

Medial Skeletal Diagram: A Generalized Medial Axis Approach for 3D Shape Representation

MINGHAO GUO*, BOHAN WANG*, and WOJCIECH MATUSIK, CSAIL, MIT, USA

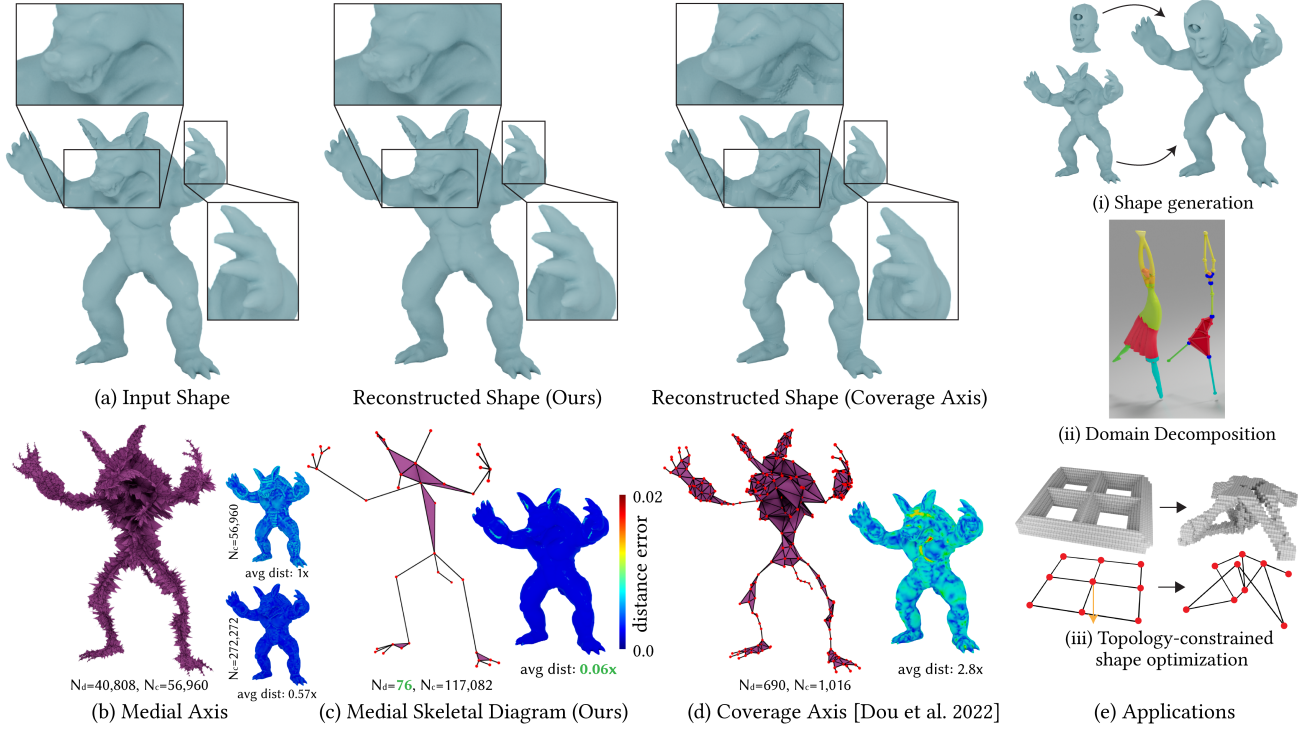


Fig. 1. We introduce the Medial Skeletal Diagram – a representation with a sparse skeleton that generalizes the medial axis. Our representation reduces the number of discrete elements (N_d) required compared to both standard and simplified medial axes ((b) and (d)), using only 76 elements compared to their 40,808 and 690, respectively. This reduction is achieved while delivering the highest reconstruction accuracy (c). The sparsity of discrete elements and the completeness of our method facilitate a broad range of applications, such as shape generation, mesh decomposition, and shape optimization (e). In terms of continuous parameters (N_c), our method uses 117,082 scalars, whereas the medial axis and coverage axis use 56,960 and 1,016 scalars. Experimentally, even a 4.8 $\times$ increase in the number of scalars for the medial axis (2.3 $\times$ our method) – undesirably resulting in an approximate proportional increase in the number of discrete elements to 201,323 – still yields an average reconstruction error that is 9 $\times$ higher than that of our method.

We propose the Medial Skeletal Diagram, a novel skeletal representation that tackles the prevailing issues around skeleton sparsity and reconstruction accuracy in existing skeletal representations. Our approach augments the continuous elements in the medial axis representation to effectively shift the complexity away from the discrete elements. To that end, we introduce

*Both authors contributed equally to this research.

Authors' Contact Information: Minghao Guo, guomh2014@gmail.com; Bohan Wang, wangbh11@gmail.com; Wojciech Matusik, wojciech@csail.mit.edu, CSAIL, MIT, Cambridge, MA, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2024 Copyright held by the owner/author(s).

ACM 1557-7368/2024/12-ART217

<https://doi.org/10.1145/3687964>

generalized enveloping primitives, an enhancement over the standard primitives in the medial axis, which ensure efficient coverage of intricate local features of the input shape and substantially reduce the number of discrete elements required. Moreover, we present a computational framework for constructing a medial skeletal diagram from an arbitrary closed manifold mesh. Our optimization pipeline ensures that the resulting medial skeletal diagram comprehensively covers the input shape with the fewest primitives. Additionally, each optimized primitive undergoes a post-refinement process to guarantee an accurate match with the source mesh in both geometry and tessellation. We validate our approach on a comprehensive benchmark of 100 shapes, demonstrating the sparsity of the discrete elements and superior reconstruction accuracy across a variety of cases. Finally, we exemplify the versatility of our representation in downstream applications such as shape generation, mesh decomposition, shape optimization, mesh alignment, mesh compression, and user-interactive design.

CCS Concepts: • Computing methodologies → Shape analysis.

Additional Key Words and Phrases: geometry representation, medial axis transform, skeleton, shape analysis

ACM Reference Format:

Minghao Guo, Bohan Wang, and Wojciech Matusik. 2024. Medial Skeletal Diagram: A Generalized Medial Axis Approach for 3D Shape Representation. *ACM Trans. Graph.* 43, 6, Article 217 (December 2024), 23 pages. <https://doi.org/10.1145/3687964>

1 Introduction

Geometric shape representations are fundamental to diverse areas of computer graphics and geometric modeling. Among them, the shape skeleton stands out as a powerful and intuitive tool for understanding and manipulating shapes, with a broad range of applications including 3D reconstruction [Tang et al. 2019], shape segmentation [Lin et al. 2020], character animation [Baran and Popović 2007], and shape correspondence [Wong et al. 2012]. As a prominent example among skeletal representations, the *medial axis* [Blum 1967] is ubiquitous in shape approximation, simplification, and analysis tasks. Its approximative counterpart, *medial axis transform* (MAT), comprises a *discrete* simplicial complex (termed *medial mesh*) and a *continuous* radius function defined on each vertex of the complex. Consequently, MAT enjoys many advantages including capture of components and protrusions, and preservation of homotopy [Lieu-tier 2003].

Nonetheless, like most existing skeletal representations, the medial axis representation encounters an inevitable trade-off between *sparsity* in the skeletal structure and *completeness* in the reconstruction. The skeletal structure should ideally be as simple and sparse as possible to allow for easy interpretation and modification. However, it is equally important to retain sufficient complexity to faithfully reconstruct all geometric details and features. Balancing such conflicting requirements poses a significant challenge. The excessive complexity hinders the application of medial axis to shape generation and interactive shape editing, which require a sparser skeleton but a complete representation. Numerous research efforts have been dedicated to simplifying the discrete part of the skeletal representation to enhance its manageability and visual cleanliness [Dou et al. 2022; Li et al. 2015; Lin et al. 2021; Tagliasacchi et al. 2016]. However, despite these advancements, the simplified skeleton is still bounded by excessively many elements with compromised reconstruction accuracy, resulting in degraded quality of the skeleton and a loss of geometric detail.

To conquer these limitations, we present a novel skeletal representation that significantly reduces the number of discrete elements while preserving the completeness characteristic of the medial axis. Our method stems from the insight that the complexity of the medial axis skeleton can be decimated by an augmented representation of the *continuous* elements. Our representation shifts the burden from the number of discrete elements to the number of continuous parameters, achieving a significant reduction in discrete elements at the cost of increasing the number of continuous parameters. Discrete elements are often challenging to optimize due to their combinatorial nature, which results in severe computational bottlenecks. In contrast, continuous elements can be formulated and solved efficiently using off-the-shelf optimization frameworks.

Our versatile skeletal representation, named the *medial skeletal diagram* (MSD), extends the medial axis using *generalized enveloping primitives*. Compared with the uniform, linearly interpolated primitives used in the medial axis, our generalized enveloping primitives are *non-uniform* and *nonlinearly interpolated*. For example, the medial axis assumes consistent and uniform radii, whereas in our medial skeletal diagram, each medial sphere has varying radii along different directions. Such generalized primitives can effectively cover complex local regions, which in turn reduces the number of primitives required to represent a given shape. As a result, the medial skeletal diagram can represent arbitrary closed manifold surfaces, with the raw medial axis being a special case.

To construct a medial skeletal diagram for an arbitrary input, we start by building a forward procedure. This procedure takes a subset of points from the raw medial axis as its input and produces an MSD as its output. Utilizing *restricted Voronoi diagrams* (RVD) and *restricted Delaunay triangulation* (RDT), we determine the discrete skeletal structure based on the input set of points, which effectively “remesh” the medial axis. Following this, generalized enveloping primitives are generated at each element of the discrete skeletal structure to align with the input mesh. With this established procedure, transforming a subset of medial axis points into a medial skeletal diagram, we then introduce an optimization pipeline. This pipeline is designed to identify the optimal subset of points, ensuring that the resulting medial skeletal diagram covers the input shape with the fewest necessary primitives. In the final stage, the optimized primitives undergo a refinement phase, aimed at achieving precise alignment with the target mesh in both its geometric form and tessellation.

The contributions of this paper are summarized as follows:

- We propose a novel 3D shape representation (in Sec. 4) that extends the medial axis approach using generalized enveloping primitives, enabling both sparse skeletal structures and accurate representation for arbitrary closed manifold surfaces.
- We introduce a computational pipeline to construct the medial skeletal diagram from any given shape (in Sec. 5). Our pipeline finds the best MSD which maximizes the coverage of the input shape with a minimal number of primitives. A subsequent refinement process precisely aligns optimized primitives with the target mesh, both in terms of geometry and tessellation.
- We validate the medial skeletal diagram on a benchmark of 100 shapes to demonstrate the sparsity of the skeleton and the completeness of the representation (in Sec. 6). Furthermore, we showcase the versatility and advantages of our representation in the context of shape optimization, shape generation, mesh decomposition, mesh alignment, mesh compression, and user-interactive design.

2 Related Work

Medial axis transformation. MAT represents a 3D shape using thin-centered structures and is able to jointly describe shape topology and geometry. The rich history of MAT starts from early algorithms that employ seam tracing techniques to calculate the exact MAT for polyhedra [Culver et al. 2004; Milenkovic 1993; Sherbrooke

et al. 1996]. These exact algorithms primarily concentrate on simple shapes with up to 20 faces. A significant amount of subsequent research has shifted towards approximated MAT, which is more suitable for practical applications [Amenta et al. 2001b; Chazal and Lieutier 2005; Dey and Zhao 2004; Miklos et al. 2010; Pizer et al. 2003; Saha et al. 2016; Sobiecki et al. 2014]. For a more comprehensive discussion, we refer the reader to survey papers by Pizer et al. [2003] and Tagliasacchi et al. [2016].

Curve skeletonization. One type of shape representation relevant to our work is curve skeletons, which have been widely studied in rigging and interactive shape modeling [Borosán et al. 2012]. Traditional approaches compute curve skeletons following hand-crafted rules that encode geometric information. For instance, Ma et al. [2003] extract skeletons using radial basis functions. Sharf et al. [2007] apply deformable model evolution to capture the volumetric shape of an object for curve skeleton approximation. Au et al. [2008] compute the curve skeleton via mesh extraction. Livesu et al. [2012] use visual hulls to reconstruct curve skeletons. Mean curvature skeletonization [Tagliasacchi et al. 2012] creates curve skeletons by deploying mean curvature flow on the shape. Reeb graphs [Tierny et al. 2008] are also considered curve skeletons, which are constructed by tracing the isocontours of a given height function and contracting connected components into a single point. However, they are often suboptimal as skeleton junctions can be very close to the surface. Recently, Bærentzen and Rotenberg [2021] and Bærentzen et al. [2023] treat 3D shapes as spatially embedded graphs and compute curve skeletons using local operators. In shape modeling, Bærentzen et al. [2014] use polar-annular mesh to obtain a co-representation of both the mesh and the curve skeleton. Pandey et al. [2022] obtain a curve skeleton from a sequence of face-loop modeling operations on a quad mesh. Although curve skeletons are well-suited for representing organic, articulated shapes, they often suffer from capturing large thin flat components, which are prevalent in CAD models. Our representation incorporates triangles and effectively captures large, flat regions with fewer elements.

Medial axis Simplification. The medial axis representation is complete but sensitive to boundary noise. This has led to a large body of research on its simplification while preserving significant and stable parts. Angle-based filtering methods [Amenta et al. 2001a; Dey and Zhao 2002; Foskey et al. 2003; Sud et al. 2005] consider the angle in MAT during simplification but struggle to maintain the original topology. λ -medial axis methods [Chaussard et al. 2011; Chazal and Lieutier 2005] use the radius of the closest medial sphere as a pruning criterion but have limitations in feature preservation at different scales [Attali et al. 2009]. The Scale Axis Transform (SAT) [Miklos et al. 2010] effectively prunes spikes by identifying unstable medial axis points. However, SAT has high computational costs and risks topology disruption by introducing new topological structures at large scales. Delta Medial Axis [Marie et al. 2016] and Bending Potential Ratio pruning [Shen et al. 2011] primarily focus on MAT simplification but cannot precisely reconstruct the original shape. Q-MAT [Li et al. 2015] collapses edges of MAT based solely on local information, which may lead to a suboptimal spatial distribution of medial vertices. Voxel Cores [Yan et al. 2018] approximate the medial axis of any smooth shape while guaranteeing correct

topology but require fine voxel resolutions and high computational cost to achieve high geometric accuracy. Erosion Thickness [Yan et al. 2016] computes the simplified medial axis by introducing a burning process over the medial axis but is unable to reconstruct the original shape exactly. Coverage Axis [Dou et al. 2022] aims to encompass all mesh surface points while minimizing the number of internal medial spheres. MATFP [Wang et al. 2022a] computes MAT using a restricted power diagram while ensuring the preservation of both external mesh surface features and internal medial axis features. Both Coverage Axis and MATFP serve as competitive baselines for comparison in our experiments. Another line of research on point cloud data skeletonization includes methods like l_1 -medial skeleton [Huang et al. 2013], LSMAT [Rebain et al. 2019], and meso-skeleton based approaches [Tagliasacchi et al. 2012; Wu et al. 2015]. However, these methods often lack topological constraints or exhibit large errors. Recently, deep learning has been used for skeleton construction including P2MAT-NET [Yang et al. 2020], Deep medial fields [Rebain et al. 2021], and Point2Skeleton [Lin et al. 2021]. They do not guarantee accurate geometric features and suffer from generalization issues due to training data dependency. Compared to these medial axis simplification methods, our approach outputs a sparser skeleton and employs a global optimization process to obtain the skeleton as opposed to a suboptimal heuristic search. These improvements allow our approach to efficiently generate higher-quality skeletons for various applications.

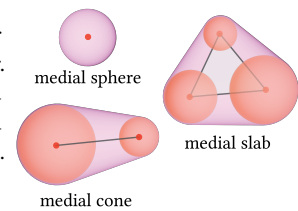
Shape Decomposition. The generalized enveloping primitives in our method share similarities with several variants proposed in the field of shape decomposition [Simari and Singh 2005; Wei et al. 2022; Zhou et al. 2015]. Examples of such methods include the use of generalized cylinders [Zhou et al. 2015] and ellipsoidal surface regions [Simari and Singh 2005], both involving the use of predefined primitives. Our method has a distinct primary objective from shape decomposition as we aim to extract a sparse and meaningful skeleton. By contrast, most shape decomposition methods do not yield a skeleton directly based on the decomposed domains. Although some of these methods, including generalized cylinder decomposition [Zhou et al. 2015], can produce a curve skeleton, they share the typical limitations of other curve skeleton techniques and do not attain the level of sparsity that our approach offers.

3 Preliminaries

3.1 Medial Axis and Medial Mesh

Consider a closed, oriented shape $\mathcal{S}$ in $\mathbb{R}^3$. The medial axis $\mathcal{M}$ is defined as the set of centers of maximal spheres inscribed within $\mathcal{S}$. Each sphere is tangent to at least two points on the boundary of $\mathcal{S}$ and does not contain any other boundary points in its interior. The MAT comprises both $\mathcal{M}$ and the radius function associated with each sphere center [Li et al. 2015].

Typically, MAT is approximated using a 2D simplicial complex known as the *medial mesh* [Li et al. 2015; Sun et al. 2015]. The medial mesh is a triangle mesh that



includes three types of basic elements: *medial vertex*, denoted by v ; *medial edge*, a line segment connecting two medial vertices, denoted by e ; and *medial face*, the convex combination of three medial vertices, defined by f . Each of these three elements corresponds to an *enveloping primitive* in $\mathbb{R}^3$: 1) *medial sphere* which is a uniform sphere centered at the corresponding medial vertex; 2) *medial cone*, which is the linear interpolation of two spheres connected by a medial edge; and 3) *medial slab*, the convex hull of three spheres connected by a medial triangle face. See the inset figure for an illustration.

3.2 Restricted Voronoi Diagram and Delaunay Triangulation

Given a finite set of points $V = \{v_k\} \subset \mathbb{R}^n$, the *Voronoi cell* of a point v_k comprises all points in $\mathbb{R}^n$ whose distance to v_k is no greater than their distance to any other point in V . It can be defined as

$$\text{Vor}(v_k, V) = \{x \in \mathbb{R}^n, \|v_k - x\| \leq \|v_l - x\|, \forall l \neq k\}.$$

The *Voronoi diagram* is the collection of all Voronoi cells. The *Delaunay triangulation* is the complex with a dual structure to the Voronoi diagram. Each d -dimensional Voronoi element is dual to an $(n-d)$ -dimensional Delaunay element, which is defined as the convex hull of points in V whose Voronoi cells have the considered Voronoi element on their boundary. For instance, when $n = 3$, a Delaunay vertex corresponds to a Voronoi volumetric cell, a Delaunay edge corresponds to a Voronoi surface, and vice versa.

The Voronoi diagram restricted within a given domain $X \subseteq \mathbb{R}^n$ is called *restricted Voronoi diagram* (RVD). RVD is defined as the collection of *restricted Voronoi cells* (RVC), each of which is the intersection between the Voronoi cell and the domain:

$$\begin{aligned} \text{RVC}(v_k, V) &= \text{Vor}(v_k, V) \cap X \\ &= \{x \in X, \|v_k - x\| \leq \|v_l - x\|, \forall l \neq k\}, \quad (1) \\ \text{RVD}(V, X) &= \{\text{RVC}(v, V), \forall v \in V, \text{Vor}(v, V) \neq \emptyset\}. \end{aligned}$$

Likewise, the *restricted Delaunay triangulation* (RVT) is the dual of the restricted Voronoi diagram, with the constraint that all its elements lie within the domain X .

4 Medial Skeletal Diagram Representation

In this section, we present how our proposed medial skeletal diagram represents 3D shapes (closed, oriented 2D manifolds). We leverage the medial axis to capture the topological and geometric information of 3D shapes but greatly improve the sparsity of the skeleton while maintaining its accuracy. Our fundamental insight is that while the convex nature of medial enveloping primitives offers advantages such as simplicity and rapid proximity query, it restricts the ability to accurately represent intricate local regions, especially those with high-curvature sharp features. This issue is depicted in Fig. 2, where it is clear that a substantial number of medial spheres are needed to encompass all vertices in sharp local regions, resulting in a complex

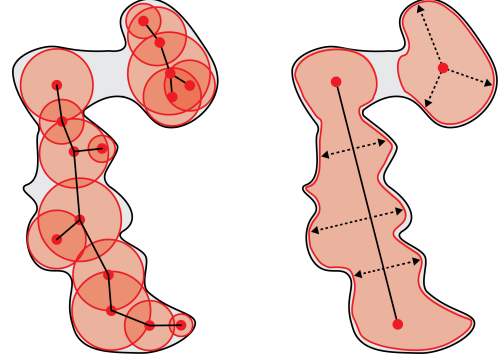


Fig. 2. **Generalized enveloping primitives in 2D.** An illustration of employing our generalized enveloping primitives (right) for representing a 2D region. Compared to the standard medial primitives (left), our representation exhibits enhanced the sparsity of the skeleton when dealing with intricate and high-curvature geometries.

and unwieldy medial axis. Furthermore, a majority of these spheres intersect with only a few vertices of the target mesh, often merely covering two vertices, which leads to considerable inefficiency. This problem is exacerbated by the fact that the medial axis becomes overly complicated and noisy even for shapes with simple topologies. An intuitive solution to these issues is to increase the complexity of the primitives associated with each skeleton element by making them more general, rather than restricting them to simple spheres. However, using arbitrarily general primitives can be problematic, as it may lead to a poorly defined skeleton. For example, as shown in Fig. 3, by simply dividing the original shape into generic subregions, there is a risk that the skeleton could potentially lie outside of the input shape. Therefore, a careful design of such general primitives is needed. To this end, we introduce *generalized enveloping primitives* for medial spheres, cones, and slabs. These new primitives have been crafted to cover complex local regions effectively with fewer units, thus facilitating the creation of a simpler skeleton.

4.1 Generalized Enveloping Primitives

We begin by outlining the formulation of generalized enveloping primitives. The basic idea is to replace the uniform, linearly interpolated primitives in the original medial axis with *non-uniform, nonlinearly interpolated* ones.

The generalized enveloping primitive is defined for all three types of medial mesh elements: v , e , and f . For convenience, we define $P := \text{cvx}(\{v_k\}_{k=1}^K) \in \{v, e, f\}$, where $\text{cvx}(\cdot)$ denotes the convex hull, $K \in \{1, 2, 3\}$ corresponds to the three types of medial mesh elements, respectively. We leverage the concept of ϵ -neighborhood [Lee 2012; Munkres 2018] to define our generalized enveloping primitive. The ϵ -neighborhood of P , denoted by P^ϵ , is defined as $P^\epsilon := \{y \in \mathbb{R}^3 \mid \|y - x\|_2 < \epsilon(x) \text{ for some } x \in P\}$. We consider the unit normal vector for each point y on the boundary ∂P^ϵ of P^ϵ . We associate a set of directional vectors for each point $p \in P$, denoted as

$$d_p(P) := \{\|n\|_2 = 1 \mid g(y, n) = p, \forall (y, n) \in N\partial P^\epsilon\}, \quad (2)$$

where $N\partial P^\epsilon = \{(y, n) \mid y \in \partial P^\epsilon, n \in N_p\partial P^\epsilon\}$ denotes the unit *normal bundle* of ∂P^ϵ and $g : N\partial P^\epsilon \mapsto P$ is a surjective mapping between the

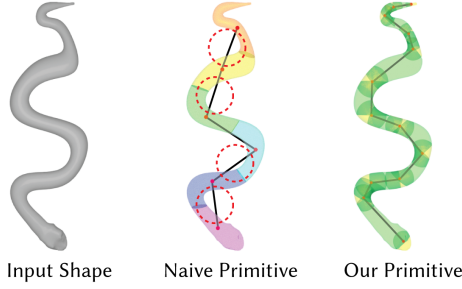


Fig. 3. **Naive primitives – created by simply dividing the original shape into generic subregions – can result in an invalid skeleton.** For illustration purpose, we selected a sparse set of points for naive primitives using the Coverage Axis [Dou et al. 2022]. The corresponding skeletons are not always contained within the primitive, and thus may not reside within the shape, as indicated by the red circles. While increasing the number of sampled points could create more subregions and enhance the likelihood of the skeleton being inside the mesh, these naive primitives do not guarantee that the skeleton is within the mesh. In contrast, our method ensures this containment.

normal bundle and the points in P . Appendix B provides a detailed analysis of the well-definedness of this surjection. We denote the set of all directional vectors of P as

$$d(P) = \{d_p(P) | \forall p \in P\}.$$

Fig. 4 shows ∂P^ϵ and $d(P)$ for all three types of medial mesh elements. In the case of a medial vertex, all the vectors in the set $d_p(P)$ belong to 2-sphere S^2 . For a medial edge, the set of associated vectors $d_p(P)$ for any interior point p forms a 1-sphere S^1 . Meanwhile, at either endpoint of the edge, $d_p(P)$ forms a 2-hemisphere. For a medial triangle, the set $d_p(P)$ exhibits different characteristics for interior and boundary points. For each point located in the interior of the medial triangle, $d_p(P)$ consists of the normal vector of the triangle plane and its negative. For points in the interior of the three edges of the medial triangle, $d_p(P)$ comprises 1-hemisphere. For each of the three vertices at the corners of the medial triangle, $d_p(P)$ consists of a sphere wedge, whose angle is equal to the external angle of the triangle at the respective corner.

By leveraging the directional vectors $d_p(P)$, we have definition of a *generalized enveloping primitive* as follows:

Definition 4.1 (Generalized enveloping primitive). Given a medial mesh element $P = \text{cvx}(\{v_k\}_{k=1}^K)$ and a smooth radius function $r(\cdot) : d(P) \mapsto \mathbb{R}_+$, the *generalized enveloping primitive* is defined by an implicit function as

$$\mathcal{E}_P(x) = \|x - p_x\|_2 - r(d_{p_x}), d_{p_x} = (x - p_x) / \|x - p_x\|_2, \quad (3)$$

where $p_x = \text{cp}(x; P)$ is the point in P which is closest to $x \in \mathbb{R}^3$.

One can observe that if $r(\cdot) \equiv \epsilon$, then the zero set of $\mathcal{E}_P(x)$ is ∂P^ϵ . Fig. 2 and 5 illustrate the generalized enveloping primitives representing local regions of a 2D and a 3D shape, respectively. Compared to the ones used in the standard medial mesh, our generalized enveloping primitives can cover a diverse range of local regions using fewer primitives.

It is crucial to highlight that our generalized enveloping primitive formulation encompasses the standard enveloping primitives used

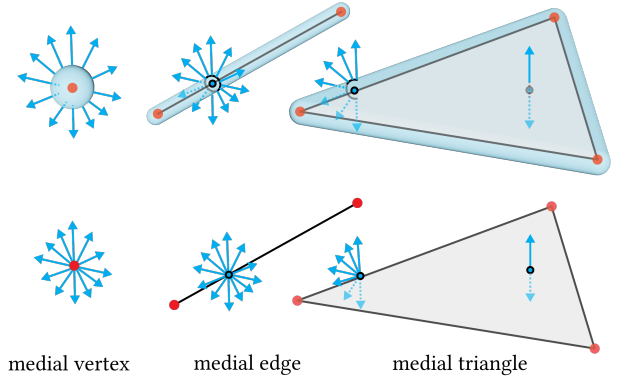


Fig. 4. **ϵ -neighborhoods and corresponding normal bundles.** Visualization of the boundary of P^ϵ for three medial mesh element types and the directional vectors $d(P)$ (only at selected points for clarity).

in the medial axis as a special case, where the radius function $r(\cdot)$ is uniform and linearly interpolated (Fig. 6). More specifically, for a medial sphere,

$$r(d) \equiv \bar{r}, \forall d \in d(P).$$

For a medial cone ($K = 2$) or a medial slab ($K = 3$) with radii $\{\bar{r}_k\}_{k=1}^K$ at its corners, for each point $p \in P$,

$$r(d) \equiv \sum_{k=1}^K w_k(p) \bar{r}_k, \forall d \in d_p(P),$$

where $w_k(p)$ is the barycentric coordinates of p on the medial element (edge or triangle).

In contrast to arbitrarily general primitives (as shown in Fig. 3), by utilizing the normal bundle of the ϵ -neighborhood to define the generalized enveloping primitive, our method ensures that the skeleton elements are contained within the primitives. Therefore, the skeleton does not extend outside the input shape or intersect inappropriately with it.

4.2 Medial Skeletal Diagram

Our medial skeletal diagram (MSD) is constructed as a non-manifold triangular mesh with a much-reduced number of elements compared to the medial axis. Each element of this mesh is equipped with a generalized enveloping primitive. Formally,

Definition 4.2 (Medial skeletal diagram). A *medial skeletal diagram* $\mathcal{D}$ is a non-manifold triangulated mesh $\mathcal{M}_S = (V, E, F)$ equipping with a generalized enveloping primitive $\mathcal{E}_P$ for each element $P \in V \cup E \cup F$. $\mathcal{D}$ represents a 3D shape $\hat{S}$ by $\hat{S} = \bigcup_P \mathcal{E}_P$.

Medial skeletal diagram provides a sparse skeleton of 3D shapes while maintaining the ability to accurately capture complex local regions and sharp features. As demonstrated in Fig. 5, under the same accuracy, the traditional medial axis requires 1,056 medial spheres and 2,449 medial slabs to represent the whole shape, resulting in a complex skeleton. By contrast, our method can efficiently and accurately cover the whole hand shape with 11 generalized medial spheres, 5 generalized medial cones, and 4 generalized medial slabs.

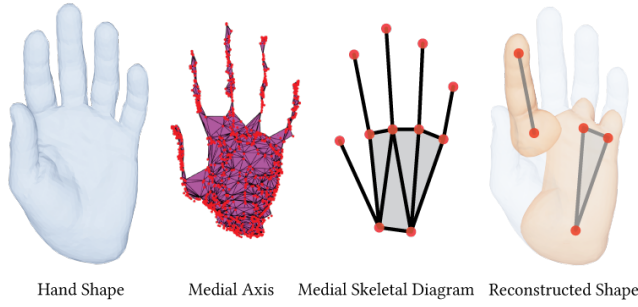


Fig. 5. **Medial skeletal diagram on Hand example.** An illustration of employing our representation for a 3D hand shape. Compared to the medial axis, our medial skeletal diagram has a sparser skeleton while maintaining the ability to accurately capture the detailed geometry. Our medial skeletal diagram offers a *complete* representation for 3D shapes.

Furthermore, our medial skeletal diagram provides a *complete* representation for 3D shapes, incorporating the standard medial axis as a special case. It is capable of representing arbitrary 3D shapes (closed, oriented 2D manifolds, akin to the medial axis requirements). Additionally, it pledges a more versatile representation compared to the standard approach. Under a worst-case scenario concerning the number of primitives needed, our method is not worse than the standard medial axis representation. This is because the upper limit of primitives necessary to fit the mesh in our model equals that in the medial axis, a correlation depicted in Fig. 6. In such instances, while the primitive remains simple, there is a trade-off in the form of an amplified skeletal complexity. Conversely, our diagram can encapsulate the entity with a single vertex for shapes representable through a solitary spherical function, such as the star domain demonstrated in Fig. 6. With the aforementioned advantages, our generalized enveloping primitives offer several benefits:

- Provide a substantially simpler skeleton compared to conventional medial primitives, as demonstrated in Sec. 6.1. This sparse skeletal structure is advantageous in various applications, as highlighted in Sec. 6.2.
- Incorporate a well-researched implicit function formulation, making them suitable for a range of tasks such as shape optimization [Chen et al. 2007], as further detailed in Sec. 6.2.1.
- Compress easily, thereby reducing the storage. In Sec. 6.2.5, we showcase a simple strategy to compress our primitives.

5 Medial Skeletal Diagram Computation

To produce our medial skeletal diagram $\mathcal{D}$ for a given closed manifold triangular mesh $\mathcal{S} = (V_{\mathcal{S}}, E_{\mathcal{S}}, F_{\mathcal{S}})$, we begin with a target shape and its medial axis. Using an arbitrary set of points from the raw medial axis, we deterministically create the medial skeleton as detailed in Sec. 5.1. Subsequently, we fit our generalized enveloping primitives to each vertex, edge, and triangle of the skeleton to match the target shape, as discussed in Sec. 5.2. The collective union of these primitives defines the final shape. This process forms an automated workflow to compute an MSD from any selection of vertices on the input medial axis.

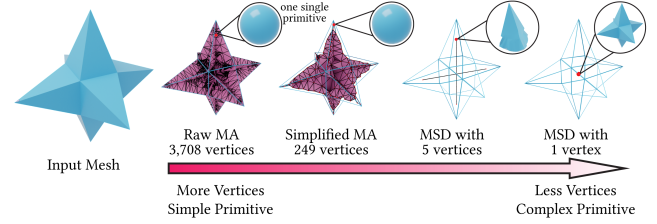


Fig. 6. **Generalization of our MSD.** Our MSD spans a broad range of the skeleton representation spectrum. At one extreme lies the medial axis representation, characterized by simple primitives but intricate skeletons. At the opposite end, we have star-shaped regions that encompass just one complex primitive. The complexity of the MSD skeleton can be modulated by adjusting the fitting capability of each primitive.

Additionally, we introduce an efficient optimization pipeline to compute an optimal MSD. It aims to minimize the count of primitives while capturing the input shape in detail, including both geometry and tessellation. Utilizing the aforementioned procedure, we employ a gradient-free optimizer to determine a vertex set ensuring that the skeleton is simple and the resultant 3D shape aligns with the input shape, as explained in Sec. 5.3. After optimization, the fitted primitives are refined through a feature-preserving refinement process considering both geometry and tessellation to achieve a precise match with the target mesh, as described in Sec. 5.4.

5.1 Medial Skeleton Construction

The idea behind constructing the medial skeleton $\mathcal{M}_S$ of a medial skeletal diagram $\mathcal{D}$ is to perform a “remeshing” of the medial axis by selecting a subset of vertices on the medial axis as constraints. We presume the input medial axis is high-quality and densely sampled. Meanwhile, by limiting the number of constrained vertices, we can maintain a sparse number of discrete elements in $\mathcal{M}_S$.

There are primarily two ways for conducting vertex-constrained remeshing on the non-manifold raw medial mesh: 1) using mesh simplification operations, including vertex removal, edge collapse, and triangle collapse, while maintaining topology preservation through link conditions [Dey et al. 1999]; or 2) constructing a Voronoi diagram restricted to the raw medial mesh and using its dual, restricted Delaunay triangulation (RDT), for remeshing. Although mesh simplification operations can provide a strict guarantee of topology preservation, the remeshing result is affected by the order in which these operations are conducted [Liang et al. 2020]. Consequently, the resulting simplified skeleton from mesh simplification cannot be reduced to a significantly small required number.

Conversely, the RDT method is order-independent but requires additional conditions to satisfy homotopy [Amenta and Bern 1998; Edelsbrunner and Shah 1994; Yan et al. 2009]. In our framework, we opt for the RDT approach, because the number of constrained vertices required to construct $\mathcal{M}_S$ is remarkably smaller than that in the raw medial axis. This means that a significant number of order-dependent operations would be required if the mesh simplification methods were used.

More specifically, given the target mesh $\mathcal{S}$ and its raw medial mesh $\mathcal{M}$, the algorithm of $\mathcal{M}_S$ construction takes a set of points on the medial mesh as inputs, namely, $V \subset \mathcal{M}$. These vertices can

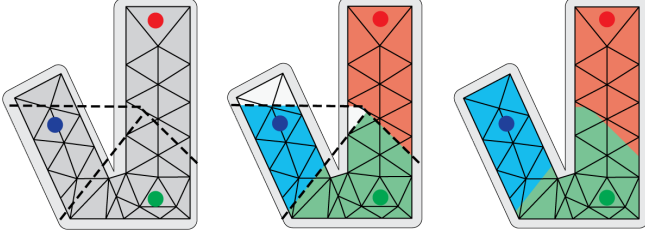


Fig. 7. **Illustration of RVC and its modification.** The initial RVC (bounded by dash lines) may contain more than one connected component (CC) of the medial mesh $\mathcal{M}$. We assign each separate unassigned CC (white regions in the middle figure) to the nearest neighboring CC.

either be manually selected or obtained from the global optimization described in Sec. 5.3. The restricted Voronoi diagram $\text{RVD}(\mathcal{V}, \mathcal{M})$ is then constructed following Eq. (1). We split any element in $\mathcal{M}$ that intersects with a Voronoi face into two parts at the intersection and insert necessary vertices or edges to maintain it as a triangle mesh. Each vertex $\mathbf{v}$ corresponds to a restricted Voronoi cell $\text{RVC}(\mathbf{v}, \mathcal{V})$. Note that the $\text{RVC}(\mathbf{v}, \mathcal{V})$ might have more than one connected component (CC) of $\mathcal{M}$. Similar to [Xin et al. 2022], we modify each RVC to contain only one CC for the use of the following Delaunay triangulation. Specifically, we first perform a breadth-first search (BFS) rooted at $\mathbf{v}$ on $\mathcal{M}$ within each $\text{RVC}(\mathbf{v}, \mathcal{V})$. We then assign the CC expanded by BFS to $\mathbf{v}$. For those CCs not assigned to any vertices, i.e., they do not contain any vertex from the input set, we assign them to their nearest neighboring CC based on the distance between the RVC site and the center of the CC. Fig. 7 shows an example of the RVC modification. Since the raw medial mesh has only one CC for a closed shape, this assignment ensures that every element of $\mathcal{M}$ has an associated $\mathbf{v}$ and that each modified RVC is a single CC.

Then, the medial skeleton is obtained by RDT, which is dual to the RVD. Each $\mathbf{v} \in \mathcal{V}$ corresponds to a volumetric cell, i.e., the modified RVC. Each $\mathbf{e} \in \mathcal{E}$ connects two $\mathbf{v}$ if their corresponding modified RVCs share a common face of $\mathcal{M}$. Each $\mathbf{f} \in \mathcal{F}$ joins the vertices whose corresponding RVCs share a common edge of $\mathcal{M}$. We triangulate the polygonal surfaces of $\mathbf{f}$ to ensure $\mathcal{F}$ only consists of triangle faces. Note that RDT can yield volumetric objects when several RVC cells share a common vertex of $\mathcal{M}$. To address this, we adopt the thinning process described in [Wang et al. 2022a] to convert all tetrahedra in the RDT into two-dimensional sheets, thereby ensuring $\mathcal{M}_S$ does not contain any solid components.

While the standard RDT approach successfully constructs the $\mathcal{M}_S$ for most shapes, there are two additional considerations needed, particularly for shapes of non-zero genus (example shown in Fig. 8): 1) neighboring RVCs can share more than one isolated interface (bold dash lines in Fig. 8); and 2) elements of the $\mathcal{M}_S$ might intersect with the target mesh $\mathcal{S}$, resulting in parts of the element being located outside the mesh.

To handle both situations, we first revise the edges of $\mathcal{M}_S$ and then update the associated RDT triangles. To address 1), we perform a BFS to identify all shared elements of $\mathcal{M}_S$ between every pair of neighboring RVCs, allowing us to locate their isolated interfaces. For each interface, we add a piecewise linear curve to $\mathcal{M}_S$ in $\mathcal{S}$ that originates from and terminates at the two vertices corresponding to

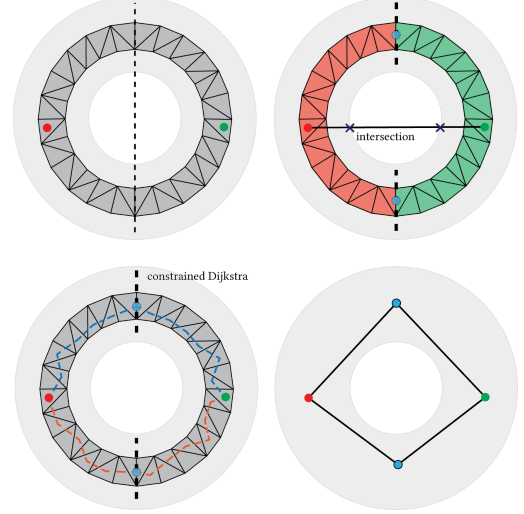


Fig. 8. **Revising RDT for a torus.** The initial RVCs for two vertices of the torus exhibit two interfaces (bold dash lines), and the edge connecting these vertices intersects with the mesh (top right). We add additional vertices at each interface and compute the constrained shortest path (bottom left). The revised RDT (bottom right) remains entirely within the mesh and effectively captures the topology of the shape.

the RVCs and passes through this interface. For situation 2), if an edge intersects with $\mathcal{S}$, we iteratively subdivide the edge until the tessellated edge is in $\mathcal{S}$. For each newly added point, we project it to the shortest path between these two endpoints on $\mathcal{M}$. Upon revising all the edges, we proceed to update the triangles incorporating any edges that have been revised in the previous two steps. Due to the addition of the edges, the triangles change to polygons. Therefore, we then perform triangulation of them. This approach effectively avoids triangle intersections with the target mesh for all the samples we tested in the paper. As an alternative, subdividing intersecting triangles and deforming them to fit inside the mesh could be explored in future work. Note that performing this revision results in a final medial element set for the medial skeletal diagram that may contain more vertices than the initial input set $\mathcal{V}$. We denote the final constructed medial skeleton as $\mathcal{M}_S = (\hat{\mathcal{V}}, \hat{\mathcal{E}}, \hat{\mathcal{F}})$, where $\mathcal{V} \subseteq \hat{\mathcal{V}}$.

5.2 Local Primitive Fitting

Primitive fitting involves discretizing the generalized enveloping primitive $\mathcal{E}_P$ in the space of the directions $d(P)$ for a given medial element P . This is achieved by the discretization of the ϵ -neighborhood boundary ∂P^ϵ , which provides a spatial tessellation for $\mathcal{E}_P$. A piecewise linear triangulated surface, denoted as $\mathcal{T}_{\partial P^\epsilon} = (V_\epsilon, E_\epsilon, F_\epsilon)$, is used to discretize ∂P^ϵ . For meshes scaled to the unit cube, we set $\epsilon = 0.1$. This leads to the spatial discretization of directional vectors $d(P)$ aligning with the vertex normal vectors of $\mathcal{T}_{\partial P^\epsilon}$. We denote these unit discretized directional vectors as $\mathbf{d} \in \mathbb{R}^{|V_\epsilon| \times 3}$.

The discretized generalized enveloping primitive, denoted as $\mathcal{T}_{\mathcal{E}_P} = (V_{\mathcal{T}}, E_{\mathcal{T}}, F_{\mathcal{T}})$, is a triangle mesh derived by equipping $\mathcal{T}_{\partial P^\epsilon}$ with a discretized radius function $\mathbf{r} \in \mathbb{R}_+^{|V_\epsilon|}$. Each vertex of $\mathcal{T}_{\mathcal{E}_P}$ can only move along the direction $\mathbf{d}_i$, with distance indicated by r_i . Here, $E_{\mathcal{T}} = E_\epsilon$, $F_{\mathcal{T}} = F_\epsilon$, and $V_{\mathcal{T}} = \{v_i + r_i \mathbf{d}_i, \forall v_i \in V_\epsilon\}$. Corresponding

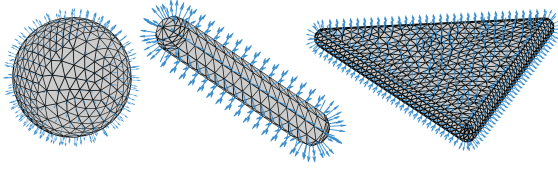


Fig. 9. **Discretized generalized enveloping primitives.** We use triangle meshes to represent the discretized primitives. The directional vectors $\mathbf{d}(P)$ become vertex normals (the blue vectors) in the discrete cases.

to the continuous case as in Def. 4.1, we use $\mathcal{E}(x; \mathcal{T}_{\mathcal{E}_P})$ to denote the implicit function of $\mathcal{T}_{\mathcal{E}_P}$ for any point $x \in \mathbb{R}^3$.

To be more specific, for the three types of medial elements, when we have a medial vertex $\mathbf{v}$, $\mathcal{T}_{\mathcal{E}_P}$ is represented by an isotropically meshed unit sphere centered at $\mathbf{v}$ with radius ϵ . By modifying $\mathbf{r}$, each vertex of $\mathcal{T}_{\mathcal{E}_P}$ moves along the direction extending from the sphere center to the vertex itself. For a medial edge $\mathbf{e}$, we represent $\mathcal{T}_{\mathcal{E}_P}$ using two components: a uniformly meshed cylindrical wall and two end cap hemispheres. For a medial triangle $\mathbf{f}$, $\mathcal{T}_{\mathcal{E}_P}$ consists of three parts: two triangle planes, three half-cylindrical walls, and three spherical wedges. Fig. 9 illustrates the discretized mesh $\mathcal{T}_{\mathcal{E}_P}$ and the discretized directional vectors $\mathbf{d}$ for three cases. In general, the resolution of the primitive meshes is set such that it matches the input mesh resolution. For the generalized enveloping primitives of the medial vertices, we use a mesh with 1,588 vertices. For medial edges and triangles, the mesh resolution is determined by the size of the element. On average, the primitive mesh for an edge has 4,037 vertices and the mesh for a triangle has 1,360 vertices.

The goal of fitting primitive $\mathcal{T}_{\mathcal{E}_P}$ to the target mesh $\mathcal{S}$ is to find an optimal $\mathbf{r}^*$ such that $\mathbf{r}^*$ is smooth across $\mathcal{T}_{\mathcal{E}_P}$ and matches the local region of $\mathcal{S}$. The smoothness prevents the occurrence of excessively large radii along certain directions, guaranteeing the locality of the primitives. We define the following energy optimization problem:

$$\begin{aligned} \mathbf{r}^* = \arg \min_{\mathbf{r} \in \mathbb{R}_+^{|\mathcal{V}|}} & E_{\text{smooth}}(\mathbf{r}) + w E_{\text{expansion}}(\mathbf{r}) \\ \text{s.t. } & \mathbf{r} - \mathbf{r}^{\text{tgt}} \leq \mathbf{0}, \end{aligned} \quad (4)$$

where the energies are defined as

$$E_{\text{smooth}}(\mathbf{r}) = \mathbf{r}^\top \mathbf{L} \mathbf{r}, \quad E_{\text{expansion}}(\mathbf{r}) = \|\mathbf{r} - \mathbf{r}^{\text{tgt}}\|_2^2. \quad (5)$$

In the optimization problem, E_{smooth} penalizes non-smoothness of the radii across $\mathcal{T}_{\mathcal{E}_P}$. We choose $\mathbf{L}$ to be the graph Laplacian matrix of the initial $\mathcal{T}_{\mathcal{E}_P}$. $\mathbf{r}^{\text{tgt}}$ determines how much the primitives grow. $E_{\text{expansion}}$ aims to expand $\mathcal{T}_{\mathcal{E}_P}$ and increase the radii $\mathbf{r}$ to a larger target value $\mathbf{r}^{\text{tgt}}$, while the constraint prevents the penetration of $\mathcal{T}_{\mathcal{E}_P}$ into the target mesh $\mathcal{S}$. $\mathbf{r}^{\text{max}} \in \mathbb{R}_+^{|\mathcal{V}|}$ is the maximal expansion radii of all directions. This is precomputed by measuring the distance from v_i to the first intersection point between the target mesh $\mathcal{S}$ and the ray that originates at v_i with directional vector $\mathbf{d}_i$. In Fig. 10, we show how weight w balances the trade-off between the expansion and the mitigation of exceedingly long radii. In our implementation, we choose $w = 10$ consistently for all the examples.

Eq. (4) is a quadratic programming problem that can be solved using standard techniques, such as the interior point method. To

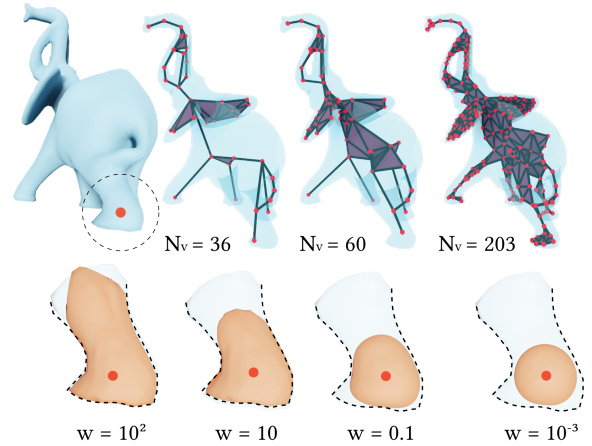


Fig. 10. **Effect of w in local primitive fitting.** The weight w balances the trade-off between expansion and smoothness. An overly large w can induce excessively long radii, especially in tubular shapes, while a notably small w may over-smooth the primitive, failing to capture necessary geometry details (bottom row). As a result, a more smoothed primitive requires more skeleton elements, and vice versa (top row). We choose $w = 10$ in our implementation.

improve performance, we employ an alternating optimization approach inspired by [Bouaziz et al. 2014] instead of a Newton-type solver. For each iteration, we first identify the index set $\mathcal{Z}$ which includes all directions having a radius larger than $\mathbf{r}_i^{\text{max}}$. Then, we optimize the following modified objective:

$$\tilde{\mathbf{r}}^{(t)} = \arg \min_{\mathbf{r}} E_{\text{smooth}}(\mathbf{r}) + w E_{\text{expansion}}(\mathbf{r}) + \sum_{i \in \mathcal{Z}} \Delta^{(t)}(r_i), \quad (6)$$

$$\Delta^{(t)}(r_i) = \hat{w} \|r_i - r_i^{\text{max}}\|_2^2, \quad (7)$$

where $\hat{w}$ is a constant ($\hat{w} = 10^4$) for the contact constraints. We execute up to 15 iterations; however, our solver typically converges in fewer than 5 iterations, with an average fitting time of 0.08 seconds per primitive. Compared to Newton-type optimization methods, the objective is quadratic and can be minimized with a single linear solver, thereby speeding up the optimization process. This fast optimization is achievable thanks to the definition of our primitives, where the deformation is restricted to the direction $\mathbf{d}$. Compared to allowing arbitrary deformations, our approach reduces the DOFs for $\mathbf{r}$ by a factor of three, thereby simplifying penetration constraints. Moreover, the positive $\mathbf{r}$ keeps the skeleton inside the primitive.

In our implementation, the initial radii of enveloping elements are computed based on the closest distance from the shape to their centers. To increase the robustness of primitive expansion, we progressively increase $\mathbf{r}^{\text{tgt}}$ by 10% of the initial radius for each ray direction and solve Eq. (4) at each progression. $\mathbf{r}^{\text{tgt}}$ is increased 25 times in our experiments.

5.3 Global Optimization

In the previous sections, we discussed how the construction of the medial skeletal diagram $\mathcal{D}$ for a given target mesh is determined by the input set of candidate points on the given medial mesh $\mathcal{M}$. We

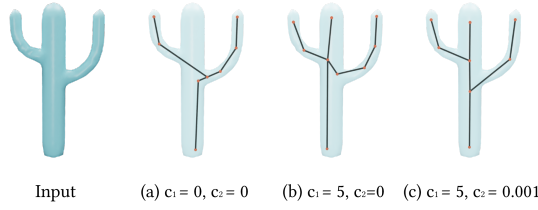


Fig. 11. **Effect of each energy term.** (a) Without any regularization, the skeleton vertices are unevenly distributed along the input medial axis. (b) Incorporating $E_{\text{centrality}}$ addresses this issue. (c) Furthermore, introducing E_{count} reduces the vertex count (from 9 vertices to 8 vertices), leading to a simpler skeleton.

denote these candidate points by V and the resulting medial skeletal diagram by $\mathcal{D}(V) = (\mathcal{M}_S(V), \{\mathcal{T}_{\mathcal{E}_P}\}_{P \in \mathcal{M}_S(V)})$. The problem remains to find an optimal V^* . While existing methods such as Dou et al. [2022], which simplifies a medial axis, might intuitively be adapted to our medial skeletal diagram representation for obtaining optimal candidate points, they face several limitations: Dou et al. [2022] only considers sphere primitives when formulating the linear programming optimization, which is inadequate for cylinders and slabs. As the authors note [Dou et al. 2022], there is an intrinsic trade-off between the number of sphere primitives used and the reconstruction error, which is governed by the dilated radius of each sphere. In our representation, however, every primitive type contributes to the reconstruction of the original shape. Therefore, there is a need for an optimization algorithm that not only considers all primitives but also strives to find a minimal set of primitives necessary for an accurate reconstruction. We present a global optimization scheme that aims to obtain such optimal $\mathcal{D}(V^*)$. The global optimization is formulated as follows:

$$V^* = \arg \min_{V \subset \mathcal{M}} E_{\text{coverage}}(\mathcal{D}(V)) + c_1 E_{\text{centrality}}(V) + c_2 E_{\text{count}}(V), \quad (8)$$

where each energy term is defined as,

$$E_{\text{coverage}}(\mathcal{D}(V)) = \sum_{v \in V_S} \begin{cases} -A(v) & \text{if } \exists P \in \mathcal{M}_S(V), \text{ s.t. } \mathcal{E}(v; \mathcal{T}_{\mathcal{E}_P}) < \delta \\ 0 & \text{o.w.} \end{cases},$$

$$E_{\text{centrality}}(V) = \sum_{v \in V} \|v - c(\text{RVC}(v, V))\|_2^2,$$

$$E_{\text{count}}(V) = ||V| - |\hat{V}||.$$

Here, $A(v)$ represents the normalized vertex area of v with respect to the total surface area of S , $P \in \hat{V} \cup \hat{E} \cup \hat{F}$ is the primitive, and $c(\cdot)$ denotes the centroid of the restricted Voronoi cell.

The objective in Eq. (8) consists of three energy terms. $E_{\text{coverage}}(\cdot)$ measures the target mesh coverage of $\mathcal{D}$. This energy is calculated by enumerating all the vertices of the target mesh S ; if a vertex v is located within a small distance δ ($\delta = 10^{-4}$) from any fitted primitive, it is then considered to be covered by $\mathcal{D}$, and its vertex area contributes negatively to the energy value. $E_{\text{centrality}}(\cdot)$ enforces the uniformity in the distribution of RVCs within the space. This is inspired by Lloyd's algorithm for Voronoi relaxation [Lloyd 1982], which aims to achieve uniformly sized Voronoi cells by iteratively moving each input point toward the center of its corresponding

Voronoi cell. In our context, we strive for a uniformly distributed medial skeletal diagram by penalizing the distance between the input selected point and the corresponding RVC centroid. $E_{\text{count}}(\cdot)$ imposes a penalty for the additional vertices introduced during the construction of the medial skeletal diagram to minimize the complexity of the resulting structure. In Fig. 11, we show that each term plays an important role in regularization. Given the non-differentiability of the objective, we use the Nelder-Mead (NM) algorithm [Richardson and Kuester 1973], a gradient-free optimization method, to optimize Eq. (8) and find the optimal set V^* .

Algorithm 1: Compute Energy

Input: medial axis $\mathcal{M}$, target shape S , and set of points $\bar{V}$

Output: energy value E

```

 $V \leftarrow \bar{V}$  closest-distance projection onto  $\mathcal{M}$ 
Construct the medial skeleton  $\mathcal{M}_S(V)$  (Sec. 5.1)
Fit primitives  $\{\mathcal{T}_{\mathcal{E}_P}\}_{P \in \mathcal{M}_S(V)}$  to  $S$  (Sec. 5.2)
 $E \leftarrow E_{\text{coverage}}(\mathcal{D}(V)) + c_1 E_{\text{centrality}}(V) + c_2 E_{\text{count}}(V)$ 
return  $E$ 

```

To initialize the optimization, we begin with a predefined number of initial candidate points $V^{(0)}$ selected from the medial axis $\mathcal{M}$. The selection is performed by maximizing the shortest path distance on the $\mathcal{M}$ between each pair of points. Then, we use NM simplex algorithm to optimize objective (8) evaluated by Alg. 1. Since the NM simplex algorithm cannot guarantee any trial points are on $\mathcal{M}$, we project the input set of points $\bar{V}$ onto $\mathcal{M}$ at the beginning of each iteration as shown in Alg. 1. The iteration proceeds until 1) the objective (8) no longer decreases for a consecutive of five iterations; 2) the primitives fully cover the target mesh; or 3) the maximal number of iterations is reached.

To speed up the optimization, we incrementally optimize V^* . We configure the initial set $V^{(0)}$ to be small, which can be as few as one vertex ($|V^{(0)}| = 1$). We then optimize Eq. (8) for t iterations, yielding the result $\tilde{V}^{(t)}$. For every t iteration, we check if there are individual regions remaining uncovered. If that is the case, we add a candidate vertex at the volume center for the first n largest regions. The newly added vertices $V_{\text{new}}^{(t)}$ combined with the existing candidate vertices form the final candidate set $V^{(t+1)} = [\tilde{V}^{(t)}; V_{\text{new}}^{(t)}]$. The iteration proceeds until the stopping criteria are met. We only optimize $V_{\text{new}}^{(t)}$ while freezing the optimized $\tilde{V}^{(t)}$ for performance consideration. In our implementation, we set $c_1 = 1$ and $c_2 = 10^{-3}$ consistently for all the examples. Fig. 12 shows the intermediate results and energy convergence curves for three examples.

5.4 Feature-Preserving Refinement

Our optimization pipeline produces a medial skeletal diagram with maximal coverage of the target mesh. As we discretize the generalized enveloping primitives to a triangle mesh, however, despite the radii of primitive mesh vertices might align with the target mesh, the triangle faces of the primitive mesh could still significantly deviate from the target mesh due to the differences in tessellations of the primitive mesh and the target mesh. To address this, we propose

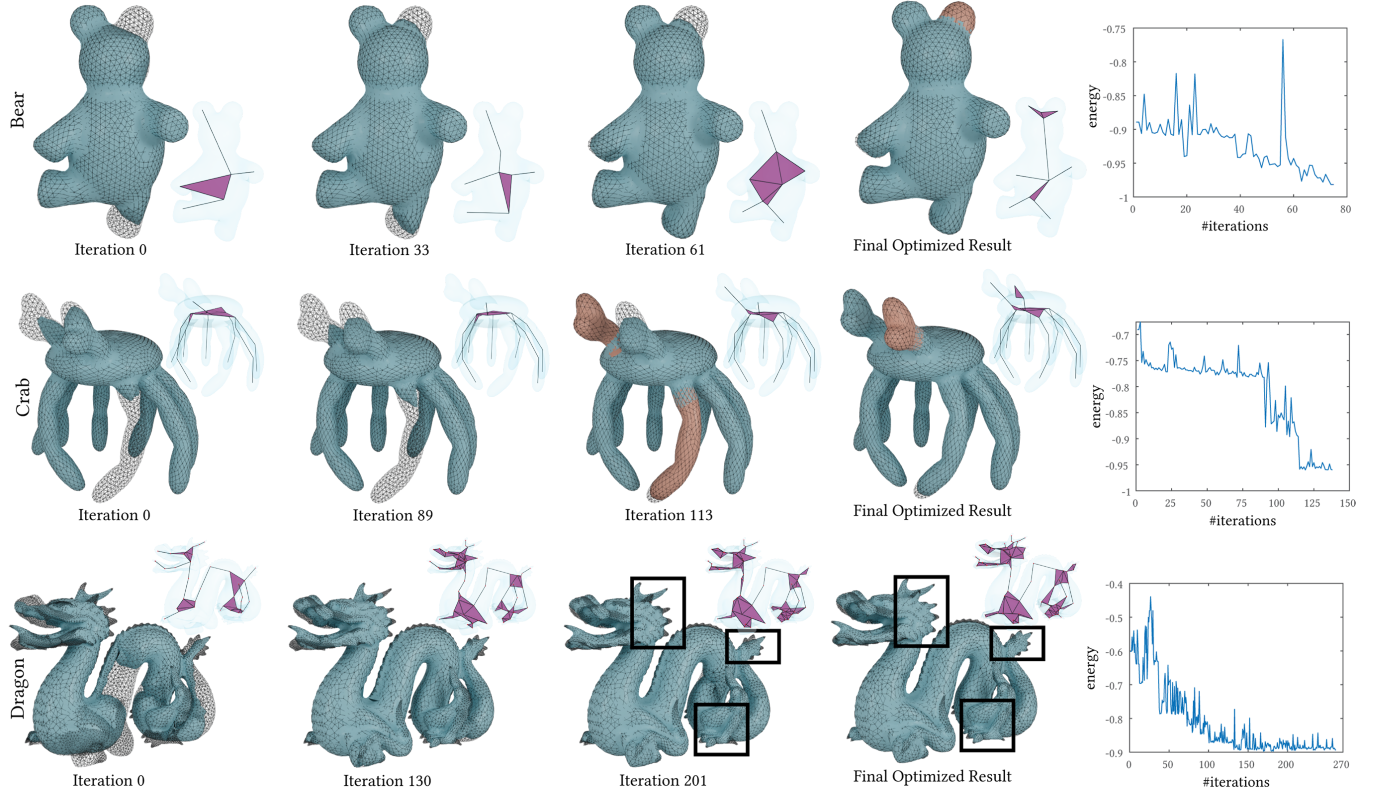


Fig. 12. **Intermediate results and energy convergence curves of the global optimization.** The global optimization is framed to obtain an optimal medial skeletal diagram that strives to maximize target mesh coverage while minimizing the number of primitives used. A gradient-free optimization method is carried out to efficiently optimize the energies. During the optimization, we incrementally add new points to each isolated uncovered volume and fit primitives (shown in the orange color). The geometric details such as dragon horns, tails, and claws are successfully captured, as highlighted by the rectangles. Note that energy spikes are encountered due to the introduction of new trial points by the NM optimizer, as it does not ensure a consistent decrease in energy.

a *robustness* refinement process that allows our representation to “exactly” match the target mesh in terms of geometry and “almost exactly” mirror its tessellation. By “almost exactly,” we indicate that the tessellation of the target mesh is entirely encompassed within the refined primitive. Moreover, our refinement method produces post-refinement primitives with *consistency*, which satisfy the definition of generalized enveloping primitive, retaining the advantages of these primitives as discussed in Sec. 4.1. Guaranteeing consistency filters out many trivial refinement methods; e.g., simply dilating the primitive and boolean-intersecting with the target mesh may violate the consistency, as the resulting primitive may exhibit multiple radius values along one direction for a profoundly concave shape.

The core strategy is to refine each fitted generalized enveloping primitive mesh so that both its radius function and tessellation in the regions that are close to the target mesh are precisely matched with the target mesh. To achieve both consistency and robustness, our refinement process consists of four steps, as shown in Fig 13. For each fitted generalized enveloping primitive $\mathcal{T}_{E_p}$, 1) we initially identify the sub-surface of the target mesh that resides within a δ distance to $\mathcal{T}_{E_p}$ and project its tessellation onto $\mathcal{T}_{E_p}$. This step ensures that the tessellation of the sub-surface, including vertices,

edges, and triangles, is fully encompassed within $\mathcal{T}_{E_p}$. 2) We then remove redundant edges that do not contribute to any geometric and tessellation features of the target mesh. 3) Next, we update the vertices positions of $\mathcal{T}_{E_p}$ such that the geometry of the sub-surface exactly aligns with the target mesh. Because the primitive mesh encompasses the tessellation of the target mesh, updating the vertex position makes the primitive mesh match the target mesh exactly in the selected sub-surface. 4) We finally carry out a mesh clean-up by removing small edges and triangles without affecting the matched geometry and tessellation. We implement the above four steps using exact rational arithmetic to ensure the robustness of our process. For more detailed information on the implementation, we direct readers to consult the supplementary materials. In Fig. 14, we show three example primitives undergoing the refinement process. One benefit of our post-refinement is that it aligns the tessellation of each primitive with the corresponding local region of the input surface mesh. Even when using primitives with a dense meshing resolution during the local primitive fitting stage, the resolution of the post-refined primitives is determined solely by the resolution of the input mesh. This allows for the use of densely meshed primitives in the local fitting stage to capture intricate geometric details of the input

mesh while ensuring that the resolution of the final reconstructed mesh closely matches that of the input mesh, avoiding unnecessary complicated continuous parameters in MSD.

6 Experiments

Our experiments are run on a desktop with an AMD Ryzen 9 5950X 16-core CPU and 64GB RAM. We use GMP, MPFR, and CGAL for low-level arithmetic and geometric kernels. To demonstrate the effectiveness and robustness of our method, we evaluate our algorithm on 100 closed manifold triangle meshes. They are sourced from V-HACD dataset [Mamou et al. 2016], AIM@Shape Repository [Cohen-Steiner and Morvan 2003], and the McGill 3D shape benchmark [Fang et al. 2008]. The size of all shapes is normalized to the $[0, 1]$ range. For all shapes in our benchmark, we apply the same set of parameters to the local fitting (mentioned in Sec. 5.2). For the global optimization, we set the initial number of candidate vertices $|V^0| = 10$. After the first 50 iterations, we add either three new vertices or five new vertices every 30 iterations ($n = 3$ or $n = 5$), depending on which one gives the better result. The coefficient for each energy term is mentioned in Sec. 5.3. All the experimental results are conducted using a consistent parameter setting. We choose to use the simplified medial axis obtained from [Dou et al. 2022] as the initial start of medial skeletal diagram construction to strike a balance between computational speed and skeleton accuracy. Note that our approach is not strictly tied to this choice, but can readily accommodate other variants of medial axis representations as inputs. Statistics of the computational cost required to construct our representation are shown in Fig. 15 and Table 1.

Baselines. We include six baselines: (1) MAT [Amenta and Bern 1998], the raw medial axis transformation; (2) MATFP [Wang et al. 2022a], a state-of-the-art technique for computing the MAT while preserving features; (3) LS Skeleton [Bærentzen and Rotenberg 2021], a curve skeletonization approach grounded in local separators; (4) Coverage Axis (CA) [Dou et al. 2022], a shape skeletonization method that simplifies from MAT; (5) Point2Skeleton (P2S) [Lin et al. 2021], a deep learning-based method for constructing medial meshes from point clouds; (6) CoACD (ACD) [Wei et al. 2022], a state-of-the-art method for convex decomposition.

Metrics. Following [Dou et al. 2022; Lin et al. 2021; Wang et al. 2022a], we use the two-sided average vertex-to-surface distance error, denoted as $\bar{\epsilon}$, to evaluate the surface reconstruction accuracy using medial meshes. $\bar{\epsilon}^1$ is the one-sided average vertex-to-surface distance from the original surface to the reconstructed surface and

$\bar{\epsilon}^2$ is the distance in the reverse direction. We also report the number of medial elements, including the vertices $\#v$, edges $\#e$, and triangles $\#f$, to compare the sparsity of discrete elements in different representations. To offer a more complete overview, we detail the number of continuous parameters ($\#c$) associated with different representations. For baseline methods, we report $\#c$ as the total degree of freedom of medial spheres. It is computed by the multiplication of the number of medial vertices in the skeleton by four, as each medial vertex is equipped with a radius (one scalar) and a 3D position (three scalars). For our method, we report the total number of directional vectors for all primitives plus the number of medial vertices multiplied by three (each vertex has a 3D position). To ensure a fair comparison of reconstruction error, in the case of MAT, we increase the target mesh resolution without changing its geometry to obtain a skeleton that has a comparable magnitude of $\#c$ to that of our method. For MATFP, LS, CA, and P2S, following their original papers, we reconstruct the mesh from the skeleton using medial spheres, cones, and slabs. In the case of ACD, where there is no reconstruction involved, we only report the number of decomposed domains (represented by $\#v$) and $\#e$ for every pair of adjacent domains. The number of unique edges $\#e$ of MATFP and the number of faces $\#f$ of LS Skeleton are omitted from the table as these counts are zero across all shapes. We consider an edge unique if it is not a triangle edge in MATFP. Since LS only produces curve skeletons, it does not contain any triangles.

6.1 Comparisons and Discussions

The quantitative comparisons between our method and the baselines, conducted on the 100 meshes, are presented in Table 2. Our method uses the fewest skeletal elements and achieves the lowest two-sided mean distance error relative to all other methods examined. Despite the increased number of continuous parameters in our representation, as detailed in subsequent sections (Sec. 6.2.1 and 6.2.5), this does not introduce complications but rather aids in finding a better solution. Our mean reconstruction error is 10% of MATFP and 7.8% of MAT, the two methods yielding the least reconstruction error among all baselines. While MATFP employs 91.0% fewer continuous parameters with reconstruction errors that are 10 times greater compared to our method, increasing the number of MATFP’s continuous parameters to roughly match those of our method—by densely sampling the input mesh—only reduces its reconstruction errors by 60%, which is still 6 times higher than that of our approach. On the other hand, despite that MAT uses more continuous parameters, it fails to seamlessly align with the target mesh given a finite number of medial primitives.

The simplicity of our representation’s discrete parameters enhances the computational speed of shape reconstruction. To highlight this advantage, we compared the shape reconstruction running time of our representation with those of MATFP. We choose SDF with a 256-resolution voxel grid and marching cubes to perform shape reconstruction in both methods, because unioning primitives on SDF for MATFP is much faster than on meshes. For MATFP, the SDF value of each voxel is determined by calculating the minimal distance to all medial vertices and subtracting the radius of the corresponding medial sphere. For ours, we calculate the SDF

Table 1. Average Timing for global optimization and refinement.

Per Iter.	Projection	0.006s
	MSD Construction	4.12s
	Primitive Fitting	4.64s
	Energy Evaluation	2.71s
	Total Global Opt.	11.57s
	Total Global Opt. #iter.	51
	Refinement	10.78s

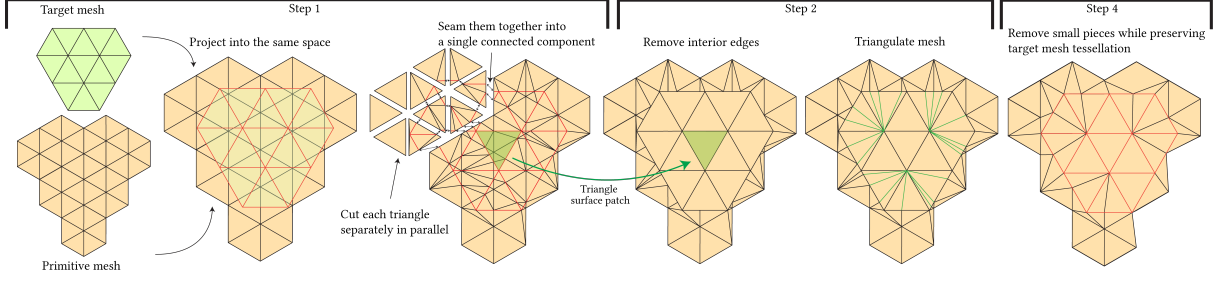


Fig. 13. **The procedure of our feature-preserving refinement.** The process consists of four steps. In the first step, we refine the primitive mesh such that it contains the tessellation of the target mesh (step 1). Next, we remove redundant edges in each triangle surface patch and produce a triangulated primitive mesh (step 2). Then, the vertex positions are updated based on their primitive type (step 3). Finally, we clean up the small triangles and edges as much as possible while preserving the target mesh tessellation (step 4). Here for 2D illustration, we omit step 3 and refer the readers to Fig. 14 for a 3D example.

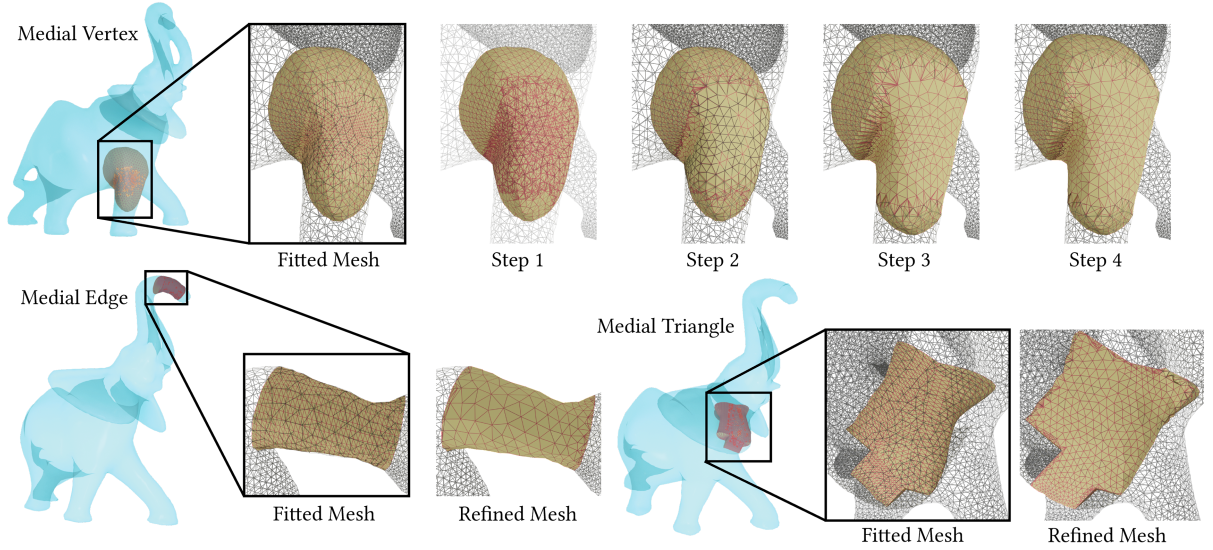


Fig. 14. **Results of our feature-preserving refinement on primitive meshes of all three types.** In the first row, we show the step-by-step intermediate results obtained during the process of refining a medial vertex primitive mesh. The primitive mesh undergoes refinement aligning with the target mesh in terms of both geometry and tessellation. The second row offers additional examples featuring the refinement process for the medial edge primitive mesh and medial triangle primitive mesh.

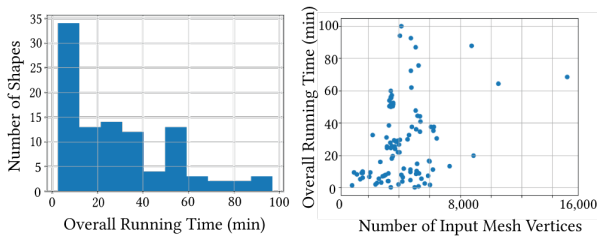


Fig. 15. **Statistics of the computational cost for constructing medial skeletal diagram from an input shape,** including a histogram of overall running time across the 100 shape benchmark and the complexity of input mesh v.s. running time. The computational time is principally influenced by the input mesh size (impacting feature-preserving refinement) and the shape complexity (impacting global optimization).

value by querying the minimal value among all closest distances to all primitive meshes for each voxel. We conduct this comparison using six shapes selected from the benchmarking dataset of 100

shapes, specifically choosing those with the highest numbers of primitives in our representation. The results show that the average reconstruction speed of our method is 2.4× faster MATFP (our method: 17.5s vs. MATFP: 41.1s). Moreover, our method achieved 40% fewer reconstruction errors (our method: 6.9×10^{-4} vs. MATFP: 1.1×10^{-3}).

To demonstrate the effectiveness of our feature-preserving refinement, we also measure the two-sided mean distance error of the fitting-only results. Among 100 shapes, our fitting-only reconstruction reports 0.22 error, 7× larger than the error after the refinement post-processing. Nonetheless, it is still smaller than the errors of all existing methods. Furthermore, we also measure the two-sided Hausdorff distance between the input shape and the reconstructed shape. Our representation reports 1.81 on average over 100 shapes, lower than the mean distance error of both LS and P2S. Moreover, ours is also lower than the two-sided Hausdorff distance of both CA and MATFP, which are 3.64 and 2.22, respectively.

Table 2. **Quantitative comparisons on the number of discrete and continuous parameters in the representation and shape reconstruction error on the benchmark of 100 shapes.** We count the number of vertices ($\#v$), the number of edges ($\#e$), the number of triangles ($\#f$), and the sum of them ($\#d$) for discrete elements. In addition, we provide the number of continuous parameters in row $\#c$. For the reconstruction errors, $\bar{\epsilon}^1$ is the one-sided average vertex-to-surface distance from the original surface to the reconstructed surface, $\bar{\epsilon}^2$ is the distance in the reverse direction. $\bar{\epsilon}$ is the two-sided average vertex-to-surface distance error. We compare our method with six existing methods: MAT, MATFP (MFP), CoACD (ACD), LS Skeleton (LS), Point2Skeleton (P2S), and Coverage Axis (CA). Our medial skeletal diagram demonstrates superior sparsity on discrete elements and the highest reconstruction accuracy compared to other representations.

		MAT	MFP	ACD	LS	P2S	CA	Ours
$\#v$	Min	17,955	692	3	4	100	57	3
	Max	37,705	4,119	192	195	100	738	277
	Avg	32,539	2,110	27	57	100	229	26
$\#e$	Min	0	–	2	3	0	0	0
	Max	144	–	639	192	35	131	81
	Avg	15	–	57	57	3	18	9
$\#f$	Min	31,042	1,445	–	–	136	0	0
	Max	82,297	10,232	–	–	346	1,437	457
	Avg	63,279	4,835	–	–	233	385	24
$\#d$	Min	95,833	6,945	84	114	337	632	58
	Max	–	–	–	–	–	–	–
$\#c$	Min	71,820	2,768	–	16	400	228	4,594
	Max	150,820	16,476	–	780	400	2,952	497,770
	Avg	130,154	8,440	4,423	229	400	918	94,168
$\bar{\epsilon}^1$	Min	0.06	0.044	–	0.59	0.76	0.17	0
	Max	3.09	3.18	–	24.34	2.92	12.37	0.41
	Avg	0.31	0.31	–	4.42	1.31	0.54	0.029
$\bar{\epsilon}^2$	Min	0.11	0.029	–	0.39	1.14	0.14	0
	Max	0.48	0.68	–	10.03	10.35	0.83	0.098
	Avg	0.24	0.15	–	2.54	2.22	0.34	0.008
$\bar{\epsilon}$	Min	0.40	0.31	–	4.43	2.22	0.54	0.031
	Max	(+129%)	–	–	(+1324%)	(+615%)	(+74%)	(–90%)

To investigate the number of parameters used in each medial axis methods compared with our MSD, we measure “sparsity” of the parameters by the compression ratio of the input mesh size ($\#v \times 3 + \#t$) to the number of representation parameters ($\#c + \#d$ defined in Table 2) and “completeness” by the reconstruction error. Our representation strikes a trade-off between “sparsity” and “completeness”. This trade-off can be visualized as a curve (Fig. 16): At one extreme, our MSD achieves a low reconstruction error (0.031) at the sacrifice of compression ratio (0.9). To achieve such reconstruction accuracy, existing methods either require more DoFs or incur higher errors with the same DoFs as ours (such as Fig. 1 (b)). Furthermore, for MSD, when achieving a reconstruction error similar to MATFP (0.31) our compression ratio is more efficient (6.6 vs. 1.9 for MATFP). However, the advantage diminishes when the compression ratio exceeds 8, where CA achieves better accuracy. This occurs because, at high compression ratios (where both $\#d$ and $\#c$ are limited), the reconstruction accuracy depends more on the skeleton complexity. Since our MSD’s skeleton is relatively sparse, it results in a higher reconstruction error under these conditions.

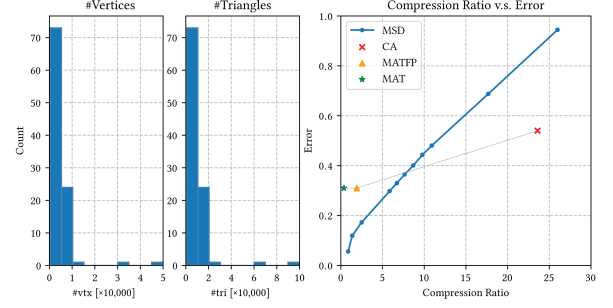


Fig. 16. **Statistics of the compression ratio v.s. reconstruction accuracy.** The left panel shows histograms illustrating the distribution of the number of vertices ($\#vtx$) and the number of triangles ($\#tri$) across the triangle meshes of the 100 shapes used in the experiments. The right panel presents a curve illustrating the relationship between the compression ratio and reconstruction accuracy. Our method achieves superior reconstruction accuracy when the compression ratio is less than 8.

In addition to quantitative comparisons, we also show a set of qualitative results on various shapes from these 100 meshes in Fig. 17 and 18. Compared with other baselines, our representation produces visually simpler skeletons and more accurate reconstruction.

Comparison with Medial Axis Transform. We compare our method with MATFP [Wang et al. 2022a] in the context of the quality of the reconstructed mesh. MATFP is highly regarded for its remarkable precision and is tailored to preserve external features such as sharp edges and corners of the input mesh surface. This algorithm is essentially a superior variant of the medial axis transforms. MATFP requires two input thresholds for controlling sharp feature detection. We use the default parameters provided within the official source code for all 100 shapes. Our method demonstrates a better performance with MATFP regarding reconstruction accuracy. This is attributed to our refinement process which exactly matches the geometry and tessellation of the reconstructed shape from our method with the input mesh – a feature that is notably missing in existing medial axis representations including MATFP. What truly distinguishes our method is its ability to achieve these results using significantly fewer medial elements. This simplicity is largely due to our utilization of generalized enveloping primitives. Thus, while maintaining a high level of accuracy akin to MATFP, our method outperforms it in terms of the skeleton simplicity, with a reduction in the number of medial elements by two orders of magnitude.

Comparison with Simplified Skeleton-based Methods. We consider two simplified skeleton-based methods: Coverage Axis [Dou et al. 2022] and Point2Skeleton [Lin et al. 2021]. Both methodologies are based on a similar process that simplifies the raw medial axis to preserve the geometric and topological features of the input mesh surface. Coverage Axis employs Mixed Integer Linear Programming (MILP) to minimize the count of medial vertices while ensuring comprehensive coverage of all the input mesh surface points. We use scaling offset, whose value is set by binary search to ensure the number of vertices is about 400. This tactic results in superior reconstruction accuracy when compared to other baseline methods, owing to its explicit optimization for this particular objective.

On the other hand, Point2Skeleton adopts a data-driven strategy, leveraging deep neural networks to predict skeletal points and their interconnections. As it is a data-driven method, however, it does not generalize to arbitrary input shapes. The output of a neural network necessitates a fixed number of vertices (100 in the official implementation), which can pose challenges when generalizing to unseen shapes that deviate from the training dataset. Both simplified skeleton methods rely on spheres, cones, and slabs for surface reconstruction. The primary objective of these methods is to optimize point coverage, so inadvertently introduce errors in the surface reconstruction process. This stems from the nature of these primitives - their geometric shapes are fixed and lack the flexibility to adapt to arbitrary regions, which can result in an inaccurate approximation of the original mesh surface. By contrast, our approach leverages the mesh representation of generalized primitives and benefits from the refinement technique. Neptune and Seahorse in Fig. 17 are two evident examples.

Comparison with Curve Skeletons. LS Skeleton [Bærentzen and Rotenberg 2021] is a state-of-the-art method for curve skeletonization. Notable for its simplicity and smoothness in representing shapes, LS Skeleton shares a common drawback with other curve skeleton-based methods: it struggles to accurately represent large flat regions present in the input shape. This is particularly problematic for CAD models that frequently feature such flat regions. Screwdriver and Fan disk example in Fig. 17 provide evidence. In this case, a noticeable advantage of our approach is its ability to represent an entire rectangular region with a few triangles while maintaining high reconstruction accuracy. By contrast, LS Skeleton falls short in this regard.

Rather than explicitly reflecting detailed geometry features as curve skeletons/MAT, our skeleton captures topological structures while primitives effectively represent local geometry details of the shape, as determined through global optimization.

6.2 Applications

In this section, we attempt our representation through several applications, demonstrating the simplicity of the discrete elements and the expressiveness of the generalized enveloping primitives.

6.2.1 Shape Optimization. In the previous sections, we have shown the remarkably sparse discrete structure of our medial skeletal diagram, which is a direct result of our design intention - to shift complexity from discrete to continuous elements. In this application, we further demonstrate that the added complexity of these continuous elements does not impede the usability and applicability of our representation. Conversely, it gives a better-converged compliance across all examples compared to the medial axis representation, i.e., each domain being a sphere, a cone, or a slab.

To substantiate this, we apply our representation to a topology-constrained shape optimization task. This task takes an initial 3D skeletal graph, which specifies the topology of the shape, and optimizes the shape to achieve a predefined objective while maintaining the same topological structure. We choose the objective to be compliance minimization, a well-known topology optimization task. The goal of this task is to optimize a shape that can withstand input load

with minimal compliance given the maximal allowed material usage. Conventionally, such optimization is conducted within the space of a voxel grid, where each voxel carries a binary value to denote the presence of the material. Here, we re-frame the optimization within the space of our representation. We start by taking the initial 3D skeletal graph as the discrete structure of our medial skeletal diagram and constructing initial generalized enveloping primitives upon this discrete structure. The optimization variables are the continuous parameters of our medial skeletal diagram, which include the positions of the vertices and the radii of each primitive. Fig. 19 and 20 show the experimental setup.

The optimization problem is formulated as follows:

$$\begin{aligned} \arg \min_{\hat{\mathbf{v}}, \hat{\mathbf{r}}} E(\mathbf{u}, \hat{\mathbf{v}}, \hat{\mathbf{r}}) &= \frac{1}{2} \mathbf{u}^T \mathbf{K}(\mathbf{X}) \mathbf{u}, \\ \text{s.t. } \mathbf{K}(\mathbf{X}) \mathbf{u} &= \mathbf{f}, \\ \mathbf{X}_i &= \theta(x_i; \hat{\mathbf{v}}, \hat{\mathbf{r}}), \forall x_i \in \Omega \\ \sum \mathbf{X}_i &\leq V_{\text{vol}} \cdot |\Omega|, \\ \mathbf{u}_{\text{BC}} &= 0, \end{aligned}$$

where $\hat{\mathbf{v}}$ is the concatenation of the medial vertex positions, which in turn determines the locations of generalized enveloping primitives, $\hat{\mathbf{r}}$ is the concatenation of the radius values $\mathbf{r}$ of all primitives, $\mathbf{X}_i$ is the density of voxel x_i in the given domain Ω , $\mathbf{K}(\cdot)$ is the tangent stiffness matrix of a linear elastic material, $\mathbf{u}$ is the displacement of the shape under external force $\mathbf{f}$, $\mathbf{u}_{\text{BC}}$ is the displacement of the grid points on the boundary condition, and V_{vol} is the desired volume fraction of the final shape relative to the total voxel space. We refer the readers to Chen et al. [2007] and Kumar [2023] for the definition of $\mathbf{K}(\cdot)$ and other standard aspects of the compliance minimization problem. Note that $\theta(x_i; \hat{\mathbf{v}}, \hat{\mathbf{r}})$ bridges the continuous parameters of our medial skeletal diagram with the voxel grid space, defined as

$$\theta(x_i; \hat{\mathbf{v}}, \hat{\mathbf{r}}) = H\left(f\left(\bigcup_k -\mathcal{E}_k(x_i; \mathcal{T}_{\mathcal{E}_p}(\hat{\mathbf{v}}, \hat{\mathbf{r}}))\right)\right).$$

Here, $\mathcal{E}_k(\cdot; \mathcal{T}_{\mathcal{E}_p}(\hat{\mathbf{v}}, \hat{\mathbf{r}}))$ is the implicit function (as defined in Sec. 5.2) of the k -th generalized enveloping primitive $\mathcal{T}_{\mathcal{E}_p}$ constructed using $\hat{\mathbf{v}}, \hat{\mathbf{r}}$. $f(\cdot)$ refers to Rvachev disjunction function [Chen et al. 2007] converting boolean compositions into a real-value function, where the sign of the output value indicates the boolean value of the input compositions: $f(\phi_s \cup \phi_t) = \phi_s + \phi_t + \sqrt{\phi_s^2 + \phi_t^2}$. $H(\cdot)$ is Heaviside step function. In essence, $\theta(x_i; \hat{\mathbf{v}}, \hat{\mathbf{r}})$ outputs a value of one for voxels within the union of primitives, and zero otherwise. We solve the optimization using gradient descent. Appendix A provides details of the gradient calculation.

We conduct two sets of experiments using the aforementioned formulation. The first set is to optimize shapes under the same boundary conditions and external forces within a voxel space of $60 \times 40 \times 6$ while maintaining topology constraints using four distinct user-defined skeletal graphs built upon a shared set of vertices. The input value V_{vol} is set to 0.2. The optimization successfully finds the minimum in all four scenarios, with each converging to a valid solution that matches the topology of the corresponding input skeletal graph. The optimized shapes, along with their compliance values,

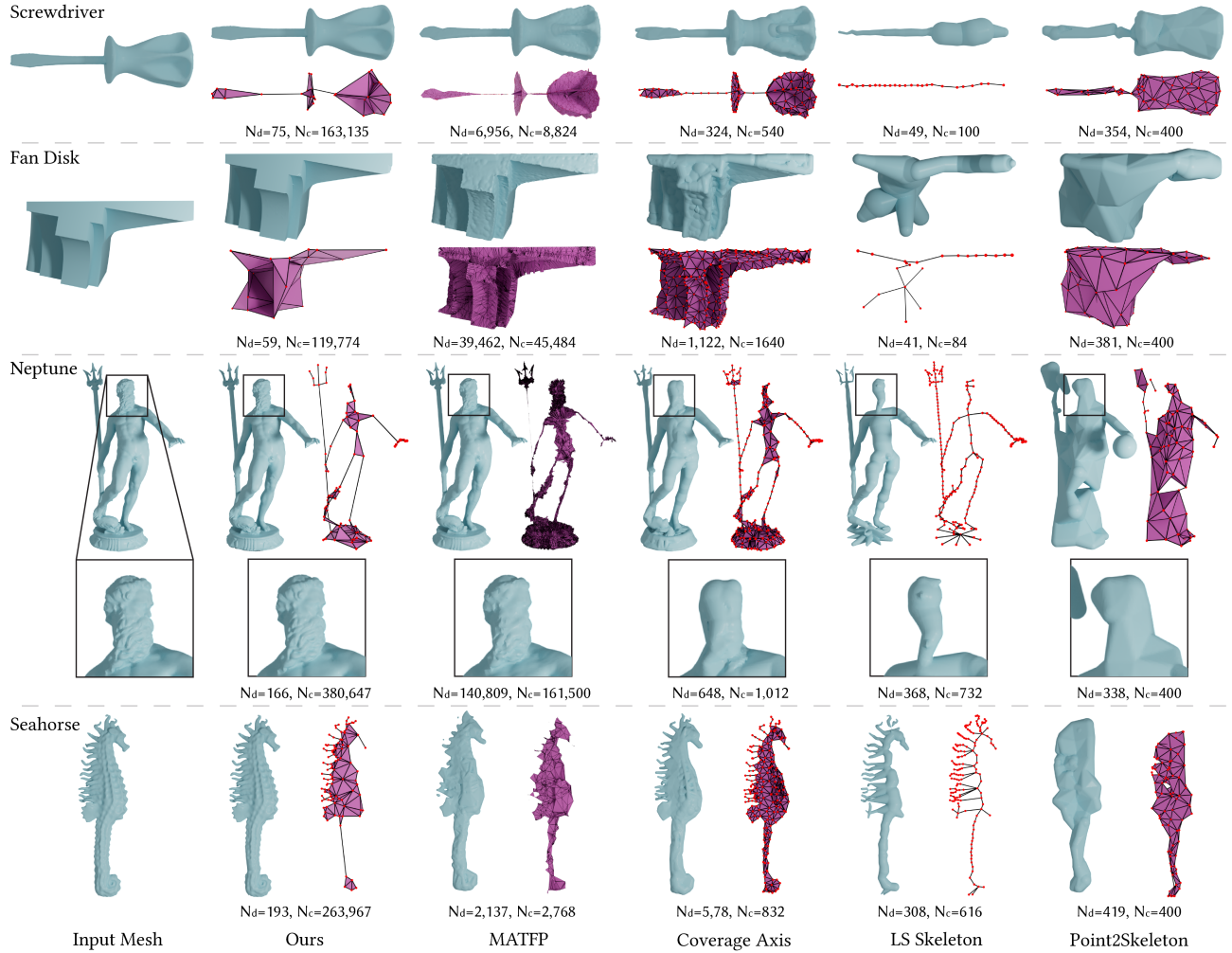


Fig. 17. **Qualitative comparison of the skeletal representation and reconstructed mesh (part 1/2)**, involving four baseline methods and our proposed method. We omit the visualization of medial vertices for MATFP due to their excessive quantity. Our method achieves the highest qualitative reconstruction accuracy by using a relatively small number of discrete elements (N_d), albeit at the cost of using a larger set of continuous parameters (N_c).

are shown in Fig. 19. Moreover, all minimizers from our representation are lower than those using the medial axis representation, with an average 14% lower compliance value. The second experiment optimizes a shape within a larger voxel space $60 \times 60 \times 40$ using a more complex skeletal structure as input. The input value V_{vol} is set to 0.05. The optimizer also successfully yields a structure with minimal compliance ($E = 4.42$), versus minimal compliance values 6.17 from the medial axis representation. The final result is shown in Fig. 20. These experimental results demonstrate the capability of our representation to perform shape optimization with user-defined topology constraints, a feature that is non-trivial to incorporate for voxel representation in a typical topology optimization setup. Further, it also produces superior results compared to existing work with medial axis representation [Bell et al. 2012]. In our implementation, we use $|\hat{\mathbf{r}}_k| = 8$ for a medial vertex and $|\hat{\mathbf{r}}_k| = 12$ for a medial edge, which results in a total of approximately 250 DOFs in both examples – much smaller than the total number of voxels.

6.2.2 Shape Generation. We use the cut-and-paste method employed in existing shape modeling techniques [Pandey et al. 2022] to generate new shapes from a given set of input shapes. We first construct the medial skeletal diagram for each shape in the input set. We then build a corpus of sub-skeletons by cutting subgraphs from these skeletons. To generate new shapes, we assemble these sub-skeletons, adding edges between them to ensure the resulting skeleton is a single connected component. New shapes are then generated by pasting and uniting the fitted generalized enveloping primitives from the input shapes. We use mesh boolean operations implemented from libigl to unite the primitives [Jacobson et al. 2018]. Using this process, we generate shapes from PartNet [Mo et al. 2019] and leverage the provided first-level part labels to perform the cutting on the skeleton. Fig. 21 illustrates the generated shapes for chairs and tables. This process of shape generation is remarkably straightforward and efficient, yet it ensures diverse changes in topology and geometry. In Fig. 21, we also showcase several generated

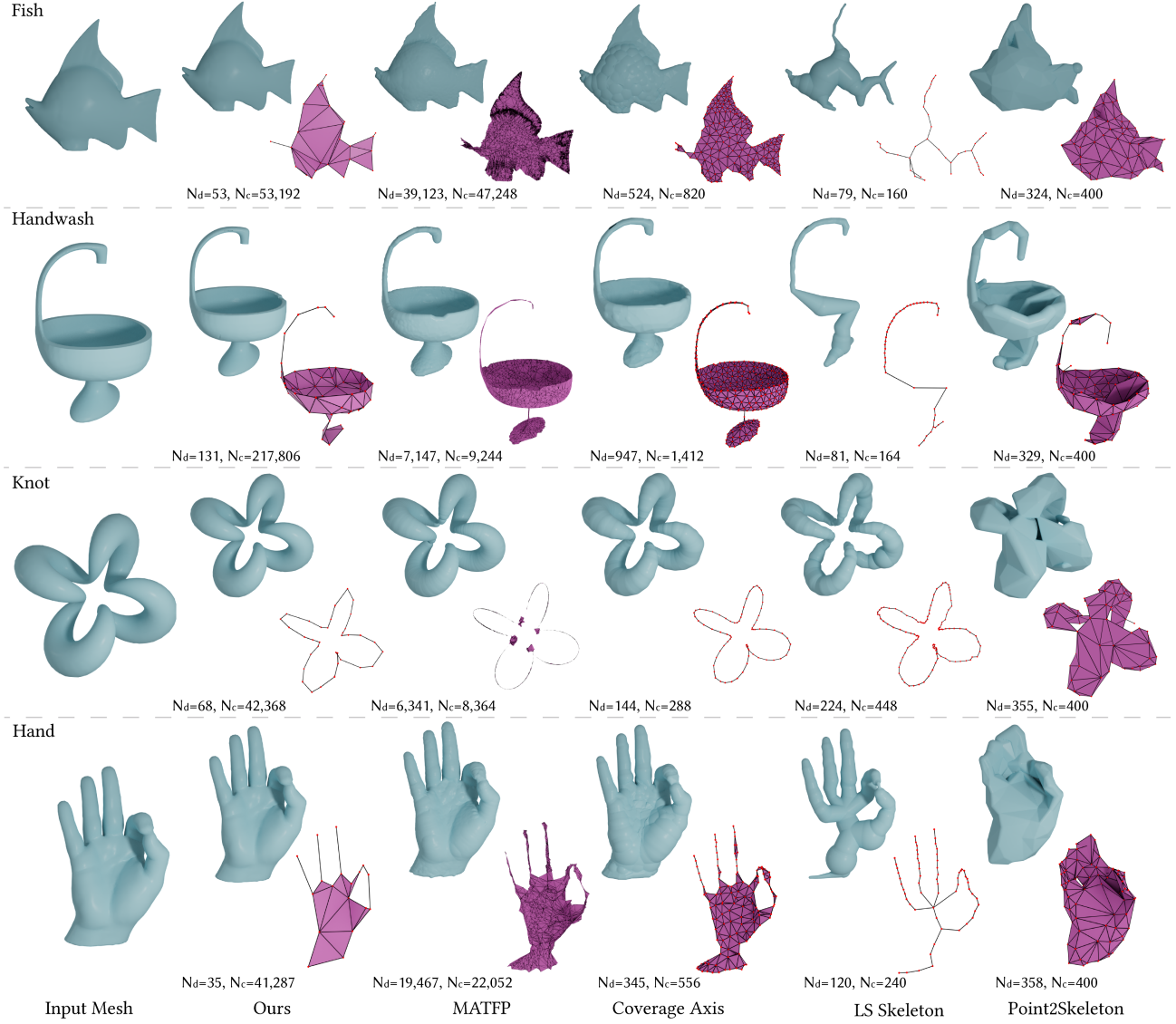


Fig. 18. **Qualitative comparison of the skeletal representation and reconstructed mesh (part 2/2)**, involving four baseline methods and our proposed method. We omit the visualization of medial vertices for MATFP due to their excessive quantity. Our method achieves the highest qualitative reconstruction accuracy by using a relatively small number of discrete elements (N_d), albeit at the cost of using a larger set of continuous parameters (N_c).

shapes derived from the 100 shapes benchmark. This streamlined process is made possible by the simplicity of the skeleton combined with the rich expressiveness of the primitives in our representation. In contrast, the medial axis representation demands a complex graph to accurately reconstruct the surface. Consequently, executing straightforward algorithms like the cut-and-paste method becomes challenging and necessitates considerable manual effort.

6.2.3 Mesh Decomposition. Due to the simplicity of our skeleton and high accuracy in reconstructing the shape, our representation can also be used to perform domain decomposition of the shape. Following the method from [Lin et al. 2021], we first identify the

skeleton vertices that encounter a dimensional change on the neighboring elements or that are non-manifold vertices and then decompose the skeleton at these vertices. As shown in Fig. 22, we can generate a meaningful and concise decomposition of the mesh, while other methods fail to do so.

Moreover, we compare the number of elements in our representation with the state-of-the-art domain decomposition method, CoACD [Wei et al. 2022]. Although these two methods do not share the same objective, a comparison can still be illuminating. Specifically, CoACD aims to decompose shapes into convex domains by cutting meshes with 3D planes, whereas our approach works by performing a union of all fitted primitives to represent the shape.

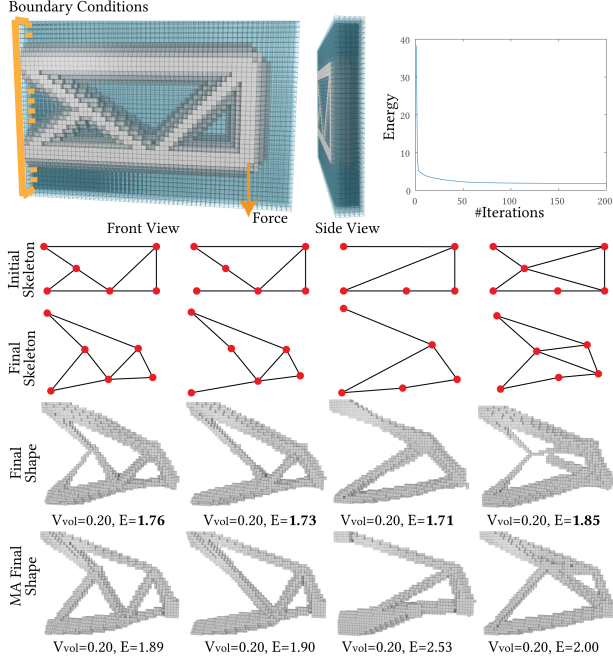


Fig. 19. **Results of shape optimization.** We run shape optimization using our medial skeletal diagram with a similar setup to topology optimization (top left), but with four input skeletons as topology constraints. The optimization converges fast (top right), and each optimized shape shares the same topology as the corresponding input (bottom). Moreover, with our representation (Final Shape), we obtained a lower converged energy in all examples compared with the medial axis representation (MA Final Shape).

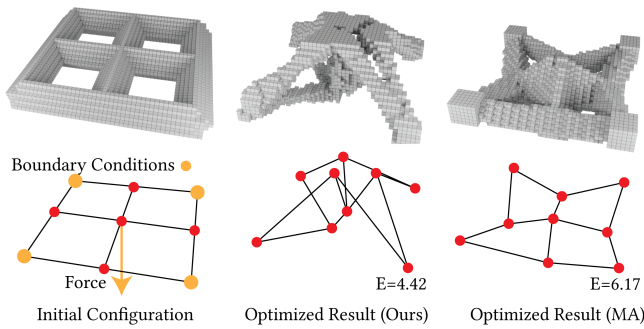


Fig. 20. **The result of shape optimization with a complex topology as constraints.** Our representation is capable of optimizing shapes with user-defined topology constraints, a feature that is non-trivial to incorporate for voxel representation. Moreover, with our representation (middle), we obtained a lower converged energy in this complex example compared with the medial axis representation (right).

Fig. 23 illustrates the comparison on the Knot example. While our method, unlike CoACD, does not ensure the convexity of each primitive, it outperforms in terms of simplicity, demonstrating a reduced number of elements compared to CoACD. The overlapping region in our representation can be fixed by mesh boolean operations.

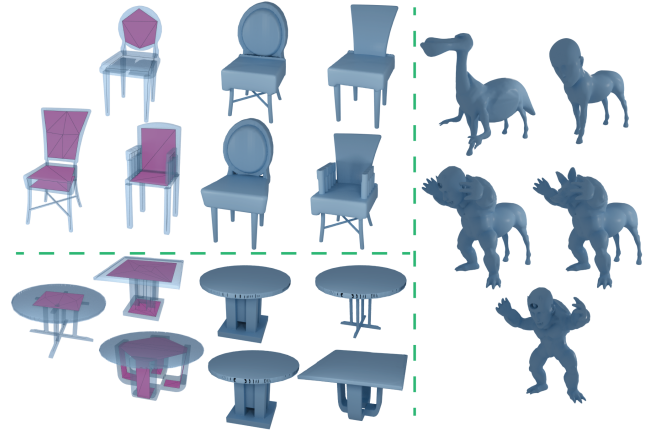


Fig. 21. **Results of shape generation.** We use a cut-and-paste method based on our medial skeletal diagram to generate new shapes. The chairs and tables are generated from shapes provided in Mo et al. [2019], with each row showing outputs generated using three mesh inputs (left). The bottom row shows five generated shapes derived from the 100 shape benchmark.

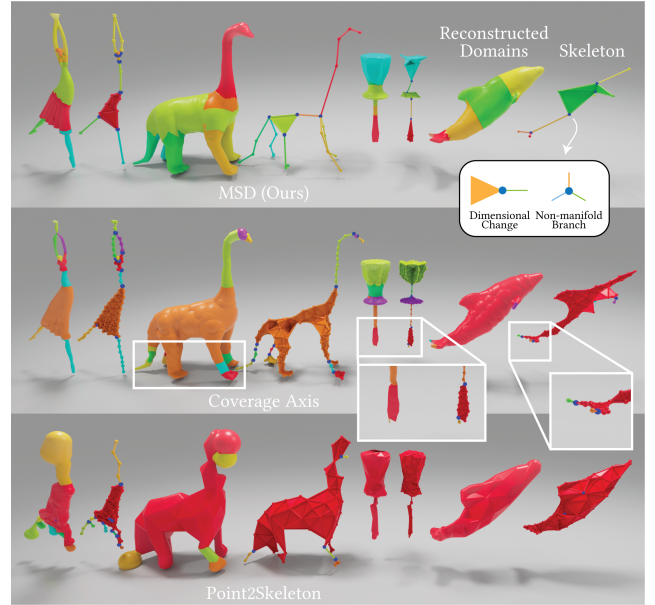


Fig. 22. **Domain Decomposition.** We adopt the method from [Lin et al. 2021] for domain decomposition. Due to the simplicity of our skeleton and its precise shape reconstruction capabilities, our representation yields meaningful domain decomposition results. Although the coverage axis typically provides high-quality decomposition in many instances, it struggles to produce accurate decomposition in certain highlighted areas (white rectangles). While Point2Skeleton excels in generating high-quality decomposition for specific shapes, like chairs, as demonstrated in their paper [Lin et al. 2021], it falls short with more general shapes.

6.2.4 Mesh Alignment. Using our representation, we can ease non-rigid alignment between different shapes. To showcase its efficacy, we select four shapes from different species within the Animal3D dataset [Xu et al. 2023]. We then non-rigidly align the source mesh

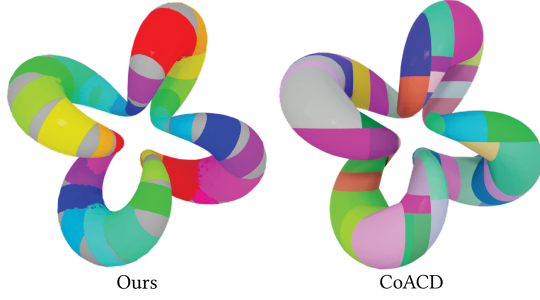


Fig. 23. **Comparison between medial skeletal diagram and CoACD on Knot example.** The number of generalized enveloping primitives in our representation is smaller than the domains decomposed by CoACD.

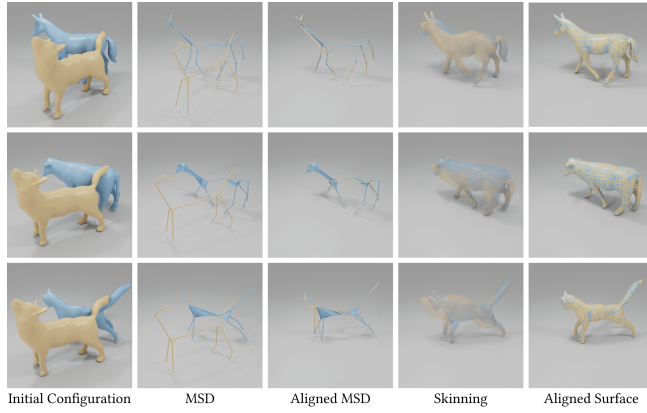


Fig. 24. **Non-rigid Alignment.** Using our representation, one can initially align the medial skeletal diagrams to simplify subsequent surface non-rigid alignment. After the skeletons are matched, the surfaces deform in line with their corresponding skeletons, providing an initial estimate for the final non-rigid alignment. This preliminary alignment considerably eases the non-rigid alignment process for the surfaces, reducing the need for manual intervention.

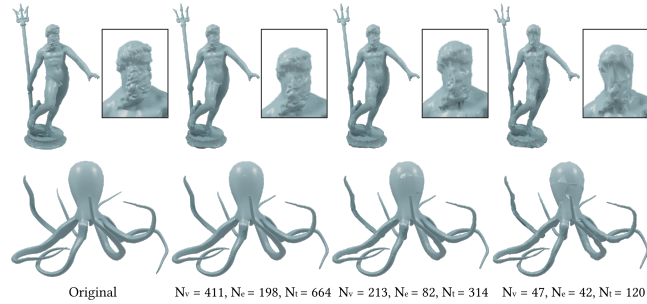


Fig. 25. **Mesh Compression.** Our MSD can be easily used for mesh compression. For each primitive type, we define a canonical mesh with three different resolutions. Primitives at high resolution can be directly downsampled to lower resolutions using the predefined canonical mesh, thereby achieving mesh compression.

(highlighted in yellow) with the other three meshes (depicted in blue), as illustrated in Fig. 24. Instead of directly aligning the source mesh to the target, we first compute our MSD for each shape and

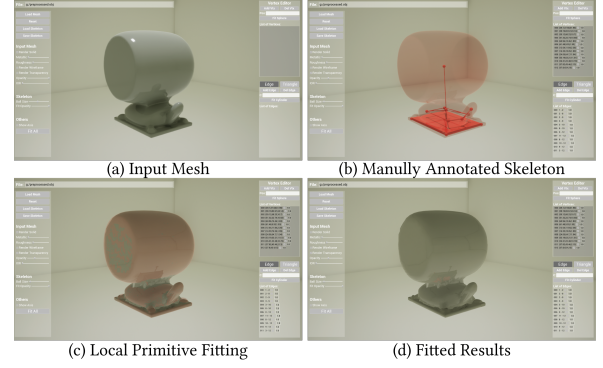


Fig. 26. **Constructing medial skeletal diagram using our GUI.** Users can manually annotate all the skeletal elements and use our proposed local primitive fitting to obtain each primitive.

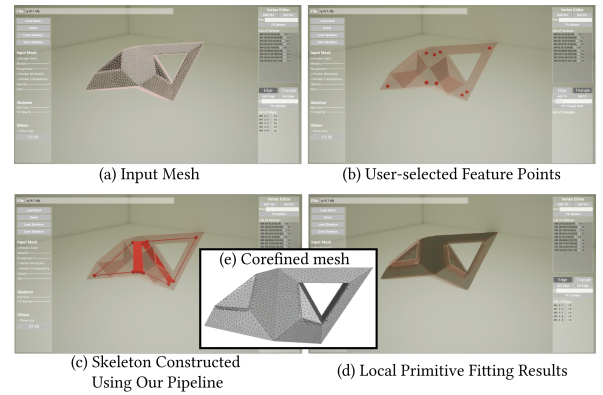


Fig. 27. **Constructing medial skeletal diagram with user-selected feature points using our GUI.** Users can designate only the skeletal vertices where our medial skeletal diagram construction pipeline automatically produces the corresponding skeleton and the reconstructed mesh.

then non-rigidly align their skeletons. Due to the simplicity of these skeletons, alignment is more straightforward than with surface meshes. Once the skeletons are aligned, the surface mesh is deformed according to the skeleton's deformation. In our examples, we treat the skeleton as the kinematic object, with the surface mesh representing the surface of a tetrahedral mesh anchored to the skeleton. This mesh is then deformed using a physically-based simulation. The deformed mesh serves as an initial guess for the non-rigid alignment of the surface meshes, reducing the manual effort needed to select landmark pairs between the source and target meshes.

6.2.5 Mesh Compression. We also propose a simple approach to control the number of continuous parameters associated with each primitive, facilitating mesh compression. For each example shown in Fig. 25, we first compute its MSD. Subsequently, mesh compression is achieved by adjusting the mesh resolution for each primitive. This can be achieved by isotropic remeshing [Botsch et al. 2010] or Garland-Heckbert simplification [Garland and Heckbert 1997]. For each primitive type, we define three canonical meshes with varying resolutions. Specifically, for a sphere mesh, the resolutions are 411, 213, and 47; for a cylinder mesh, they are 198, 82, and 42; and for a

prism mesh, they are 664, 314, and 120. Each calculated generalized enveloping primitive is mapped to its corresponding canonical mesh. As shown in Fig. 25, reducing the number of continuous parameters still preserves the overall original shape.

6.2.6 User Interface. The sparsity of the skeleton in our medial skeletal diagram offers users an intuitive and convenient method for shape modeling. To demonstrate it, we implement a graphical user interface (GUI). Within this GUI, users can create skeletal elements and apply our proposed local primitive fitting to represent a given input shape. As shown in Fig. 26, the user was able to manually construct a skeleton with 13 vertices, 12 edges, and 4 triangles to represent the Drum shape. Moreover, users can manually identify and mark necessary skeletal feature points of the shape, such as topological features, and place points for an input mesh using our GUI. These selected points are then used as inputs for our construction pipeline which automatically produces a high-quality skeleton and the reconstructed mesh. An example of a double torus is illustrated in Fig. 27.

6.3 Additional Results

We also show a gallery of additional results from the benchmark of 100 shapes in Fig. 28 and 29. These results underscore the versatility of our method and its ability to handle a wide variety of shapes.

7 Limitations and Future Work

Topology preservation. Our current approach does not guarantee topological preservation, which is also a limitation of the state-of-the-art work [Wang et al. 2022a,b]. A key reason for this limitation lies in the sparsity of our medial skeletal diagram, which contradicts the sufficient condition for maintaining homotopy, which requires that sampled points be sufficiently dense to satisfy the ϵ -sampling condition, as detailed in [Amenta and Bern 1998; Yan et al. 2009]. This leads to situations where the topology of the reconstructed 3D shape might be compromised. Additionally, while we ensure that each individual primitive is homotopic to its corresponding skeleton element, the final reconstructed shape – which is a union of all the primitives – is not guaranteed to be homotopic to the skeleton. Future research may delve deeper into these topological considerations, investigating potential modifications to the current method that could ensure topological preservation, even with the sparse skeleton of our medial skeletal diagram.

RVD computation. The computational approach for calculating the shortest path during the construction of the medial skeletal diagram could be optimized. While the currently used Dijkstra’s algorithm provides a functional solution, it is not ideal for this particular application as mentioned in [Wang et al. 2020; Xin et al. 2022]. A theoretically more suitable approach would be to employ geodesic Voronoi diagram (GVD). However, this presents a challenge due to the non-manifoldness of the medial axis, the space where GVD is computed. Future work could thus seek methods for replacing RVD with GVD in non-manifold domains. Possible avenues for this could include using a non-manifold Laplacian coupled with heat diffusion [Sharp and Crane 2020] or the development or adaptation of an exact geodesic distance calculator [Surazhsky et al. 2005].

Centrality and uniqueness. Unlike the uniform, linearly interpolated primitives typically used in conventional medial axes, which generally align with the geometric center of a shape, our generalized enveloping primitives are non-convex. As a result, defining centrality for our primitives is challenging. To ensure our approach still captures the essence of centrality, we focus on maintaining local centering for each primitive during optimization. This is achieved through: 1) all vertices of the medial skeletal diagram are constrained to lie on the raw medial axis; 2) the smoothness energy used during local primitive fitting; and 3) $E_{\text{centrality}}$ in global optimization. Despite these efforts, the MSD may still exhibit a loss of “global centrality,” where the overall skeleton structure may not be centered within the global shape. This could result in an increased number of discrete elements in the MSD, as more primitives might be necessary to fully cover the input mesh. Additionally, this could potentially degrade performance in applications such as shape alignment. Future work could explore integrating global information, such as skinning weights, into the MSD construction process. Moreover, while our objective aims to achieve one optimal skeleton, it does not guarantee the uniqueness of the global minimal solution. Future research could focus on incorporating shape properties as constraints to enhance the centrality and uniqueness of the optimized skeleton.

Overlaps of primitives. From the application results, one can find that there are overlapping regions between neighboring primitives. Since the final shape is the union of these primitives, this overlapping does not pose significant problems in applications such as shape optimization and shape generation. However, we acknowledge that this overlap may violate intersection-free constraints in certain tasks, such as mesh decomposition and physical simulation. Additionally, the lack of one-to-one mapping could introduce ambiguity in shape alignment. These issues could be addressed by post-processing methods, including collision handling, Boolean operations, or multi-material meshing [Heiss-Synak et al. 2024], to generate non-overlapping primitives.

Nonlinear skeleton. Our current medial skeletal diagram consists solely of piecewise linear elements. Although this configuration has demonstrated solid performance, the inclusion of higher-order surfaces like polynomial elements [Marschner et al. 2021] could potentially enhance the representation further. Higher-order surfaces could provide more precise coverage of local regions with sparser skeletons, contributing to both the accuracy and efficiency of our representation.

Computational cost. Currently, the time efficiency of computing MSD is less than ideal. Future work that incorporates data-driven methods and graph generation may enhance computational speed, using our current method to supply training data pairs.

8 Conclusion

In this paper, we present the medial skeletal diagram, a skeletal representation that strives for both sparsity in the skeletal structure and completeness in the reconstruction. Our approach significantly reduces the number of discrete elements while preserving completeness by augmenting the complexity of the continuous elements. Our medial skeletal diagram extends the medial axis by introducing generalized enveloping primitives which effectively cover complex local

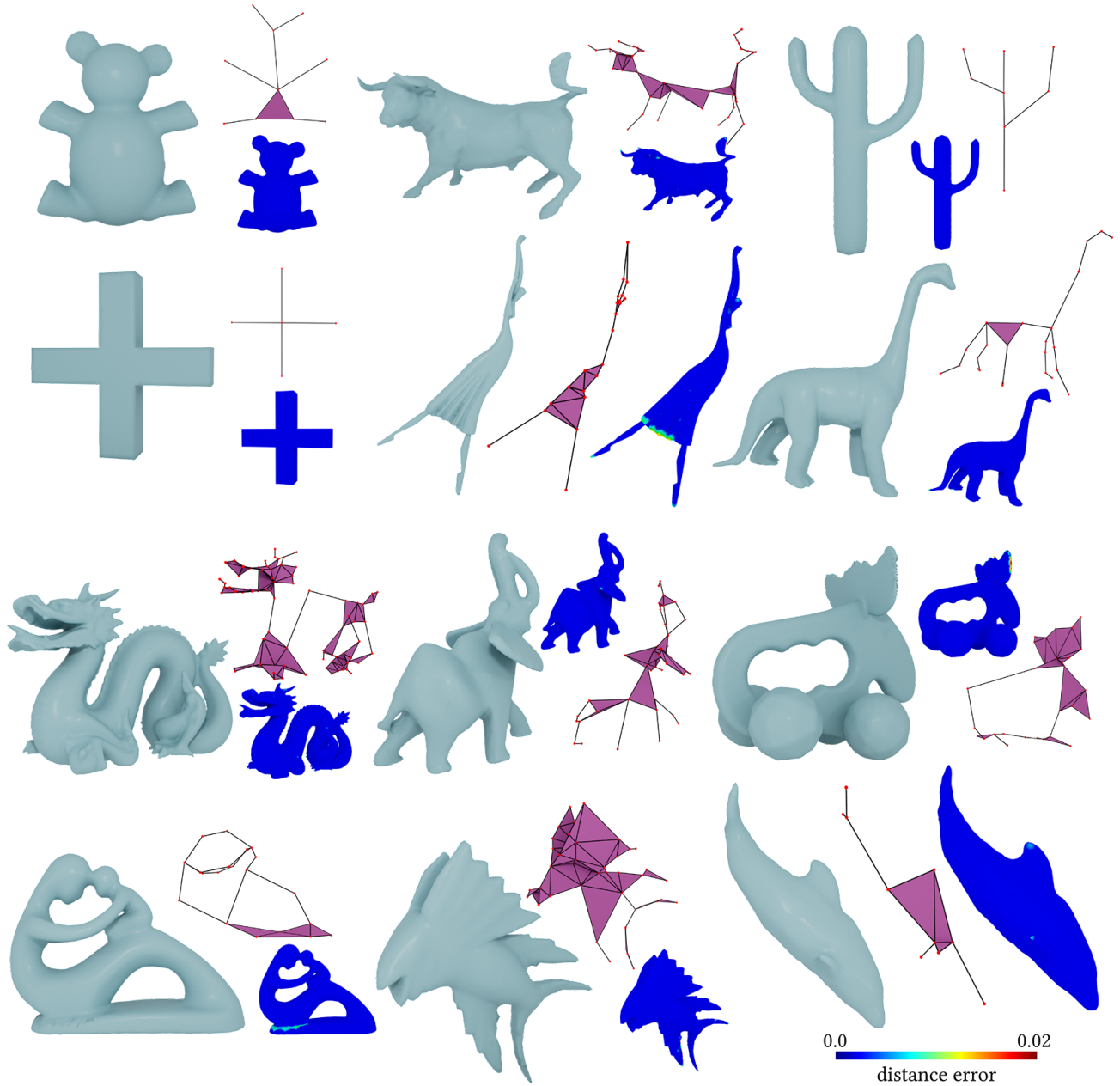


Fig. 28. **A gallery of more results of our medial skeletal diagram (part 1/2).** For each sample, we show the reconstructed shape, the discrete skeleton, and the reconstruction error map on the input mesh.

shape regions. We also present a computational pipeline to construct the medial skeletal diagram from an input shape by employing a local-global optimization paradigm to maximize shape coverage and minimize the count of discrete elements. A post-refinement process is conducted to guarantee an accurate match of our representation with the target mesh. We demonstrated the effectiveness of our method on a benchmark of 100 shapes and highlighted its diverse

applications in shape optimization, shape generation, mesh decomposition, mesh alignment, mesh compression, and user-interactive design.

Acknowledgments

The authors would like to express their gratitude for the constructive feedback provided by the reviewers. The authors also thank the authors of MATFP [Wang et al. 2022a], LS [Bærentzen and Rotenberg

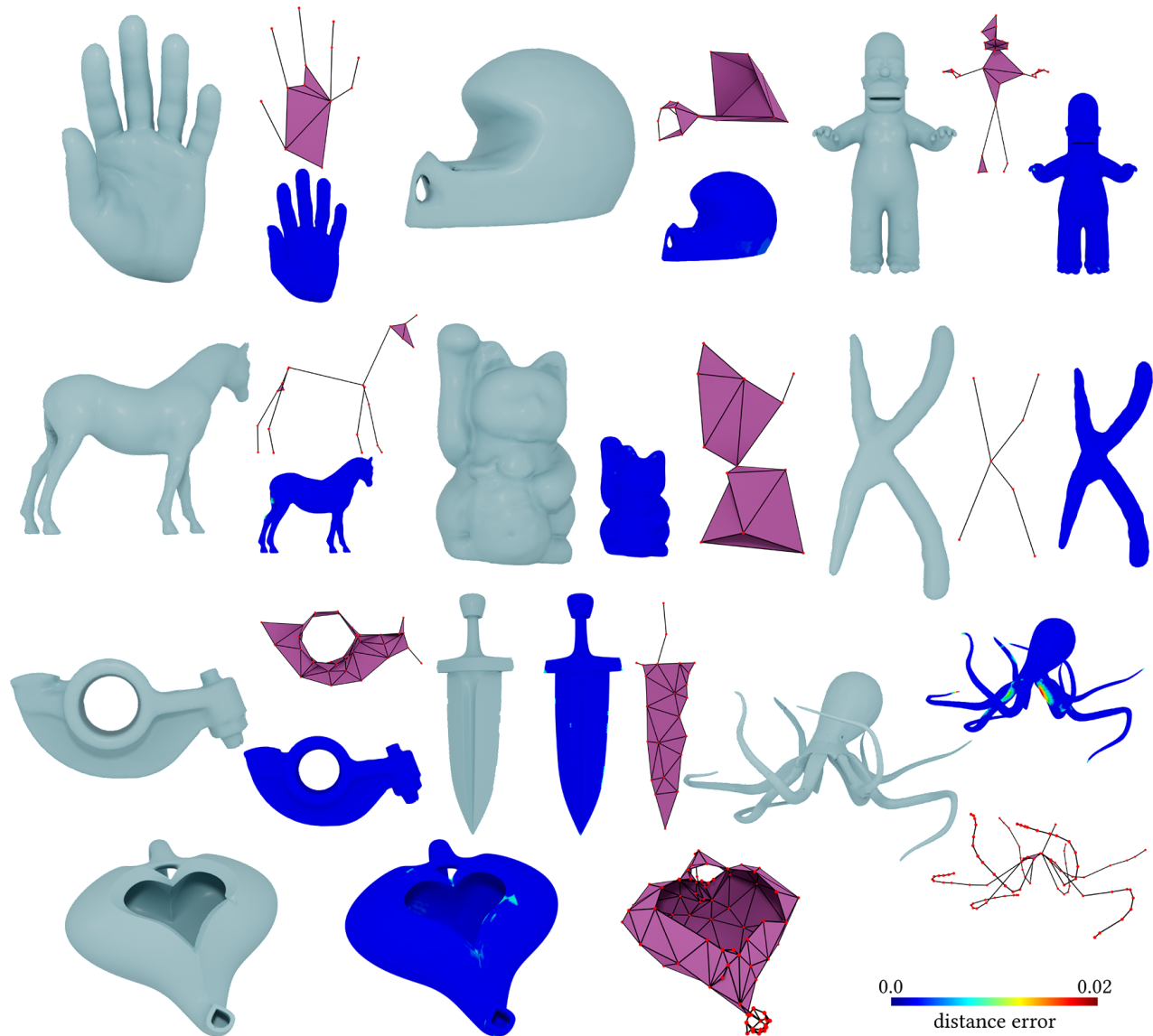


Fig. 29. A gallery of more results of our medial skeletal diagram (part 2/2). For each sample, we show the reconstructed shape, the discrete skeleton, and the reconstruction error map on the input mesh.

2021], Coverage Axis [Dou et al. 2022], Point2Skeleton [Lin et al. 2021], and CoACD [Wei et al. 2022] for making their code publicly available for research purposes.

References

- N. Amenta and M. Bern. 1998. Surface reconstruction by Voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry*. 39–48.
- N. Amenta, S. Choi, and R. K. Kolluri. 2001a. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*. 249–266.
- N. Amenta, S. Choi, and R. K. Kolluri. 2001b. The power crust, unions of balls, and the medial axis transform. *Computational Geometry* 19, 2-3 (2001), 127–153.
- D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. 2009. Stability and computation of medial axes—a state-of-the-art report. *Mathematical foundations of scientific visualization, computer graphics, and massive data exploration* (2009), 109–125.
- O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. 2008. Skeleton extraction by mesh contraction. *ACM Trans. on graphics (TOG)* 27, 3 (2008), 1–10.
- A. Bærentzen and E. Rotenberg. 2021. Skeletonization via local separators. *ACM Trans. on Graphics (TOG)* 40, 5 (2021), 1–18.
- J. A. Bærentzen, R. Abdrashitov, and K. Singh. 2014. Interactive shape modeling using a skeleton-mesh co-representation. *ACM Trans. on Graphics (TOG)* 33, 4 (2014), 1–10.
- J. A. Bærentzen, R. E. Christensen, E. T. Gæde, and E. Rotenberg. 2023. Multilevel Skeletonization Using Local Separators. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.
- I. Baran and J. Popović. 2007. Automatic rigging and animation of 3d characters. *ACM Trans. on graphics (TOG)* 26, 3 (2007), 72–es.
- B. Bell, J. Norato, and D. Tortorelli. 2012. A geometry projection method for continuum-based topology optimization of structures. In *12th AIAA Aviation Technology, integration, and operations (ATIO) conference and 14th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 5485.

- H. Blum. 1967. A transformation for extracting new descriptions of shape. *Models for the perception of speech and visual form* (1967), 362–380.
- P. Borosán, M. Jin, D. DeCarlo, Y. Gingold, and A. Nealen. 2012. Rigmesh: automatic rigging for part-based shape modeling and deformation. *ACM Trans. on Graphics (TOG)* 31, 6 (2012), 1–9.
- M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. 2010. *Polygon mesh processing*. CRC press.
- S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. on graphics (TOG)* 33, 4 (2014), 1–11.
- J. Chaussard, M. Couprie, and H. Talbot. 2011. Robust skeletonization using the discrete λ -medial axis. *Pattern Recognition Letters* 32, 9 (2011), 1384–1394.
- F. Chazal and A. Lieutier. 2005. The “ λ -medial axis”. *Graphical models* 67, 4 (2005), 304–331.
- J. Chen, V. Shapiro, K. Suresh, and I. Tukanov. 2007. Shape optimization with topological changes and parametric control. *International journal for numerical methods in engineering* 71, 3 (2007), 313–346.
- D. Cohen-Steiner and J.-M. Morvan. 2003. Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*. 312–321.
- T. Culver, J. Keyser, and D. Manocha. 2004. Exact computation of the medial axis of a polyhedron. *Computer Aided Geometric Design* 21, 1 (2004), 65–98.
- T. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. 1999. Topology preserving edge contraction. *Publications de l’Institut Mathématique* 66 (1999).
- T. K Dey and W. Zhao. 2002. Approximate medial axis as a voronoi subcomplex. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*. 356–366.
- T. K Dey and W. Zhao. 2004. Approximating the medial axis from the Voronoi diagram with a convergence guarantee. *Algorithmica* 38 (2004), 179–200.
- Z. Dou, C. Lin, R. Xu, L. Yang, S. Xin, T. Komura, and W. Wang. 2022. Coverage Axis: Inner Point Selection for 3D Shape Skeletonization. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 419–432.
- H. Edelsbrunner and N. R Shah. 1994. Triangulating topological spaces. In *Proceedings of the tenth annual symposium on Computational geometry*. 285–292.
- R. Fang, A. Godil, X. Li, and A. Wagan. 2008. A new shape benchmark for 3D object retrieval. In *Advances in Visual Computing: 4th International Symposium, ISVC 2008, Las Vegas, NV, USA, December 1-3, 2008. Proceedings, Part I* 4. Springer, 381–392.
- M. Foskey, M. C Lin, and D. Manocha. 2003. Efficient computation of a simplified medial axis. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*. 96–107.
- M. Garland and P. S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 209–216.
- P. Heiss-Synak, A. Kalinov, M. Strugaru, A. Etemadi, H. Yang, and C. Wojtan. 2024. Multi-Material Mesh-Based Surface Tracking with Implicit Topology Changes. *ACM Trans. Graph.* 43, 4, Article 171 (sep 2024), 14 pages.
- H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen. 2013. L1-medial skeleton of point cloud. *ACM Trans. Graph.* 32, 4 (2013), 65–1.
- A. Jacobson, D. Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. <https://libigl.github.io/>.
- P. Kumar. 2023. HoneyTop90: A 90-line MATLAB code for topology optimization using honeycomb tessellation. *Optimization and Engineering* 24, 2 (2023), 1433–1460.
- J. M Lee. 2012. *Smooth manifolds*. Springer.
- P. Li, B. Wang, F. Sun, X. Guo, C. Zhang, and W. Wang. 2015. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Trans. on Graphics (TOG)* 35, 1 (2015), 1–16.
- Y. Liang, F. He, and X. Zeng. 2020. 3D mesh simplification with feature preservation based on whale optimization algorithm and differential evolution. *Integrated Computer-Aided Engineering* 27, 4 (2020), 417–435.
- A. Lieutier. 2003. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type than its medial axis. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*. 65–75.
- C. Lin, C. Li, Y. Liu, N. Chen, Y.-K. Choi, and W. Wang. 2021. Point2skeleton: Learning skeletal representations from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4277–4286.
- C. Lin, L. Liu, C. Li, L. Kobbelt, B. Wang, S. Xin, and W. Wang. 2020. Seg-mat: 3d shape segmentation using medial axis transform. *IEEE transactions on visualization and computer graphics* 28, 6 (2020), 2430–2444.
- M. Livesu, F. Guggeri, and R. Scateni. 2012. Reconstructing the curve-skeletons of 3D shapes using the visual hull. *IEEE transactions on visualization and computer graphics* 18, 11 (2012), 1891–1901.
- S. Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.
- W.-C. Ma, F.-C. Wu, and M. Ouhyoung. 2003. Skeleton extraction of 3D objects with radial basis functions. In *2003 Shape Modeling International*. IEEE, 207–215.
- K. Mamou, E. Lengyel, and A. Peters. 2016. Volumetric hierarchical approximate convex decomposition. *Game engine gems* 3 (2016), 141–158.
- R. Marie, O. Labbani-Igbida, and E. M. Mouaddib. 2016. The delta medial axis: a fast and robust algorithm for filtered skeleton extraction. *Pattern Recognition* 56 (2016), 26–39.
- Z. Marschner, P. Zhang, D. Palmer, and J. Solomon. 2021. Sum-of-squares geometry processing. *ACM Trans. on Graphics (TOG)* 40, 6 (2021), 1–13.
- B. Miklos, J. Giesen, and M. Pauly. 2010. Discrete scale axis representations for 3D geometry. In *ACM SIGGRAPH 2010 papers*. 1–10.
- V. Milenkovic. 1993. Robust Construction of the Voronoi Diagram of a Polyhedron.. In *CCCG*, Vol. 93. Citeseer, 473–478.
- K. Mo, S. Zhu, A. X Chang, L. Yi, S. Tripathi, L. J Guibas, and H. Su. 2019. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 909–918.
- J. R Munkres. 2018. *Analysis on manifolds*. CRC Press.
- K. Pandey, J. A. Bærentzen, and K. Singh. 2022. Face Extrusion Quad Meshes. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.
- S. M Pizer, K. Siddiqi, G. Székely, J. N Damon, and S. W Zucker. 2003. Multiscale medial loci and their properties. *International Journal of Computer Vision* 55 (2003).
- D. Rebain, B. Angles, J. Valentin, N. Vining, J. Peethambaran, S. Izadi, and A. Tagliasacchi. 2019. LSMAT least squares medial axis transform. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 5–18.
- D. Rebain, K. Li, V. Sitzmann, S. Yazdani, K. M. Yi, and A. Tagliasacchi. 2021. Deep medial fields. *arXiv preprint arXiv:2106.03804* (2021).
- J. A Richardson and J. L Kuester. 1973. Algorithm 454: the complex method for constrained optimization [E4]. *Commun. ACM* 16, 8 (1973), 487–489.
- P. K Saha, G. Borgefors, and G. S. di Baja. 2016. A survey on skeletonization algorithms and their applications. *Pattern recognition letters* 76 (2016), 3–12.
- A. Sharf, T. Lewiner, A. Shamir, and L. Kobbelt. 2007. On-the-fly curve-skeleton computation for 3d shapes. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 323–328.
- N. Sharp and K. Crane. 2020. A laplacian for nonmanifold triangle meshes. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 69–80.
- W. Shen, X. Bai, R. Hu, H. Wang, and L. J. Latecki. 2011. Skeleton growing and pruning with bending potential ratio. *Pattern Recognition* 44, 2 (2011), 196–209.
- E. C. Sherbrooke, N. M. Patrikalakis, and E. Brisson. 1996. An algorithm for the medial axis transform of 3D polyhedral solids. *IEEE Transactions on Visualization and Computer Graphics* 2, 1 (1996), 44–61.
- P. D. Simari and K. Singh. 2005. Extraction and remeshing of ellipsoidal representations from mesh data. In *Graphics Interface*. 161–168.
- A. Sobiecki, A. Jalba, and A. Telea. 2014. Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recognition Letters* 47 (2014), 147–156.
- A. Sud, M. Foskey, and D. Manocha. 2005. Homotopy-preserving medial axis simplification. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*. 39–50.
- F. Sun, Y.-K. Choi, Y. Yu, and W. Wang. 2015. Medial meshes—a compact and accurate representation of medial axis transform. *IEEE transactions on visualization and computer graphics* 22, 3 (2015), 1278–1290.
- V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. on Graphics (TOG)* 24, 3 (2005), 553–560.
- A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang. 2012. Mean curvature skeletons. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1735–1744.
- A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 2016. 3D skeletons: A state-of-the-art report. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 573–597.
- J. Tang, X. Han, J. Pan, K. Jia, and X. Tong. 2019. A skeleton-bridged deep learning approach for generating meshes of complex topologies from single rgb images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4541–4550.
- J. Tierny, J.-P. Vandebror, and M. Daoudi. 2008. Enhancing 3D mesh topological skeletons with discrete contour constrictions. *The Visual Computer* 24 (2008), 155–172.
- N. Wang, B. Wang, W. Wang, and X. Guo. 2022a. Computing Medial Axis Transform with Feature Preservation via Restricted Power Diagram. *ACM Trans. on Graphics (TOG)* 41, 6 (2022), 1–18.
- P. Wang, Z. Wang, S. Xin, X. Gao, W. Wang, and C. Tu. 2022b. Restricted Delaunay Triangulation for Explicit Surface Reconstruction. *ACM Trans. on Graphics (TOG)* 41, 5 (2022), 1–20.
- P. Wang, S. Xin, C. Tu, D. Yan, Y. Zhou, and C. Zhang. 2020. Robustly computing restricted Voronoi diagrams (RVD) on thin-plate models. *Computer Aided Geometric Design* 79 (2020), 101848.
- X. Wei, M. Liu, Z. Ling, and H. Su. 2022. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Trans. on Graphics (TOG)* 41, 4 (2022), 1–18.

- S.-K. Wong, J.-A. Yang, T.-C. Ho, Y.-C. Chen, and J.-H. Chuang. 2012. A skeleton-based approach for shape correspondence. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*. 329–336.
- S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or. 2015. Deep points consolidation. *ACM Trans. on Graphics (TOG)* 34, 6 (2015), 1–13.
- S. Xin, P. Wang, R. Xu, D. Yan, S. Chen, W. Wang, C. Zhang, and C. Tu. 2022. SurfaceVoronoi: Efficiently Computing Voronoi Diagrams Over Mesh Surfaces with Arbitrary Distance Solvers. *ACM Trans. on Graphics (TOG)* 41, 6 (2022), 1–12.
- J. Xu, Y. Zhang, J. Peng, W. Ma, A. Jesslen, P. Ji, Q. Hu, J. Zhang, Q. Liu, J. Wang, et al. 2023. Animal3D: A Comprehensive Dataset of 3D Animal Pose and Shape. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9099–9109.
- D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 1445–1454.
- Y. Yan, D. Letscher, and T. Ju. 2018. Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes. *ACM Trans. on Graphics (TOG)* 37, 4 (2018), 1–13.
- Y. Yan, K. Sykes, E. Chambers, D. Letscher, and T. Ju. 2016. Erosion thickness on medial axes of 3D shapes. *ACM Trans. on Graphics (TOG)* 35, 4 (2016), 1–12.
- B. Yang, J. Yao, B. Wang, J. Hu, Y. Pan, T. Pan, W. Wang, and X. Guo. 2020. P2MAT-NET: Learning medial axis transform from sparse point clouds. *Computer Aided Geometric Design* 80 (2020), 101874.
- Y. Zhou, K. Yin, H. Huang, H. Zhang, M. Gong, and D. Cohen-Or. 2015. Generalized cylinder decomposition. *ACM Trans. on Graphics* 34, 6 (2015), 171–1.

Appendix

A Shape Optimization Using Gradient Descent

We make the Heaviside step function differentiable by modifying $H(\cdot)$ as

$$H(x) = \begin{cases} 0 & x < \gamma, \\ -\frac{1}{4\gamma^3}x^3 + \frac{3}{4\gamma}x + 0.5 & -\gamma \leq x \leq \gamma, \\ 1 & x > \gamma \end{cases}, \quad (9)$$

where γ is a small number. We choose $\gamma = 0.01$ as the cell size of each voxel. The function is C^1 continuous in $\mathbb{R}$. Fig. 30 shows the plot of the function $H(\cdot)$.

To perform gradient descent, we first convert the inequality constraint into a quadratic energy term,

$$E_v(\hat{\mathbf{v}}, \hat{\mathbf{r}}) = \begin{cases} \frac{w}{2} \left(\sum_{x_i} \frac{\theta(x_i; \hat{\mathbf{v}}, \hat{\mathbf{r}})}{|\Omega|} - V_{\text{vol}} \right)^2 & \text{if } \sum_{x_i} \frac{\theta(x_i; \hat{\mathbf{v}}, \hat{\mathbf{r}})}{|\Omega|} > V_{\text{vol}}, \\ 0 & \text{o.w.} \end{cases} \quad (10)$$

The energy is non-zero only if the current volume exceeds the user-defined one. Next, we differentiate the static equilibrium constraints

$$\frac{\partial \mathbf{K}(\theta)}{\partial \theta} \mathbf{u} + \mathbf{K}(\theta) \frac{\partial \mathbf{u}}{\partial \theta} = 0. \quad (11)$$

The gradient of the objective w.r.t. $(\hat{\mathbf{v}}, \hat{\mathbf{r}})$ is

$$\frac{dE}{d(\hat{\mathbf{v}}, \hat{\mathbf{r}})} = \frac{\partial E}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial (\hat{\mathbf{v}}, \hat{\mathbf{r}})} + \frac{\partial E}{\partial (\hat{\mathbf{v}}, \hat{\mathbf{r}})} \quad (12)$$

$$= \left(-\frac{1}{2} \mathbf{u}^T \frac{\partial \mathbf{K}(\theta)}{\partial \theta} \mathbf{u} + w S^T \left(S \frac{\theta}{|\Omega|} - V_{\text{vol}} \right) \right) \frac{\partial \theta}{\partial (\hat{\mathbf{v}}, \hat{\mathbf{r}})}, \quad (13)$$

where S is the summation linear operator. We apply gradient descent to optimize the objective of shape optimization. At each gradient descent iteration, we also perform a back-tracing line search to guarantee a decrease in the total energy. At each line search iteration, we first compute $\mathbf{K}$ from a new $(\hat{\mathbf{v}}, \hat{\mathbf{r}})$ and then solve the static equilibrium to obtain $\mathbf{u}$. The energy is computed as $\frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} + E_v$.

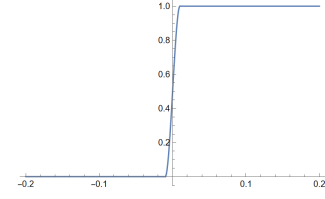


Fig. 30. The modified function $H(\cdot)$

B Well-definedness of Directional Vectors in Generalized Enveloping Primitive

Definition B.1. [Munkres 2018, ϵ -neighborhood] Let P be a smooth submanifold in $\mathbb{R}^3$ and $\epsilon : P \mapsto \mathbb{R}^+$ be a smooth positive function. The ϵ -neighborhood of P , denoted by P^ϵ , is defined as,

$$P^\epsilon := \{y \in \mathbb{R}^3 \mid \|y - x\|_2 < \epsilon(x) \text{ for some } x \in P\}. \quad (14)$$

Furthermore, the following theorem asserts the existence of a surjective map between P^ϵ and P :

THEOREM B.2. [Lee 2012, Proposition of Tubular Neighborhood Theorem] For a smooth submanifold P in $\mathbb{R}^3$, there exists a smooth positive function $\epsilon : P \mapsto \mathbb{R}^+$ such that the map $\pi_\epsilon : P^\epsilon \mapsto P$, which maps each point $y \in P^\epsilon$ to the closest point $\pi_\epsilon(y)$ in P , is a submersion. Moreover, if P is compact, then ϵ can be taken to be a constant.

Note that π_ϵ is well-defined due to the fact that P is convex, so for each $y \in P^\epsilon$, there is a unique closest point in P . This theorem ensures that for a medial mesh element P , there exists an appropriate constant ϵ to construct a smooth closed submanifold P^ϵ and establish a surjective map π_ϵ between P^ϵ and P . For more details of choosing ϵ , we refer the readers to the proof of the theorem in [Lee 2012]. If we consider the unit normal vector for each point y on the boundary ∂P^ϵ of P^ϵ , we obtain a submersion $g : N\partial P^\epsilon \mapsto P$,

$$g = \pi_\epsilon \circ f^{-1},$$

where $N\partial P^\epsilon = \{(y, n) \mid y \in \partial P^\epsilon, n \in N_p \partial P^\epsilon\}$ denotes the unit normal bundle of ∂P^ϵ , and $f : \partial P^\epsilon \mapsto N\partial P^\epsilon$, which is constructed from Gauss map of ∂P^ϵ , maps between ∂P^ϵ and $N\partial P^\epsilon$. Here f is bijective since ∂P^ϵ is convex. The map g serves as a crucial connector between the medial mesh element P and the normal vectors of its ϵ -neighborhood boundary ∂P^ϵ . Essentially, g associates each point in P with a set of unit directional vectors – the normal vectors of ∂P^ϵ , while the surjection of π_ϵ concurrently guarantees each such normal vector corresponds to a specific point in P . This provides a theoretical guarantee that our primitives are mathematically well-defined in the continuous setting, i.e., every point in P correlates with at least one directional vector through the surjection of g .

Due to the convexity of P^ϵ , the directional vectors $d(P)$ are not affected by the choice of ϵ in the continuous case. In our implementation for discrete primitives, as shown in Fig. 9, we choose $\epsilon = 0.01$ for input shapes normalized to the range $[0, 1]$.