Yuchuan Yang, and Alec Jacobson. 2023. Breaking Good: Fracture Modes for Realtime Destruction. *ACM Trans. Graph.* 42, 1, Article 10 (March 2023), 12 pages.

Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 896–904.

D. Ströter, J. M. Thiery, K. Hormann, J. Chen, Q. Chang, S. Besler, J. S. Mueller-Roemer, T. Boubekeur, A. Stork, and D. W. Fellner. 2024. A Survey on Cage-based Deformation of 3D Models. *Computer Graphics Forum* 43, 2 (2024), e15060.

David A. Stuart, Joshua A. Levine, Ben Jones, and Adam W. Bargteil. 2013. Automatic Construction of Coarse, High-Quality Tetrahedralizations that Enclose and Approximate Surfaces for Animation. In *Proceedings of Motion on Games* (Dublin 2, Ireland) *(MIG '13)*. Association for Computing Machinery, New York, NY, USA, 213–222.

Yifei Tong, Runze Tian, Xiao Han, Dingyao Liu, Fenggen Yu, and Yan Zhang. 2025. CAGE-GS: High-fidelity Cage Based 3D Gaussian Splatting Deformation. arXiv:2504.12800 [cs.GR] https://arxiv.org/abs/2504.12800

Chuhua Xian, Guiqing Li, and Yunhui Xiong. 2015. Efficient and effective cage generation by region decomposition. *Computer Animation and Virtual Worlds* 26, 2 (2015), 173–184.

Chuhua Xian, Hongwei Lin, and Shuming Gao. 2009. Automatic generation of coarse bounding cages from dense meshes. In *2009 IEEE International Conference on Shape Modeling and Applications*. 21–27.

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

Xiaosong Yang, Jian Chang, Richard Southern, and Jian J. Zhang. 2013. Automatic cage construction for retargeted muscle fitting. *Vis. Comput.* 29, 5 (May 2013), 369–380.

Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2020. Neural Cages for Detail-Preserving 3D Deformations. In *CVPR*.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

Daniel Zint, Zhouyuan Chen, Yifei Zhu, Denis Zorin, Teseo Schneider, and Daniele Panozzo. 2025. Topological Offsets. *ACM Trans. Graph.* 44, 4, Article 144 (July 2025), 19 pages.