www.cadanda.com
Computer-Aided Design AND APPLICATIONS

A Medial Axis Transformation-based Method for Cage Generation

Li Cao[1], Zhan Zhang[2], Yike Xu[3]

[1]Hefei University of Technology, lcao@hfut.edu.cn
[2]Hefei University of Technology, 2023110576@mail.hfut.edu.cn
[3]Hefei University of Technology, yecoxu@mail.hfut.edu.cn
Corresponding author: Li Cao, lcao@hfut.edu.cn

Abstract. In computer graphics, the bounding box plays a critical role across diverse applications spanning mesh-based rendering, shape deformation, physical simulation and collision detection. This study introduces an innovative methodology employing medial axis transformation (MAT) to guide the generation of optimized bounding volumes. Commencing with a 3D mesh, the skeletal architecture is systematically extracted through MAT technology. This approach facilitates precise identification of key skeletal nodes via minimal user interaction. The principal contribution of this work includes: (1) the construction of skeletal frameworks utilizing intrinsic medial axis transformation; (2) the establishment of a node selection paradigm grounded in anatomical structural analysis. Experimental validation demonstrates that the generated coarse bounding volumes effectively preserve both topological integrity and essential geometric characteristics of original meshes. The methodology satisfies various application requirements including high-fidelity mesh rendering, shape transformation accuracy, and collision detection reliability.

Keywords: Computational geometry, Mesh editing, Bounding box, Medial axis transformation, Automation, Cage generation
DOI: https://doi.org/10.14733/cadaps.2026.515-531

1 INTRODUCTION

In computer graphics, processing complex geometric meshes for animation, physical simulation, and rendering requires addressing critical tasks such as mesh deformation, collision detection or avoidance, and physics-based computations [1]. While high-resolution meshes are widely employed to capture intricate geometric details for enhanced application fidelity, direct computation on such meshes often incurs prohibitive computational complexity due to their reliance on sophisticated nonlinear optimization. This challenge intensifies with increasing demands for interactive or even real-time performance, particularly in scenarios that require frequent geometric edits.

To reconcile precision with efficiency, researchers have developed geometric proxy structures to reduce computational overhead. Among these, cages compact bounding representations of the underlying geometries have emerged as a prominent solution. These proxies not only preserve essential shape characteristics, but also demonstrate broad utility across animation, physical simulation, and rendering workflows.

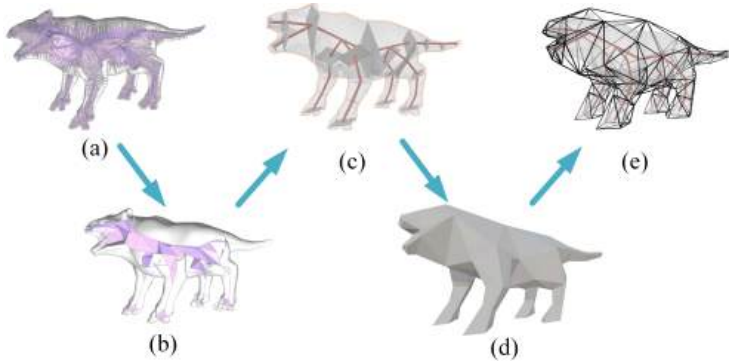

Figure 1: This pipeline initiates with a digital character mesh and its MAT. (a) The MAT undergoes simplification, during which users manually designate bending nodes. (b) At these nodes, cross-sections interfacing with the mesh are generated and symmetrically processed. (c) These cross-sections are then interconnected to construct a coarse cage. (d) which subsequently undergoes inflation to fully encapsulate the input mesh. (e) Each stage ensures geometric coherence while preserving deformation-aware topological features.

For closed, nonintersecting manifold triangle meshes, we propose a robust automated cage generation framework satisfying four criteria [2]: a) Tightly enclose the original mesh. b) Position control nodes near regions prone to bending or deformation (e.g., joints, thin protrusions). c) Maintain coarse structures for intuitive manipulation while maintaining sufficient detail for feature preservation. d) Inherit and enforce geometric symmetries inherent to the input mesh. By providing a rough, yet compact bounding structure, this approach can reduce computational costs and manual workload during mesh deformation while preserving the mesh's surface features, granting artists enhanced control. In physical simulations, employing coarse meshes for task processing can significantly accelerate simulation speed by reducing the equation solving time and collision detection duration. Moreover, the low approximation error helps minimize the discrepancy between results obtained from coarse and fine meshes.Indeed, constructing an effective cage to approximate complex shapes presents significant challenges, primarily manifested in two aspects:

(a) Algorithmic Challenges Designing robust algorithms for arbitrary input meshes is exceptionally difficult due to complex topologies and geometric details. Automatic cage generation methods often rely solely on geometric approaches, neglecting motion semantics. This causes critical failures in identifying articulation points (e.g., knees/elbows) for characters in standard T-poses.

(b) Conflicting requirements The core contradiction lies between:First, Reducing cage complexity (computational efficiency).Second, Minimizing approximation error (quality preservation) Bounding box cages satisfy basic intersection/animation constraints but fail to resolve this trade-off, limiting practical utility.

The key to addressing this challenge lies in striking a balance that advances computational efficiency and ease of manipulation while preserving maximum accuracy in capturing the crucial geometric features and topological characteristics of the original mesh. With this goal in mind, we propose a medial-axis-driven approach that empowers users to intuitively control the topology of coarse bounding boxes through an exceptionally simple interface. This method effectively resolves shape-aware issues while fulfilling all aforementioned requirements. Its core innovation lies in generating a mesh skeleton through medial axis transformation and selecting articulation points along this skeletal structure.

Specifically, this methodology initiates with the generation of a medial-axis-derived skeletal structure within the mesh. The joint node skeleton is obtained by aggregating with the Medial Axis Transform points. This technology has established a novel animation pipeline. Upon selection of these pivotal points, the system autonomously generates corresponding character shape segmentation and constructs a topologically consistent coarse cage structure. The technique establishing a novel animation pipeline that optimizes both production efficiency and artistic control. This holistic strategy enhances workflow productivity while preserving organic motion characteristics, ultimately enabling more nuanced and intentional artistic expression in animated works.

In summary, the major contributions of this work are:

- Integrates MAT with curvature-driven adaptive sampling for topology-aware cage initialization.
- Skeleton points and radii are predicted through convex combination learning of geometric transformations, with the mesh's MAT used as input to derive its joint node information.
- A fully optimized and efficient pipeline is implemented for the generation of cages.

2 RELATED WORK

2.1 The Application of Cage

Scalas et al. [3] developed an in-VR modeling system using cage-based deformation, enhanced through semantically enriched meshes with annotated functional regions. This approach enables direct manipulation of meaningful object components within the virtual environment, eliminating disruptive tool-switching cycles while maintaining structured control over complex meshes.In parametric CAD meshes, interference between components is avoided when adjusting the shape through cage.

Yuan et al. [4] in Finite Element Analysis (FEA), Cage is used as a surrogate mesh for rapid collision response. Ren et al. [5] introduce CSG-Stump, a three-layer reformulation of the classic CSG-Tree for 3D shape representation. It inherits the compact, interpretable, and editable nature of CSG-Tree while being more learning-friendly. The authors also propose CSG-Stump Net, an unsupervised end-to-end network for joint primitive detection and CSG-Stump estimation. Guo et al. [6] present ComplexGen, a framework for CAD reconstruction from point clouds. It represents CAD models as boundary representation (B-Rep) chain complexes and uses a hybrid approach of deep learning and optimization to recover construct and structured CAD models.

Li et al. [7] present SED-Net, a novel neural network with a two-branch structure for primitive fitting of point clouds. It jointly detects parametric surfaces (including B-spline patches) and edges, assembling them into a regularized and seamless CAD model. The network utilizes a two-stage feature fusion mechanism to fully leverage type, edge, and geometric features, achieving accurate and robust primitive segmentation.

2.2 Medial Axis Simplification Methods

The MAT constitutes a geometric descriptor in computational shape representation in Figure 2, formally defined as the locus of centers of maximal inscribed spheres/circles within a 3D/2D object.

The Scale Axis Transform (SAT) [8] prunes spurious spikes by scaling all intermediate spheres by $s > 1$, then removing those contained within others. This process may fill narrow gaps, potentially altering the input's homotopy type. Surviving spheres are rescaled by $1/s$ to form the final approximation. A key limitation is the non-intuitive control via parameter s: excessive scaling ($s \gg 1$) sacrifices geometric features for MAT compactness, with no direct correlation to preserved vertex count.

Q-MAT[9], a novel approach leveraging quadratic error minimization to compute structurally simple, geometrically precise, and compact MAT representations. This method introduces: (1) a new error metric for MAT approximation, and (2) a quantitative characterization of unstable MAT branches, integrated into an extended quadric error metric (QEM) framework. Q-MAT efficiently eliminates insignificant unstable branches

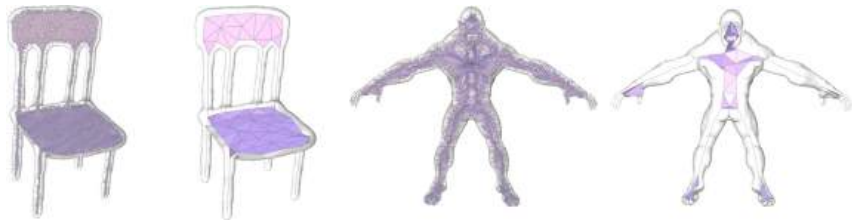

Figure 2: The left figure is Non-humanoid mesh's MAT, while the right figure is humanoid mesh's MAT.

while producing accurate piecewise-linear MAT approximations. Comprehensive validation demonstrates its superiority in speed and accuracy over conventional MAT computation methods, addressing long-standing challenges in balancing structural simplicity with geometric fidelity.

2.3 Cage Generation Methods

Skeletons can be generated manually or automatically [10], and joints are intuitively posed to define posture. By coupling the skeleton with a deformation lattice, motion is conveyed at the joints while surface detail is preserved on the lattice [11]. The approach is intrinsically local: regions devoid of bones are left immobile, preventing full-model deformation analogous to a cage.

To avoid the tedious construction of cages, [12] compresses the deformation space into a linear subspace and lets users place point or region handles. Speed comes at a price: aggressive dimension reduction allows a single point handle to affect disconnected components, and the absence of semantic awareness forces the user to sprinkle additional handles, turning cage-free into a manual refinement loop. Large-angle rotations further reveal volume collapse due to the embedded linear-blend skinning.

The first automatic closed cages were obtained by folding edges of a progressive mesh [13]. The resulting hull, however, drifts away from the surface and self-intersects under vigorous decimation. Shen et al. [14] replace the explicit hull with an implicit surface fitted in a constrained least-squares sense, yet smoothing obliterates detail, and intersections remain unchecked. Ben-Chen et al. [15] iteratively offset and simplify the mesh until a user-defined face count is reached, but accumulated drift produces self-penetrating elements. Deng et al. [16] prioritize collapses via an error metric and add a fidelity pair to preserve shape and triangle quality; thin protrusions nevertheless disappear and global intersection-free status is still not guaranteed. Sacht et al. [17] stack nested cages-fine layers hug the surface while coarse layers drive large motion. Construction time grows linearly with the number of layers, and the final hierarchy ignores motion semantics: joints are not explicitly rewarded, so deformable regions may be absorbed into overly stiff coarse cells.

Xian et al. [18] voxelize a Principal Component Analysis (PCA)-oriented bounding box and retain surface-intersecting voxels as cage vertices. The outcome is intersection-free, but its topology is locked to voxel resolution and PCA alignment discards object-intrinsic symmetries, necessitating manual mirroring. Nesme et al. [19] embed the model in an adaptive octree of hexahedra; the grid remains conservatively thick around thin features and symmetry is never encoded during subdivision. Guo et al. [20] guarantee a manifold, intersection-free cage via a two-phase shrink-and-simplify process. Control vertices are therefore seldom located at deformation hot-spots, and symmetry is enforced only as a post-process rather than a hard constraint.

Chen et al. [21] sketch a skeleton on the silhouette, extract cross-sections at joints, and loft offset contours into a cage. Le et al. [22] allow part decomposition via cutting planes, but manual sketching accumulates error and yields asymmetric or overly dense cages. Calderon et al. [23] merge voxelization with mesh simplification for fast cage creation; PCA alignment again corrupts mirroring accuracy and sharp ridges are rounded away. Casti et al. [24] automatically detect bending points from thickness variations and use a harmonic field to position orthogonal cross-sections. Thickness estimation fails on open sheets, however, and the harmonic field

is sensitive to branch ordering, frequently misaligning sections and demanding user cleanup.

In conclusion, the existing methods have respectively addressed the issues of speed, closeness or automation, but there is no solution that can simultaneously guarantee all these requirements. To address the shortcomings of the above-mentioned methods, the proposed method achieves the inheritance of the inherent geometric symmetry of the input mesh, the placement of control nodes at the joints, and strives to achieve a rough structure for operational deformation while ensuring details.

3 METHOD DESCRIPTION

The pipeline of the proposed method is illustrated in Figure 1 and summarized in the Introduction. In this section, each step is elaborated on in detail. The input consists of a triangular mesh M. First, the central axis of the mesh is calculated, and obtain the central axis skeleton S by aggregating the central axis points in Figure 3. Nodes in S with more than two connected branches are referred to as branch nodes, while nodes with only one connected branch are termed leaf nodes. Roughly speaking, the cage will comprise multiple tubular structures, each corresponding to a branch of the medial axis skeleton, connected at polyhedral joints surrounding the branch nodes[24]. To simplify the generated results, quadrilateral cross-sections are employed for the polygonal profiles of the tubular structures, which will undergo lateral subdivision at the bending nodes.

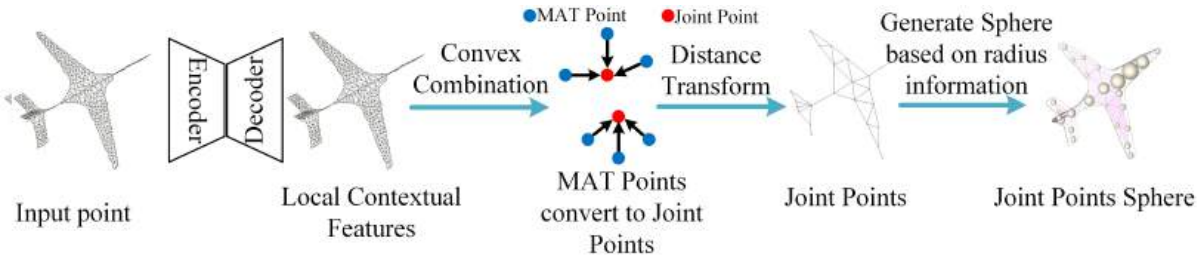

Figure 3: An overview of the pipeline of this training network. Given the MAT information of the mesh as input, skeleton points and their radii are predicted through convex combination learning of geometric transformations to obtain the information of the mesh's joint nodes.

3.1 Convert to Generate Joint Nodes

Figure 4 shows the principle of the dataset. Given the MAT information points generated from the sampling point cloud of the shape-Net mesh and the joint node information of the mesh that manually marked as input, select the part where the central axis radius of the mesh changes sharply as its joint node. The Q-mat output is manually annotated to obtain ground-truth critical points, which are then used to supervise training. Following [25], we adopt the Point2Skeleton framework to regress the final critical-point locations and radii.

Inspired by [26], where skeletal points are conceptualized as local centroids of surface point clusters, skeletal points are generated through convex combinations of input points. Given K input points $\{\mathbf{p}_i\} \in \mathbb{R}^3$, then aggregate them to produce N skeletal spheres parameterized by center coordinates $\mathbf{C}$ and radii R. Specifically, a PointNet++ [27] encoder extracts sampled points P with associated contextual features F. A dedicated loss function [25] jointly optimizes skeletal coordinates and radii by minimizing the reconstruction error:

$$L_p = \sum_{p\in\{p_i\}} \left(\min_{c\in\{c_i\}} \|p-c\|_2 - r(c_p^{\min}) \right) + \sum_{c\in\{c_i\}} \left(\min_{p\in\{p_i\}} \|c-p\|_2 - r(c) \right) \tag{1}$$

where $\{\mathbf{c}_i\}$ represents the ground truth skeletal points, $\{\mathbf{p}_i\}$ the input points, $r(c)$ the radius of a skeletal point c, and $c_p^{\min}$ the closest skeletal point to the input point p. The first term constrains each input point to

be located on the surface of its closest skeletal sphere, while the second term encourages each skeletal sphere to touch its closest input point.

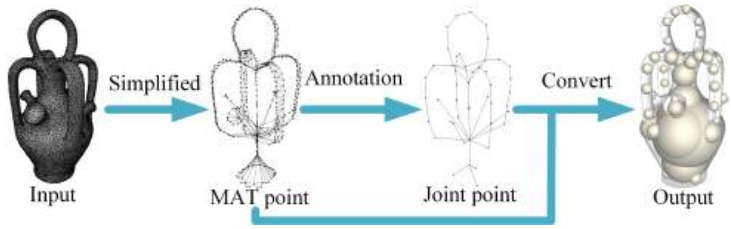

Figure 4: The MAT information is computed based on a mesh. Subsequently, joint nodes are identified and marked on the MAT, and these identified joint nodes are treated as the final dataset.

For a given mesh M, first compute its medial axis using the method described in [9]. The generated medial axis points are then aggregated to construct the medial axis skeletal S of M. Similar to [24], the medial axis skeleton S provides critical structural information, including the number of branch structures within the mesh topology and their interconnection patterns. This skeletal representation serves as the foundation for subsequent cage generation and deformation-aware processing.

3.2 Generation of Mesh Cross-sectional

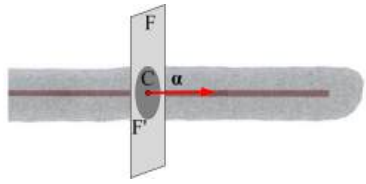

Figure 5: At joint nodes along the MAT skeleton, planes perpendicular to the MAT skeleton are constructed to extract cross-sectional profiles of the mesh. These planes intersect the input geometry to generate topologically consistent cross-sections, which serve as foundational elements for subsequent cage construction.

The joints nodes generated in the previous step C will induce a cutting plane F, where α denotes the directional vector of the medial axis skeleton at C in Figure 5. The cutting plane F is defined as the plane passing through C with normal vector α, which intersects the mesh M to produce a cross-section F'. When a node has $k \geq 3$ incident branches, we set α to the direction of the minimum-curvature principal component of the k tangent vectors, computed via PCA of their unit directions. This approach serves as a superior alternative to the cutting planes used in [22]. The method is applied not only to joint nodes but also to leaf nodes and branch nodes, resulting in multiple cross-sections intersecting the mesh.

3.3 Fitting Cross-sectional Patches to Quadrilateral Planes

Cross-sectional patch, for a cutting plane F we obtain a patch $P = M \cap F$, represented as an unordered set of 3-D points $\{p_i\}$ sampled from the intersected triangles. These points form the rows of the point-cloud data matrix $M \in \mathbb{R}^{|P|\times 3}$ used in Eq. (2). Singular value decomposition (SVD) is employed to achieve data compression and dimensionality reduction for cross-sectional patches. Smaller singular values in the SVD spectrum typically correspond to noise or less significant geometric information. By truncating these smaller

singular values, noise is effectively reduced, and data quality is enhanced. Specifically, all points within a cross-sectional patch are projected onto a plane through dimensionality reduction. Extreme boundary points are the four vertices of the 2-D axis-aligned bounding box of the projected points after SVD dimensionality reduction, and the four extreme boundary points of the projected point cloud are identified to fit a quadrilateral plane. This method not only facilitates noise removal and mesh simplification but also supports intuitive visualization of high-dimensional geometric data. Mathematically, a real or complex matrix M can be decomposed into three constituent matrices via SVD:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*} \tag{2}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal (or unitary) matrices, and $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values. The truncated SVD retains only the dominant singular values, ensuring geometric fidelity while suppressing noise-a critical step for generating structurally coherent quadrilateral planes as the basis for cage construction.Implementation Steps:

a) Data preparation: Obtain object cross-sectional patches from the preprocessing stage, represented as a set of 3D points in space.

b) SVD application: Perform SVD on the point cloud's data matrix to identify principal directions and remove noise.

c) Dimensionality reduction: Project the data onto the two most significant principal components derived from the SVD results, effectively reducing dimensionality to a 2D plane.

d) Boundary point extraction: On the 2D plane, locate the four extreme boundary points of the projected point distribution and use them to fit a quadrilateral.

3.4 Symmetry

In digital modeling, most characters are manually modeled only on one half, while the other half is generated by mirroring the mesh across a symmetry plane. Therefore, leveraging such symmetry during cage generation is critical to ensure consistent cage structures for both sides of symmetrical characters.

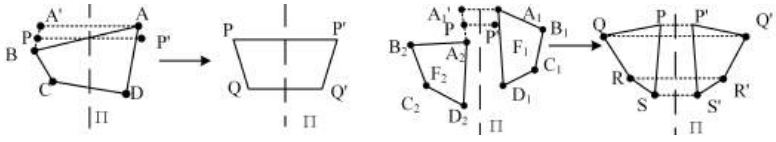

Figure 6: The left figure is Type I symmetric faces, achieving self-symmetry of the cross-section, while the right figure is Type II symmetric faces, achieving symmetry of the cross-section pairs.

The method described in [28] is employed to automatically detect the global symmetry plane and to guide the generation of symmetrical cross-sections. Symmetrical cross-sectional faces are divided into two categories, one is Self-Symmetric Faces: Faces lying directly on the symmetry plane, which lack mirrored counterparts and thus require symmetry with themselves.another is Paired Symmetric Faces: Faces located on both sides of the symmetry plane, which must be mirrored pairwise to ensure geometric consistency.

The strategy for generating Type I symmetric faces is illustrated in Figure 6. When the symmetry plane Π intersects a cross-section, point A on the right side of the symmetry axis is mirrored across Π to generate its symmetric counterpart A'. The closest point B to A' on the original cross-section is identified. The midpoint P of the line segment AB and its mirrored counterpart P' across Π are computed as the symmetrized points for A and B. Similarly, points C and D are processed to obtain their symmetric counterparts Q and Q'. These points are connected to form the quadrilateral $P'PQQ'$, which serves as the symmetrized counterpart of the original quadrilateral $ABCD$. This approach ensures geometric consistency between mirrored regions

while preserving local structural features critical for symmetric cage generation.If multiple vertices yield the same minimal distance, we select the one with the smallest index to guarantee deterministic output.

The strategy for generating Type II symmetric faces is illustrated in Figure 6. Given a symmetry plane Π, all points A_1 on a quadrilateral face F_1 located on one side of Π are mirrored across Π to compute their symmetric counterparts A_1'. Within the four vertices A_2, B_2, C_2, D_2 of the target quadrilateral face F_2 (to be symmetrized with F_1), the vertex A_2 with the smallest Euclidean distance to A_1' is identified. The average position of A_1' and A_2 is calculated and denoted as point P, which becomes a vertex of the final symmetrized face. This procedure is iteratively applied to the remaining vertices to derive points Q, R, and S. These points are then reflected across Π to generate their symmetric counterparts P', Q', R', and S', resulting in two symmetric quadrilateral faces: $PQRS$ and $P'Q'S'R'$.

This method efficiently generates symmetrized quadrilateral cross-sections while preserving geometric coherence across the symmetry plane, thereby enabling the rapid construction of an asymmetric cage suitable for deformation and animation workflows. It is important to note that symmetry detection constitutes merely one step in the cage generation pipeline for symmetric models. For asymmetric models, this step must be omitted since its purpose is to preserve the inherent symmetric semantics of the input model. Consequently, asymmetric models require no additional processing beyond the standard pipeline.

3.5 Coarse Cage Generation

The coarse cage tailored to the input mesh M is constructed by traversing along the medial axis skeleton S from the leaf nodes to the user-specified bending nodes. The quadrilateral faces derived at these nodes (via symmetry-aware cross-section fitting and SVD-based planar approximation) are interconnected to form a low-resolution cage structure that faithfully encapsulates the global topology of M. This initial cage serves as a deformable scaffold for subsequent refinement stages.

A method analogous to [22] is employed to generate tubular structures for the cage. For each pair of adjacent quadrilateral cross-sections, perform Delaunay triangulation on their combined vertex sets and extract triangular faces from the resulting convex hull. This process yields discrete tubular structures composed of triangular faces. Crucially, the triangulation between adjacent quadrilateral pairs remains mutually independent, thereby preventing cumulative distortion propagation along character limbs. Each tubular structure corresponds to a branch of the medial axis skeleton and interconnects at bending nodes.

After constructing all tubular components, they are unified into a construct cage by welding the tubular structures surrounding branch nodes of the medial axis skeleton. At branch nodes, then perform Delaunay triangulation on all quadrilateral faces extending from the node to generate a more complex convex hull. Finally, guided by the connectivity of the medial axis skeleton, all cage components are assembled to form the coarse cage. This hierarchical approach ensures topological consistency while maintaining geometric fidelity to the input mesh. For the generation of rough cages at leaf nodes, we initially construct them inside the model rather than outside. An expansion step is then applied to push the cages outward, ensuring they encompass the target geometry.

3.6 Cage Expansion

Following the generation of the coarse cage, the cage vertices are refined to ensure the cage fully encloses the input mesh. Since all vertices of the initially constructed cage lie precisely on the surface of mesh M, these vertices are iteratively offset outward to guarantee no cage face intersects M in Figure 7. This expansion process is constrained by two key objectives:

- **Minimal normal variation**: The directional changes of triangular face normals are minimized to preserve local smoothness.

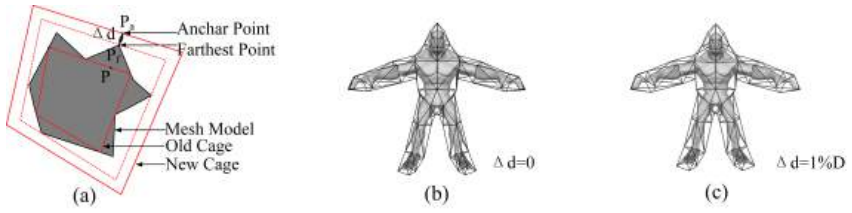

Figure 7: The cage vertices are displaced outward to tightly enclose the input mesh within distance Δd, while preserving triangle normals to maintain structural coherence for deformation in the left. Middle is a tightly bound cage. Right is a loosely bound cage. D represents the diagonal length of the bounding box of the input mesh.

- **Distance control**: The distance between the cage surface and M is regulated through a distance field constraint, as visualized in Figure 7.

First, since the mesh model is partitioned into multiple parts M' by the cross-sections passing through the specified points, it suffices to search for the farthest points only within the mesh surfaces of these respective parts. it identify the vertex $\mathbf{P}_f$ on mesh M' that lies farthest from the triangular face F of the original cage along its normal $\mathbf{n}_t$ (Eq. 3). To obtain the final anchor point $\mathbf{P}_a$, $\mathbf{P}_f$ is displaced along $\mathbf{n}_t$ by a distance Δd (typically set to 0 for tight wrapping). Crucially, this displacement is applied only if $\mathbf{P}_f$ lies in the positive half-space of $\mathbf{n}_t$ (i.e., outside the original cage). If $\mathbf{P}_f$ resides in the negative half-space (inside the cage), the anchor point $\mathbf{P}_a$ is instead set to the centroid $\mathbf{g}_t$ of face F (Eq. 4).

Each triangular face F is then reconstructed by displacing its centroid $\mathbf{g}_t$ along $\mathbf{n}_t$ to $\mathbf{P}_a$ [22]. This process is applied to all faces to generate the refined cage. The refinement ensures the cage closely conforms to M while maintaining geometric regularity.

$$\mathbf{P}_f = \arg\max_{\mathbf{p}\in\mathbf{M}'} \left((\mathbf{p}-\mathbf{g}_t)\cdot\mathbf{n}_t\right) \tag{3}$$

$$\mathbf{P}_a = \begin{cases} \mathbf{P}_f + \Delta d\mathbf{n}_t, & \text{if } (\mathbf{P}_f-\mathbf{g}_t)\cdot\mathbf{n}_t \geq 0 \\ \mathbf{g}_t, & \text{otherwise} \end{cases} \tag{4}$$

$$\mathbf{v}_i' \;=\; \arg\min_{\mathbf{x}}\Big(\|\mathbf{x}-\mathbf{v}_i\|^2 + \lambda \sum_{f\in\mathcal{N}(i)} \|\mathbf{n}_f - \mathbf{n}_f^{\text{orig}}\|^2\Big) \quad \text{s.t.} \quad (\mathbf{x}-\mathbf{P}_a^t)\cdot\mathbf{n}_t = 0 \;\; \forall t \ni i. \tag{5}$$

The final cage vertex $\mathbf{v}_i'$ is determined by the following constraint formula: First, for the point $\mathbf{x}$ that satisfies the conditions, a hard - constraint is applied through $(\mathbf{x}-\mathbf{P}_a^t)\cdot\mathbf{n}_t = 0$ to ensure that all new cage vertices are in the same plane as the anchor point $\mathbf{p}_a$, so as to guarantee that the cage can enclose the mesh model. Second, the Euclidean distance between the new vertex and the original vertex is calculated using $\|\mathbf{x}-\mathbf{v}_i\|^2$ to prevent the new vertex from moving too far and reduce the detail differences with the original model, where $\mathbf{v}_i$ represents the original vertex. Thirdly, $\lambda\sum_{f\in\mathcal{N}(i)}\|\mathbf{n}_f-\mathbf{n}_f^{\text{orig}}\|^2$ is used to calculate the weighted sum of the differences between the normal vectors of all faces adjacent to the original vertex $\mathbf{v}_i$ before and after the change. Our goal is to ensure that the normal vectors of the adjacent faces before and after the change are as consistent as possible with the original state to prevent local sharp creases or distortions and other problems. Here, λ is used to balance the influence of the above two items on the final result, $\mathcal{N}(i)$ represents all the adjacent faces of $\mathbf{v}_i$, $\mathbf{n}_f$ is the normal vector of face f after the change, and $\mathbf{n}_f^{\text{orig}}$ is the normal vector of face f before the change. After multiple iterations, the $\mathbf{x}$ that makes the value of this formula the smallest is finally selected as the final cage vertex $\mathbf{v}_i'$.

To prevent geometric distortion, outward displacement is constrained by a user-defined distance threshold Δd, ensuring the final cage tightly envelops the input mesh M while retaining deformation-ready structural properties. This equilibrium between containment and flexibility makes the cage suitable for downstream animation tasks.

4 RESULTS AND DISCUSSION

This method is implemented in Python and evaluated on a Windows 10 workstation with an Intel Xeon 6248 CPU (2.5GHz) and 64GB RAM. Prior to preprocessing, all input meshes are normalized by scaling into a unit bounding sphere. We employ Q-MAT [9] for medial axis computation and simplification, Tetgen [29] for tetrahedral mesh generation, CGAL [30] for convex hull calculations.

In Table 1 We evaluate the difference between the shapes converted from the skeletons and the ground truth shapes using the Chamfer distance (CD) and Hausdorff distance (HD), referred to as CD-Con and HD-Con, respectively. ground truth shapes are the joint node information generated and labeled using Q-mat. shapes converted from the skeletons are the union balls of the predicted skeletal spheres, inflated by their radii. The L1-medial skeleton [31] generates structured representations consisting solely of 1D curves, which can lead to significant errors when abstracting non-tubular shapes. The DPC method [32] is capable of producing both surface-like and curve-like skeletons; however, its representations consist of unstructured points with no topological constraints, resulting in inconsistencies in thin structures.Compare our method with these two methods.

Table 1: Comparison Results of L1, DPC, and Ours Methods Based on CD and HD Evaluation Metrics.

	CD-Con↓			HD-Con↓		
	L1	DPC	Ours	L1	DPC	Ours
animal	0.0476	0.0485	0.0483	0.3216	0.2466	0.2256
beast	0.1326	0.0759	0.0451	0.4820	0.2468	0.1678
chair	0.1041	0.0853	0.0424	0.3453	0.2584	0.1745
hand	0.1542	0.0712	0.0335	0.3956	0.1850	0.1382

4.1 Comparisons

The results are compared with two state-of-the-art cage generation methods: one fully automatic and one interactive. For the automatic approach, the methods by Guo and Zhang et al. [20] and Sara Casti et al. [24] are utilized; for the interactive approach, the technique by Le and Deng [22] is considered. To ensure a fair comparison, all methods are evaluated on identical meshes. The following evaluation metric is proposed to assess the quality of the generated cages:

To validate the effectiveness of this proposed evaluation metric, we generated three distinct cage configurations as demonstrated in Figure 8. The quantitative scores and associated parameters for these configurations are systematically presented in Table 2. A critical observation reveals that the configuration achieving the lowest evaluation score simultaneously exhibits optimal visual compliance with geometric cage requirements, particularly in edge alignment and symmetry.

- c: Volume of the generated cage
- v: Volume of the input mesh
- f: Number of faces in the cage

The ratio $\frac{c}{v}$ quantifies encapsulation tightness (lower values indicate tighter wrapping), and f measures cage simplicity (fewer faces preferred). Both evaluation metrics follow the lower-is-better principle.

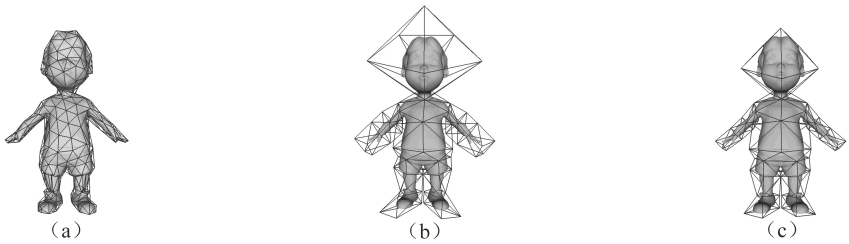

Figure 8: Among three generated instances for the same mesh, instance (a) achieves the smallest volume difference but requires the highest number of faces. Instances (b) and (c) share the same face count, yet instance (b) exhibits (a) larger volume difference compared to (c).

Table 2: Three boy parameter results

Mesh	v	c	c/v ↓	f ↓
a	0.2815	0.3333	1.6136	705
b	0.2815	0.8578	3.0464	499
c	0.2815	0.4544	1.1836	264

Figure 9 provides a comparative analysis between this method and the automated cage generation approach proposed in [20]. While the latter demonstrates competent enclosure of the character within relatively tight volumetric bounds, critical deficiencies emerge under closer inspection. Specifically, the vertex distribution in their generated cages exhibits a construct absence of semantic alignment with animation-driven requirements.

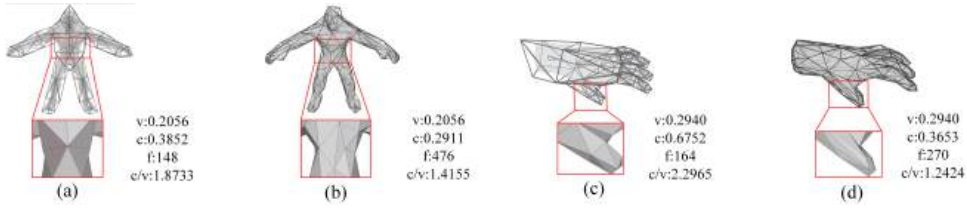

Figure 9: The left column illustrates this method's accurate processing of vertex semantics in critical anatomical regions (waist and finger joints), whereas the right column exposes the limitations of Method [20] in preserving semantic correspondence within these biomechanically significant areas.

To evaluate these discrepancies, Figures 9 and Figure 10 juxtapose the structural output of both methodologies. Key failure modes of the baseline method become apparent in Figures 9(b) and 9(d): (1) Inadequate surface segmentation at biomechanically critical regions (waist and finger joints), leading to non-physical bend deformation during articulation; (2) Overly dense triangulation (average the number of surfaces of this method is twice as many as ours.) that violates the simplicity principle essential for practical animation workflows. Although their approach achieves superior geometric tightness, this comes at the prohibitive cost of computational tractability and deformation controllability.

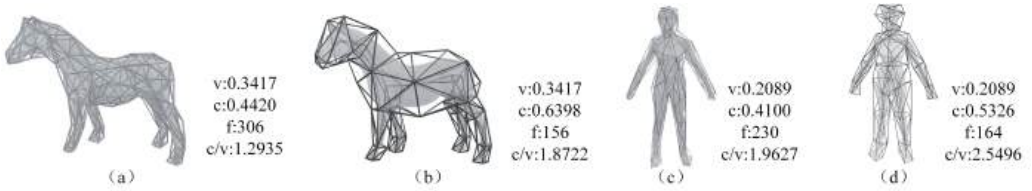

Figure 10: Visual comparison between the automatic method [20] (left), and this method (right). General-purpose cages are not suitable for animation, as they introduce unnecessary vertices, and do not align with the semantic features of the character.

Figure 11 demonstrates comparisons between this method and the interactive approaches proposed in [22] and [24]. Compared to [24], which requires direct user-defined mesh cutting to generate cages significantly increasing interaction complexity our method introduces symmetric cross-section operations for meshes with symmetry axes. This enhancement enables our cages to better align with the intrinsic animation semantics of symmetrical meshes. Specifically, users need only select nodes on the simplified MAT skeleton to trigger automatic cutting plane generation.

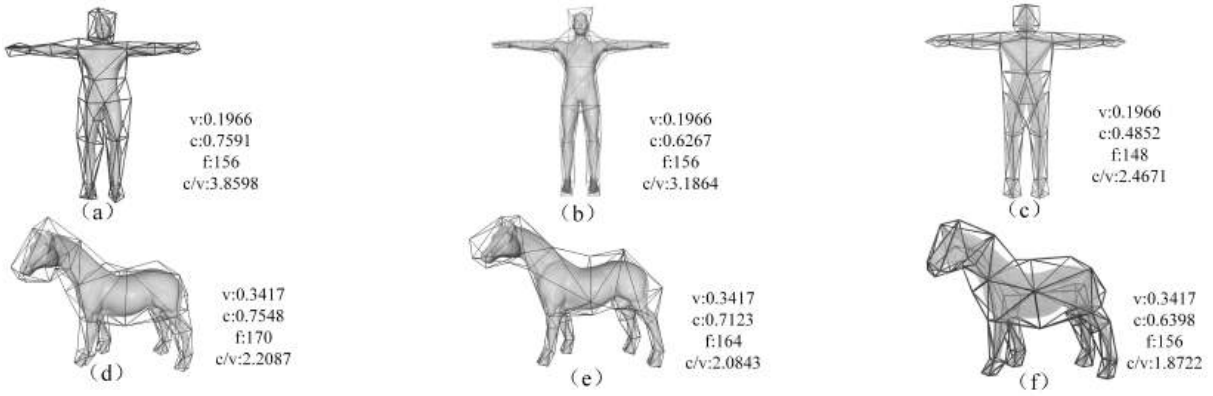

Figure 11: Comparison between cages obtained with: interactive cutting planes [22] (left column); skeleton based cage generation guided by harmonic fields [24](middle column); the proposed method (right column). ManTpose and Horse datasets.

In contrast to [24] that requires both input meshes and skeletal structures, the proposed method operates solely on the input geometry, substantially reducing preprocessing requirements. Furthermore, this method exhibits superior compatibility with non-humanoid and non-animal meshes lacking explicit skeletal structures, such as the airplane and chair shown in Figure 12. These results validate our framework's ability to generate high-quality cages for geometrically diverse meshes without skeletal priors. To ensure fairness, we have added all quantitative results for the beast and hand model in Figure 13 of experimental results has been added including the results of ours, [20], [22], and [24].

A gallery of cages generated by this method is shown in Figure 14. Notably, this approach effectively handles geometrically complex objects. Furthermore, it successfully generates high-quality cages even for non-humanoid meshes, demonstrating robust adaptability across diverse topological structures.

Table 3 summarises the tightness-to-simplicity trade-off achieved by our method on the full benchmark set.Across all categories the average c/v ratio is 2.21, demonstrating consistently tight wrapping, while the

Table 3: Size of meshes and timing for the whole pipeline: $\mathbf{T}_{\mathrm{pp}}$ is the predicted time of the joint node; $\mathbf{T}_{\mathrm{cg}}$ is the time of cross-sectional generated by the joint node and fit them into quadrilaterals; $\mathbf{T}_{\mathrm{rg}}$ is the rough cage generation time; $\mathbf{T}_{\mathrm{ce}}$ is the time required for cage expansion; $\mathbf{T}_{\mathrm{all}}$ is the sum of the time in the previous columns rounded to integer. All time are in seconds.

Mesh	v	c	c/v ↓	f ↓	$\mathbf{T}_{\mathrm{pp}}$	$\mathbf{T}_{\mathrm{cg}}$	$\mathbf{T}_{\mathrm{rg}}$	$\mathbf{T}_{\mathrm{ce}}$	$\mathbf{T}_{\mathrm{all}}$
animal	0.1931	0.4236	2.1927	205	0.86	2.67	0.53	613	617
beast	0.2056	0.3852	1.8733	148	0.83	1.12	0.47	525	527
chair	0.1476	0.2955	2.0017	80	0.53	0.54	0.31	314	315
hand	0.2940	0.6752	2.2965	164	0.65	1.42	0.43	529	531
bug	0.2807	0.8895	3.1685	268	1.12	2.36	0.64	691	695
man	0.2089	0.5326	2.5496	164	0.84	1.12	0.46	576	579
antcat	0.1879	0.3480	1.8518	252	0.92	2.14	0.68	642	646
Tposeman	0.1966	0.6267	3.1864	156	0.84	1.03	0.45	542	544
plane	0.0669	0.1190	1.7773	148	0.64	1.79	0.44	183	186
octupus	0.0805	0.1326	1.6471	428	1.85	4.57	1.05	956	964
vase	0.2040	0.4341	2.1276	114	0.59	1.36	0.37	362	365
horse	0.3417	0.6398	1.8722	156	0.59	1.78	0.42	574	578

face count f remains below 430 in every case. Our method is faster in predicting joint nodes and eliminates the need for cumbersome manual interaction. Currently, most of the computational time is spent on the expansion step.

Table 4: Comparison of cage result parameters.

Mesh	v	c	c/v ↓	f ↓	Mesh	v	c	c/v ↓	f ↓
beast(ours)	0.2056	0.3852	1.8733	148	hand(ours)	0.2940	0.6752	2.2965	164
beast[22]	0.2056	0.3976	1.9338	163	hand[22]	0.2940	0.6945	2.3622	161
beast[20]	0.2056	0.2911	1.4155	476	hand[20]	0.2940	0.3653	1.2424	270
beast[24]	0.2056	0.4015	1.9528	171	hand[24]	0.2940	0.6859	2.3329	173
Tposeman(ours)	0.1966	0.4852	2.4671	148	horse(ours)	0.3417	0.6398	1.8722	156
Tposeman[22]	0.1966	0.6267	3.1864	156	horse[22]	0.3417	0.7548	2.2087	170
Tposeman[20]	0.1966	0.3631	1.8468	295	horse[20]	0.3417	0.4420	1.2935	306
Tposeman[24]	0.1966	0.7591	3.8598	156	horse[24]	0.3417	0.7123	2.0843	164
woman(ours)	0.2089	0.5326	2.5496	164	woman[22]	0.2089	0.5448	2.6079	152
woman[20]	0.2089	0.4100	1.9627	230	woman[24]	0.2089	0.5147	2.4638	162

A detailed pairwise evaluation is given in table 4. For identical input meshes our cages exhibit comparable or lower c/v values while using fewer faces. Only a few examples were exception where [20] obtains a slightly tighter cage at the cost of more triangles.These numbers confirm that the MAT-driven pipeline moves the Pareto frontier towards higher simplicity without sacrificing enclosure quality, directly addressing criteria (a) and (c) formulated in Section 1. The average c/v ratio is 2.21, which is lower than 2.46 in [22] and 2.54 in [24]. Although slightly higher than 1.55 in [20], our average f is 156, which is only about half of 296 in [20], a decrease of 47%.

4.2 Limitation

This method inflates the coarse cage by extruding its triangular faces along the displacement vectors toward precomputed anchor points. This constrained displacement is While effective for most cases, extreme geometric configurations (e.g., high-curvature regions) may induce local self-intersections due to non-uniform anchor distributions.

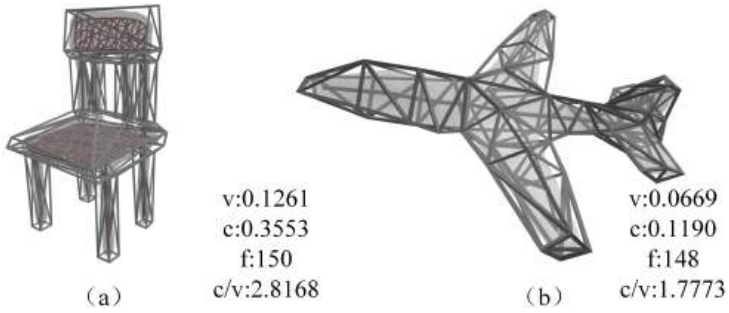

Figure 12: The cage of explicit skeletal structures.left is chair, right is plane.

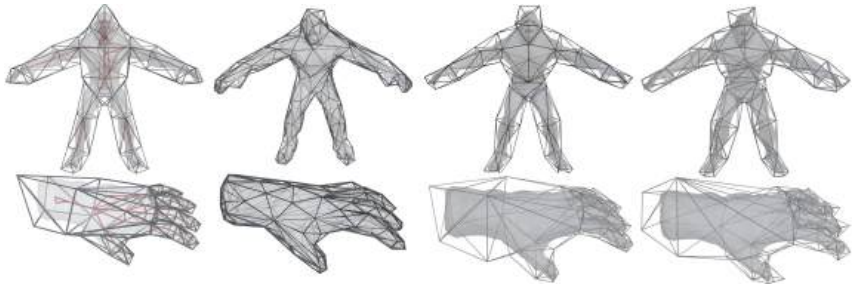

Figure 13: Comparison between cages obtained with: our method; [20]; [24]; [22]. Beast and Hand result.

5 CONCLUSION AND FUTURE WORK

We propose a novel cage generation framework guided by the MAT of input meshes. Unlike existing approaches that require explicit skeletal priors, this method derives structural guidance through MAT simplification, significantly reducing user interaction complexity while [20] producing high-quality cages. Although numerous mature cage generation methods exist in 3D modeling, this future work will focus on extending these principles to 2D image-based cage generation, enabling shape manipulation through single-view inputs while preserving topological coherence.

REFERENCES

[1] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011.

[2] Daniel Ströter, Jean-Marc Thiery, Kai Hormann, Jiong Chen, Qingjun Chang, Sebastian Besler, Johannes Sebastian Mueller-Roemer, Tamy Boubekeur, André Stork, and Dieter W Fellner. A survey on cage-based deformation of 3d models. In *Computer Graphics Forum*, volume 43, page e15060. Wiley Online Library, 2024.

[3] Andreas Scalas, Yuanju Zhu, Franca Giannini, Ruding Lou, Katia Lupinetti, Marina Monti, Michela Mortara, and Michela Spagnuolo. A first step towards cage-based deformation in virtual reality. In *Smart Tools and Applications in computer Graphics-Eurographics Italian Chapter Conference*, pages 119–130. Eurographics, 2020.

[4] Yu-Jie Yuan, Yu-Kun Lai, Tong Wu, Lin Gao, and Ligang Liu. A revisit of shape editing techniques: from the geometric to the neural viewpoint. corr (2021), 2021.

[5] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, et al. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12478–12487, 2021.

[6] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics*, 41(4):1–18, 2022.

[7] Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. Surface and edge detection for primitive fitting of point clouds. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–10, 2023.

[8] Joachim Giesen, Balint Miklos, Mark Pauly, and Camille Wormser. The scale axis transform. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 106–115, 2009.

[9] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics*, 35(1):1–16, 2015.

[10] Andreas Vasilakis and Ioannis Fudos. Skeleton-based rigid skinning for character animation. *GRAPP*, 9:302–308, 2009.

[11] Fabrizio Corda, Jean-Marc Thiery, Marco Livesu, Enrico Puppo, Tamy Boubekeur, and Riccardo Scateni. Real-time deformation with coupled cages and skeletons. In *Computer Graphics Forum*, volume 39, pages 19–32. Wiley Online Library, 2020.

[12] Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics*, 34(4):1–11, 2015.

[13] Hugues Hoppe. Progressive meshes. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 111–120. 2023.

[14] Chen Shen, James F O'Brien, and Jonathan R Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004 Papers*, pages 896–904. 2004.

[15] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Spatial deformation transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–74, 2009.

[16] Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. Automatic cage building with quadric error metrics. *Journal of Computer Science and Technology*, 26(3):538–547, 2011.

[17] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. Nested cages. *ACM Transactions on Graphics*, 34(6):1–14, 2015.

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

[19] Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. In *ACM SIGGRAPH 2009 papers*, pages 1–9. 2009.

[20] Jia-Peng Guo, Wen-Xiang Zhang, Chunyang Ye, and Xiao-Ming Fu. Robust coarse cage construction with small approximation errors. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

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

[22] Binh Huy Le and Zhigang Deng. Interactive cage generation for mesh deformation. In *Proceedings of the 21st ACM SIGGRAPH symposium on interactive 3D graphics and games*, pages 1–9, 2017.

[23] Stéphane Calderon and Tamy Boubekeur. Bounding proxies for shape approximation. *ACM Transactions on Graphics*, 36(4):57–1, 2017.

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

[25] Cheng Lin, Changjian Li, Yuan Liu, Nenglun Chen, Yi-King Choi, and Wenping Wang. Point2skeleton: Learning skeletal representations from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4277–4286, 2021.

[26] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition*, pages 4867–4876, 2020.

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[28] Daniele Panozzo, Yaron Lipman, Enrico Puppo, and Denis Zorin. Fields on symmetric surfaces. *ACM Transactions on Graphics*, 31(4):1–12, 2012.

[29] Si HTetGen. a delaunay-based quality tetrahedral mesh generatoracm. *Trans Math Softw2015412Art*, 11:363318083.

[30] Efi Fogel and Monique Teillaud. The computational geometry algorithms library cgal. *ACM Communications in Computer Algebra*, 49(1):10–12, 2015.

[31] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Transactions on Graphics*, 32(4):65–1, 2013.

[32] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 international conference on 3D vision (3DV)*, pages 298–307. IEEE, 2019.

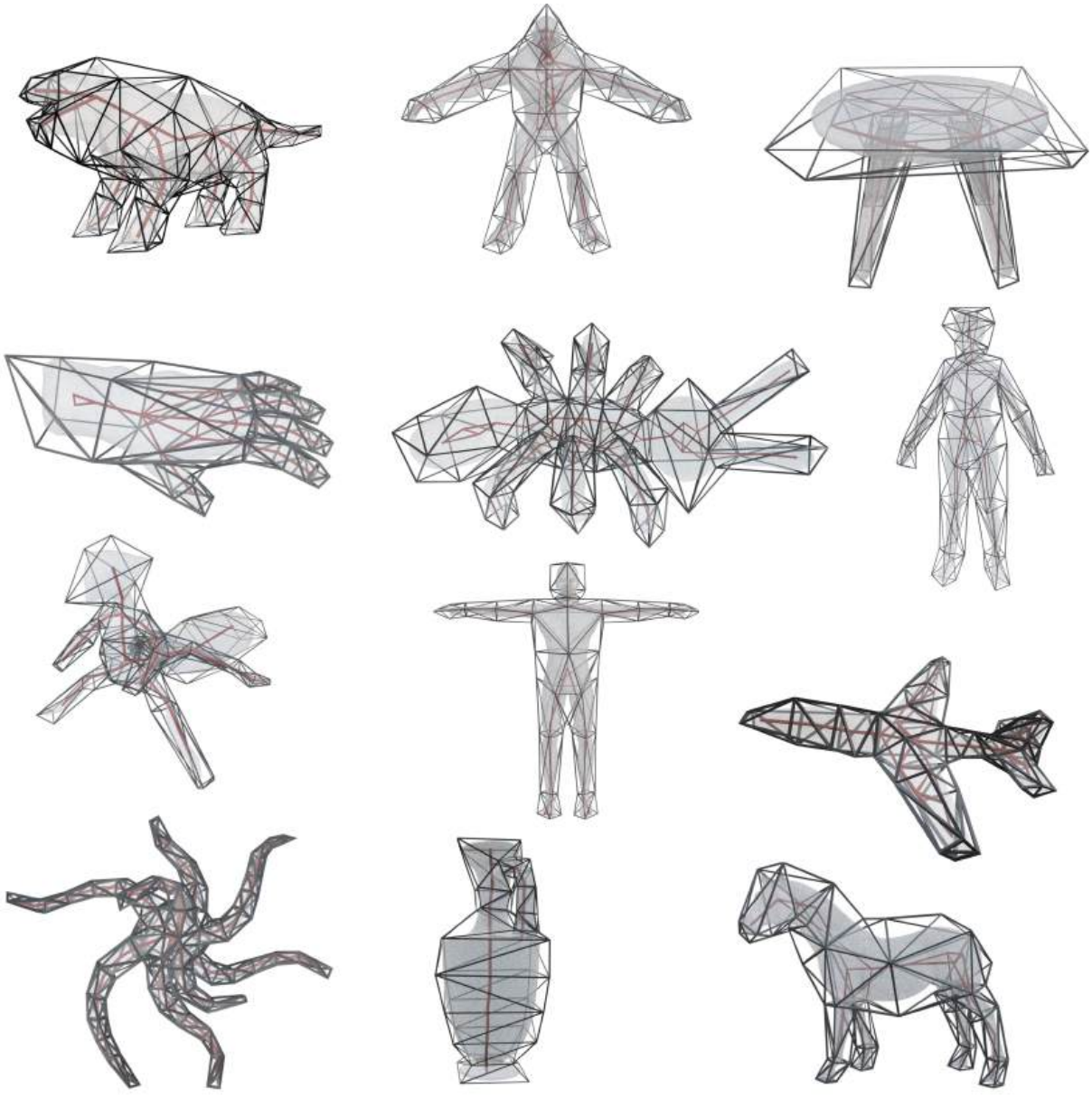

Figure 14: The collection of cages for meshes with this method.