Curve Skeletonization in Continuous domain for Meshes and Point Clouds

Jai Bardhan
TCS Research
jai.bardhan@cvut.cz

Ramya Hebbalaguppe
TCS Research
ramya.hebbalaguppe@tcs.com

Aravind Udupa
IIT Delhi
ara.udupa@gmail.com

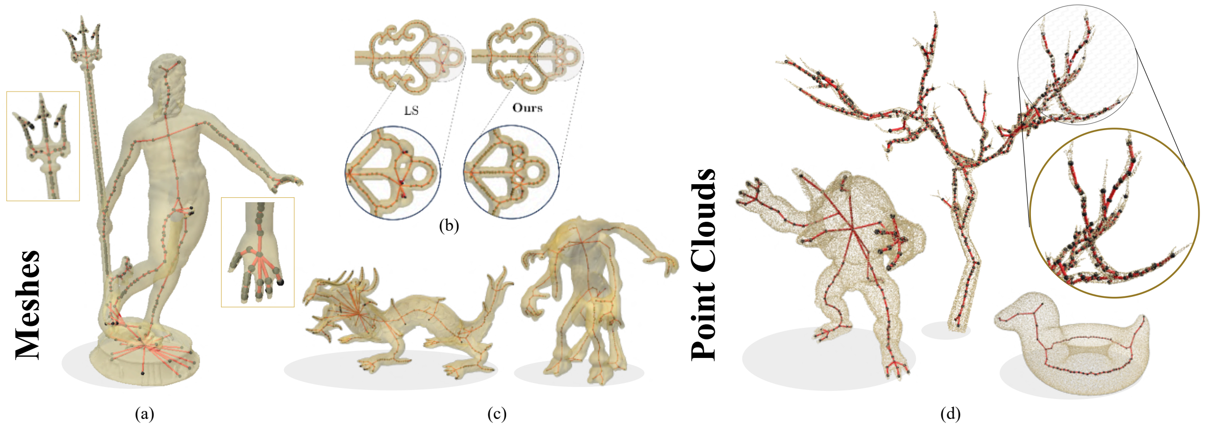


Figure 1. **Representative results of the proposed `CSCD` on diverse 3D shapes from various benchmark datasets - Meshes (left) and Point Clouds (right):** (a) We show the result of our method on the `neptune` mesh – The inset illustrates the excellent skeletal quality for both the hand and the trident; (b) We show a comparison of `CSCD` to a contemporary method (LS [2]) for `Copper-key` – `CSCD` reconstructs the holes of the shape better; (c) `CSCD` captures fine details in complex meshes like `xyzrgb-dragon` (Stanford Library) and `TID:133568` (Thingi10k); (d) Our framework generalizes across domains, performing well on point clouds such as `bob`, `armadillo`, and the intricate `dead tree`. See Appendix H for more results.

Abstract

Advancements in 3D curve skeletonization are accelerating progress across a wide range of applications. However, developing robust skeletonization algorithms that capture intricate object details remains challenging. Skeletonization via Local Separators (LS) offers an efficient graph-based approach but suffers from representation inaccuracies due to its discrete nature. To address this, we introduce CSCD, a novel framework for Curve Skeletonization in the Continuous Domain, generalizing LS to manifolds. Specifically, we present two realizations: CSCD-M for meshes and CSCD-PC for point clouds. CSCD-M leverages the intrinsic triangulation of a mesh for resilience to noise and improved topological preservation, while CSCD-PC employs tufted Laplacians for enhanced robustness. To our knowledge, CSCD-M is the first intrinsic method for curve skeletonization. Our results show CSCD-M matches LS performance across diverse meshes and outperforms LS (TOG'21) on benchmarks like Thingi10k dataset. CSCD-PC qualitatively outperforms CoverageAxis++ (Eurographics'24) and EPCS (CAG'23). Finally, we demonstrate the efficacy of CSCD in a few downstream tasks: object classification, shape segmentation, identifying handles, tunnels, and constrictions in objects.

Project Website: `https://cscd-skel.pages.dev`

1. Introduction

3D object representation is a fundamental problem in computer graphics/ vision, as it aims to capture the shape, structure, and appearance of objects in a digital format. Various representations have been developed over the years, each with strengths and limitations. Meshes [48, 51], point clouds, distance fields [20], and recently NeRFs [15, 32] are commonly used for geometry processing. These representations, however, can be detailed or complex for certain applications, especially in shape and motion modeling.

Curve skeletons have emerged as a powerful alternative. They capture the topology and geometry of the object through a set of 1-D connected curves that lie in the medial axis of the object and approximate key geometrical features. Curve skeletons are invaluable for a multitude of applications including shape segmentation [36], matching [17], retrieval [43], animation [29], reconstruction [13].

Challenges in curve skeletonization: Despite their utility, the computation of curve skeletons is fraught with challenges, necessitating algorithms that are both robust and sensitive to nuances such as capturing fine-grained shapes and structure [36, 47]. There is a lack of a clear definition of curve skeletons for 3D objects. This has led to a multitude of hand-crafted methods, each with its strengths and weaknesses. Most of these methods rely on the idea that for tubular shapes, there exists a 1D structure that preserves the shape of the topology. Among them: (1) geometric features-based methods rely on identifying key geometric features of the shape, but struggle with complex shapes or noisy data; (2) local decimation methods progressively simplify the objects while maintaining the topological structure, but generally fail to capture the high fidelity features; (3) division based methods divide the shape into regions, and compute skeleton points for these regions; (4) learning based methods like [25, 57] use machine learning techniques to generate skeletons, but usually fail to generalize to unseen objects, and (5) Medial Axis Transform based methods identify the medial axis/plane and prune to obtain the curve skeletons. Representative works include: [1, 5, 6, 8, 11, 12, 23, 24, 27, 33, 35, 46, 52, 53]. Recently, a skeletonization technique based on local separators (LS) [2] has shown particular promise to produce curve skeletons with higher fidelity, capturing the finer details of the shape, where methods such as MCF [46] and L_1-medial skeletonization [18] sometimes fall short. The LS method works by constructing local separators on graphs to divide the graph locally into non-overlapping regions, then calculating the centroid for each region to form the nodes of the skeleton. We summarize the native domain (representation) of operation for a few methods in Tab. 1. We want to take note that while point cloud/graph based methods may be applied to meshes, they are not natively developed for meshes and therefore miss out face-level information available.

Although the LS method achieves strong results, it is limited by its discrete representation, which can produce noisy curve skeletons and sensitivity to input quality (e.g., poorly triangulated meshes or noisy point clouds). Moreover, the absence of a continuous formulation hinders its applicability to continuous manifolds and restricts integration with representation-specific algorithms (Sec. 2.4).

Motivated by high-quality results in [2] and the need to address the limitations, we introduce CSCD, a novel framework for **C**urve **S**keletonization in the **C**ontinuous **D**omain, generalizing LS to manifolds (see Fig. 2).

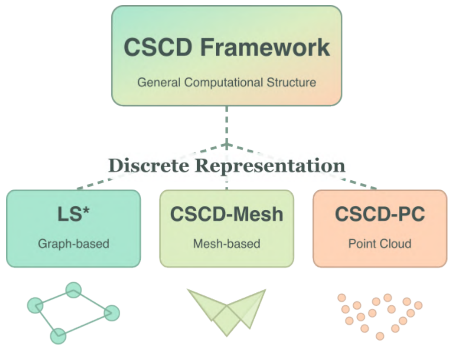


Figure 2. **CSCD Framework Overview:** Our framework generalizes LS [2], such that a graph-based realization results in an algorithm similar to LS (Appendix K.1), mesh-based realization leads to CSCD-M, and a point cloud realization leads to CSCD-PC (Sec. 3).

Method	Graph	Mesh	Point Cloud
LS [2]	✓	✗	✗
MSLS [3]	✓	✗	✗
ROSA [45]	✗	✗	✓
MCF [46]	✗	✓	✗
EPCS [23]	✗	✗	✓
CA++ [12, 53]	✗	✓	✓
CSCD (Ours)	✓ (App. K.1)	✓CSCD-M	✓CSCD-PC

Table 1. Native representations for various curve skeletonization algorithms. CSCD enables construction of skeletonization algorithms tailored to each representation.

Rationale for CSCD: CSCD operates on manifolds rather than discrete structures like graphs, offering advantages for 3D shape analysis, particularly of surface and geometric features [21, 44, 49]. (1) As continuous representations, manifolds more accurately capture intrinsic geometry and topology (e.g., curvature, geodesic distances), whereas graph discretization can distort these properties. (2) CSCD leverages differential geometry tools such as the Laplace-Beltrami operator and intrinsic vector fields, which are robust to non-rigid deformations. By using domain specific implementations, we can reduce discretization artefacts that are present in graph-based methods approximations.

Our **key contributions** include:

(1) We introduce the CSCD framework, a generalization of the LS algorithm beyond graph representations (see, Fig. 2, Tab. 1).

(2) We introduce CSCD-M, a realization of CSCD for meshes that operates upon the intrinsic triangulation of mesh. This is the first method to operate on intrinsic triangulation offering robustness by construction. CSCD-M performs comparable or better than LS and is $\sim 60\%$ faster, on average, on our set of meshes of varying sizes and complexity (see Tab. 6).

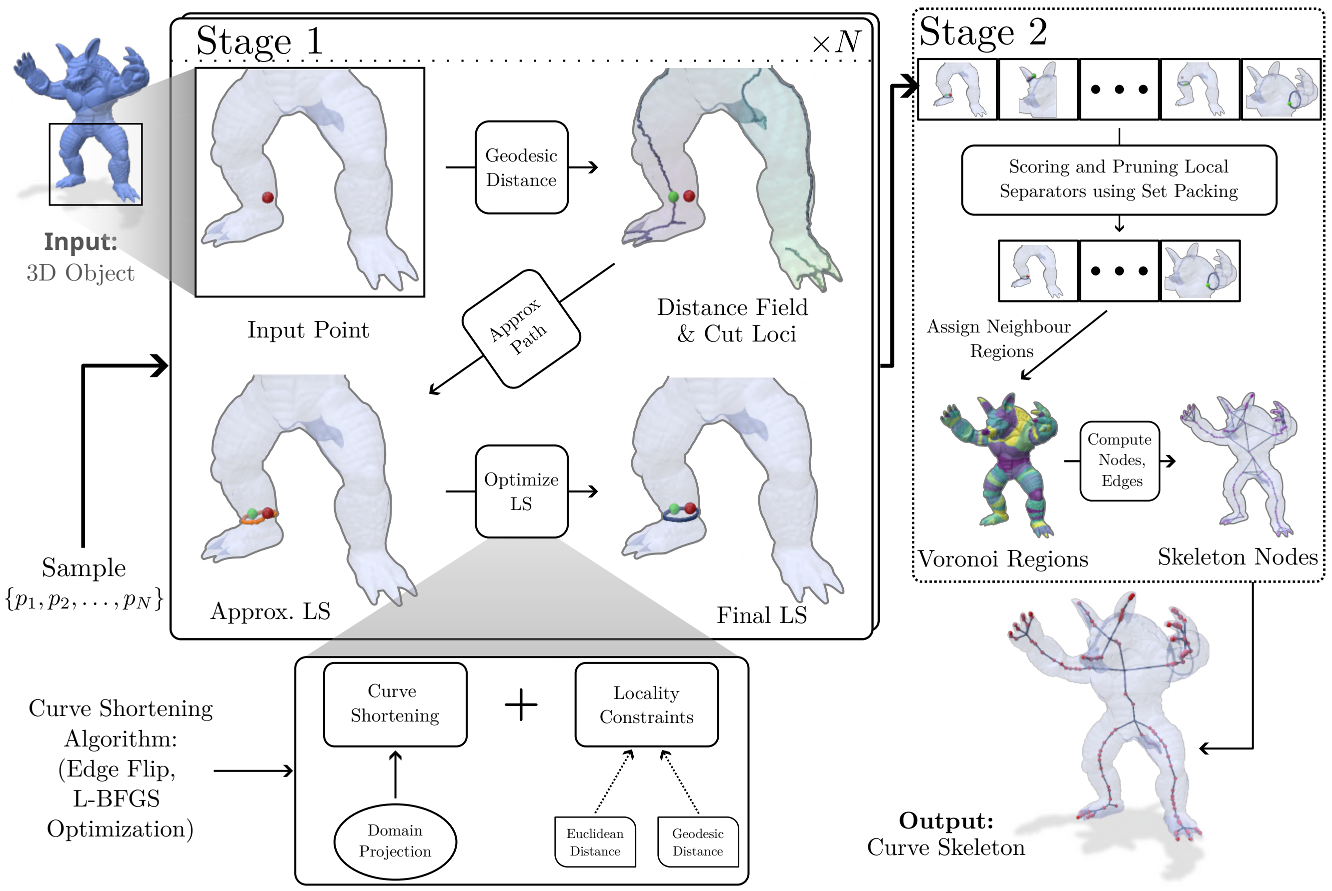


Figure 3. **[Schematic of CSCD for Curve Skeletonization].** The entire Curve Skeletonization framework can be divided into two stages. In Stage 1, we calculate the various local separators given the 3D object as input(input can be a mesh or a point cloud). Once we have a sampled point, we calculate the geodesic distance to all other points and find the cut loci of the point to identify the target cut locus. Then an approximate local separator (LS) is constructed, followed by the LS optimization. The optimization is specific to the particular representation and is in the presence of a locality constraint. In Stage 2, we calculate the skeleton from the set of local separators. First, we prune and pack the previously obtained separators. Based on the obtained separators, we divide the object into Voronoi regions. These regions correspond to nodes, and neighbouring regions are connected to give the skeleton. Post-processing is finally performed to convert cliques to stars in the resultant skeleton.

(3) Capitalizing on the generality of CSCD, we also introduce CSCD-PC, a realization of CSCD to point clouds.

(4) As a crucial component of our framework, we adapt and improve upon existing cut locus identification strategies. For CSCD-M, we develop an intrinsic formulation of the algorithm, leading to improved robustness on poorly triangulated meshes. For CSCD-PC, we introduce a novel cut locus identification strategy tailored for point clouds (see. Fig. 7a, Fig. 7b and Appendix. G).

(5) Finally, we demonstrate that our framework can be minimally modified to approximately identify handles, tunnels, and constricting loops, thereby extending its applicability beyond skeletonization (see App. Fig. 22). Our improved skeletonization yields better results for downstream applications, (App. Fig. 21, Tab. 4). See, App. I.

2. CSCD

2.1. CSCD Framework

CSCD is a framework for local separators based skeletonization on manifolds. Our method takes a shape $X \in \mathbb{R}^3$ as input, where X can be any 3D representation with discrete differential operators (Eg., gradient and Laplace-Beltrami). Our goal is to generate a curve skeleton C from the input.

Our framework comprises two stages: **(1)** Finding a set of local separators that divide the manifold locally into two halves; **(2)** scoring and selecting an optimal set of non-overlapping separators through set-packing. We then assign the nearest region of the shape to each local separator, calculate centroids to obtain skeleton nodes, and connect nodes of neighboring regions. Finally, we remove cliques to form the curve skeleton. Refer to Algorithm 1 for further details,

Algorithm 1 CSCD Framework (Detailed alg. in Supplementary B)

```
Input: A 3D object $\mathcal{O}$
Output: The curve skeleton of $\mathcal{O}$ as $\mathcal{C}$
 1: $\mathcal{P} \leftarrow$ set of points on surface $\mathcal{S}$
 2: for $p \in \mathcal{P}$ do
 3:     $D \leftarrow$ geodesic distances $f(p, \mathcal{S})$
 4:     $C \leftarrow$ cut-locus mask $f(D, \mathcal{S})$
 5:     $t \leftarrow$ target cut locus from $(C, D, p, \mathcal{S})$
 6:     $\hat{l} \leftarrow$ traced path from $t$ to $p$
 7:     $l \leftarrow$ locally optimized separator from $\hat{l}$
 8: end for
 9: $L \leftarrow \{l_1, \ldots, l_{|\mathcal{P}|}\}$
10: $M \leftarrow$ overlap map where $M_{i,j} = 1$ if loops $i, j$ overlap
11: $\mathcal{L} \leftarrow$ separators after packing $(M, L)$
12: $\mathcal{R} \leftarrow$ nearest region assignments $f(\mathcal{L}, \mathcal{S})$
13: $\hat{N} \leftarrow$ node positions $f(\mathcal{R}, \mathcal{S})$
14: $\hat{E} \leftarrow$ edge connectivity $f(\mathcal{R}, \mathcal{S})$
15: $(N, E) \leftarrow$ clique-cleaned graph from $(\hat{N}, \hat{E})$
16: $\mathcal{C} \leftarrow (N, E)$
```

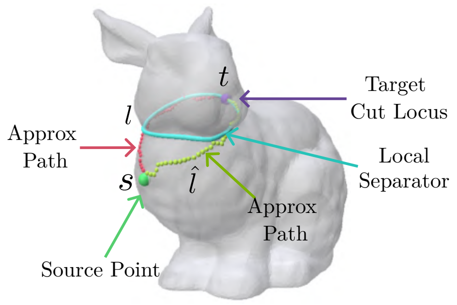


Figure 4. **[Illustration of a local separator.]** The green vertex is the source s, the purple vertex is the target cut locus t (Sec. 3.1.2). The two paths $\hat{l}$ (red and lime) are the approximate paths (Sec. 3.1.4), and the cyan curve l is the final local separator (Sec. 3.1.5). See Sec. M for Terminology/definitions.

where, $f(\cdot)$ denotes functions specific to each step.

2.2. Stage 1: Local Separator Construction

An ideal local separator is a locally short path that divides the surface into two halves with the following properties:

P1: The separator goes around geometrical features (protrusions) rather than simply dividing the local surface.

P2: The separator is *locally shortest* within a defined locality.

To construct a local separator, we start with a source point (Fig. 4) and calculate geodesic distances. Geodesics are curves on the manifold that locally minimize distance, i.e., generalizing straight lines to curved spaces. To satisfy property **P1**, the separator passes through a cut locus of the source. **Cut loci are points on the manifold where multiple minimizing geodesics from the source intersect.** At cut loci, the gradient of the distance field is not defined and the laplacian is $+\infty$. For manifolds with boundaries, we identify separator extremities as boundary points where the sum of geodesic distance gradients cancels out, similar to cut loci. In our implementations (Sec. 3), we omit this case.

One can visualize a circular wave emanating from the source: when the wavefront meets a protrusion, it splits and meets at the cut locus. The ideal local separator is the minimal loop connecting the wavefront split point to the cut locus and back. We construct these separators by optimizing an approximate loop from the target cut locus to the source through locally constrained curve shortening.

To create a set of local separators, we sample multiple (say, N) source points and repeat this procedure.

2.3. Stage 2: Constructing the Curve Skeleton

After obtaining potentially overlapping local separators, we need to select non-overlapping ones to divide the object into Voronoi regions. We score each separator and prune them through greedy set packing [22]. From the final set of non-overlapping separators, we identify neighboring regions and calculate average positions of points within regions to obtain skeleton nodes. We connect neighboring regions to form skeletal edges and remove cliques (e.g., triangles formed by connecting three neighboring regions) through iterative removal to create the final curve skeleton.

2.4. CSCD vs. LS

The LS procedure does not readily extend to continuous manifolds. In particular, (1) growing the separator set is non-trivial, as naively adding nearby points is inefficient and diverges from the original method, and (2) the absence of a clear neighborhood structure on manifolds complicates stopping criteria. We address these challenges by proposing a novel framework—a strict generalization of LS—for continuous manifolds.

3. Realization of CSCD on meshes /point clouds

Building on the above framework, we propose a method for both meshes (CSCD-M) and point clouds (CSCD-PC).

3.1. Stage1: Constructing Local Separators

3.1.1. Choice of the Geodesic Distance Method

We use the heat method for geodesic distance computation due to its efficiency and accuracy [9], making it well-suited for mesh and point cloud processing.

3.1.2. Identification of the Cut Loci

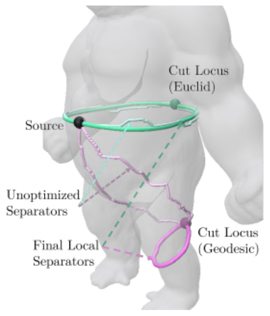
For both CSCD-M and CSCD-PC, we adapt the practical cut locus algorithm from [30], as detailed in Appendix G. The algorithm works by, starting from the farthest cut locus, identifying the cut loci as a connected graph on the surface of the mesh.

For `CSCD`-M, we adapt the algorithm for intrinsic triangulation, ensuring robustness to poor meshing. All gradient calculations in the procedure are restricted to the tangent spaces of vertices and faces through barycentric interpolation. Fig. 7a, Fig. 13 (in Appendix) compares our intrinsic implementation with the original on a poorly triangulated mesh.

For `CSCD`-PC, we introduce a novel method for cut locus identification on point clouds, with its reliability demonstrated in Fig. 7b, Fig. 14 (in Appendix).

3.1.3. Selecting the Target Cut Locus

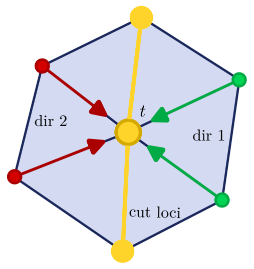
Following [2], we select the target cut locus t as the one with the smallest Euclidean distance to the source s, ensuring that the cut locus lies near a significant feature. Since intrinsic triangulation schemes lack explicit vertex positions in $\mathbb{R}^3$, we also test selecting t using the smallest geodesic distance (see App. J.1).



3.1.4. Approximate Path Construction

For `CSCD`-M, we search the neighborhood of the target cut locus for two incoming directions, separated by the cut loci graph (inset).

For `CSCD`-PC, a similar strategy is followed but with the additional requirement that the gradient directions are opposite. Two paths are constructed by greedily following vertices (or points) with the minimum geodesic distance from the source s, ensuring convergence at s. In cases where the paths meet at an intermediate vertex v, they are truncated at v. The concatenation of these paths forms a loop; Fig. 4 shows the approximate paths in red and lime green. The cut loci is visualized in yellow.



3.1.5. Optimizing the Loop

At this stage, the approximate loop only satisfies property **P1**. We aim to shorten the loop around the feature (see the cyan loop in Fig. 4). For `CSCD`-M, this is achieved using an edge flip procedure [39] in the intrinsic triangulation, while for `CSCD`-PC, an optimization-based framework from [60] is employed (see App. E and App. F).

3.1.6. Constraining the Loop

Simply shortening the curve yields a local geodesic loop that can drift significantly from the initial path. For example, a loop drawn at the bottom of a cone may slide upward toward the tip, which is undesirable for curve skeletonization.

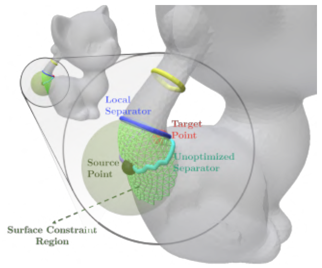


Figure 5. Constraint region for the loop optimization procedure. The green sphere shows the Euclidean sphere constraint.

Two observations guide our constraint: (1) the target cut locus is selected based on Euclidean proximity, and (2) the final separator should not lie farther from the source than the most distant point on the approximate path. Thus, we apply a bounding sphere constraint centered at the source s with a radius equal to the Euclidean distance from s to the furthest point on the approx. separator. In Fig. 5, with the constraint, the optimized separator (dark blue) remains in place, whereas without it the separator shifts upward (yellow).

In `CSCD`-M, the constraint restricts edge flips for vertices outside the sphere. In `CSCD`-PC, it is imposed as an interior point constraint in the optimization energy:

$$\mathcal{L} = \sum_{i=1}^{n} H\left(\|x_i - x_{i+1}\|_2\right) + \sum_{i=1}^{n} \lambda_i \max\left(0, \|x_i - x_s\|_2 - r\right)^2 , \tag{1}$$

where x_s, r, and H denote the source point, the Euclidean radius, and a kernel function, respectively.

3.1.7. Sampling the Separators

We avoid sampling regions that produce similar local separators by employing an adaptive sampling technique based on geodesic distances. The distance from each vertex to the constructed separators and sampled points is computed. Regions with larger distances are more likely to yield unique separators. The probability for a vertex i is given by $p_i \propto \exp(d_i) - 1$, where d_i is the minimum geodesic distance from vertex i to the existing separators and source points. The exponential weighting tends to emphasize separators around distant, sharp boundary features.

3.2. Stage 2: Constructing the Curve Skeleton

3.2.1. Scoring the Separators

We define a score for each separator to decide among overlapping candidates. Inspired by LS, a good local separator balances the two halves of the surface constraint region mentioned in sec. 3.1.6. Instead of simply counting nodes, we weigh based on the ratio of the surface areas of the two components and penalize longer loops. The final score is $s_i = \frac{A_{1,i}}{A_{2,i} \cdot l_i}$, where $A_{1,i}$ and $A_{2,i}$ (with $A_{2,i} > A_{1,i}$) are the surface areas of the two components, and l_i is the loop length.

3.2.2. Pruning Bad Separators

Tiny separators that do not enclose a significant feature are pruned using a length threshold $\tau = 3 \times \bar{d_{ij}}$, where $\bar{d_{ij}}$ is the average edge length. Separators forming handles rather than properly encircling features are removed by discarding those whose centroids lie outside the 3D shape.

3.2.3. Packing the Separators

After scoring, set packing is performed to suppress overlapping separators. We normalize the weights based on the opportunity cost of retaining one separator over another and greedily select those with the highest normalized weights. Separators that do not overlap are retained automatically.

For CSCD-M, overlapping separators are identified by testing intersections of piecewise linear curves within each face (**See Supplementary Sec. C for Derivation on determining intersection within a face**). For CSCD-PC, a distance threshold between points is used. With the selected set of non-overlapping separators, we proceed to construct the curve skeleton graph.

3.2.4. Assigning Regions

Neighboring regions of the object are assigned to each local separator by computing the geodesic distance from every vertex (or point) to each separator, followed by a Voronoi partitioning.

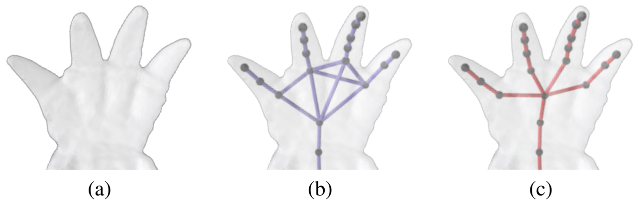


Figure 6. Cliques in the armadillo hand and their removal. (a) Original hand; (b) With cliques; (c) Cliques replaced by central star-like nodes.

3.2.5. Constructing the Graph

The centroids of the regions form the nodes of the curve skeleton graph. Nodes corresponding to neighboring regions are connected. When a region neighbors more than two others, resulting cliques are simplified by converting them to a star formation, using the centroid as the central node and removing redundant edges (see Fig. 6).

3.3. On the Discrete Nature of the Realizations

Manifold representations are inherently discrete in computers; thus, our realizations of CSCD are discrete and involve tradeoffs similar to those in adapting geodesic paths (e.g., straightest vs. shortest paths). While LS relies solely on node and edge data, our framework benefits from additional face-level information for meshes, allowing for interpolation across faces. In point clouds, local separators are constructed using dynamically computed neighborhood information. Although the complete graph is not built initially, many computations are reused, and techniques such as MLS or tangent space smoothing can refine the cut loci and loop approximations. For other discrete representations (e.g., digital surfaces), our method remains discrete, though it could incorporate continuous-level corrections if available.

4. Results

4.1. Results of the Improved Cut Loci Identification

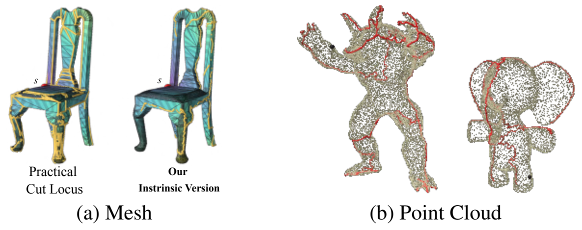


Figure 7. (a) Comparison of our (right) estimated cut loci (*in yellow*) versus the previous approach (left) [30] (of source s) on chair from Thingi10k. **Note:** Our adaptation generates robust output by selectively identifying points on the true cut loci, thereby significantly minimizing false negatives; (b) our novel cut loci identification algorithm applied to point clouds. Source point is shown in black and the resulting cut loci are rendered as red curves.

We show the results of our cut locus identification algorithm in Fig. 7a and Fig. 7b. Our adaptation produces robust output that selectively chooses the points on the cut loci, thereby significantly reducing the false negatives. For details see Appendix G.

4.2. Curve Skeletonization on Meshes

General performance of CSCD-M: We evaluate and compare our method to [2, 3, 45, 46] on diverse objects mostly from the Stanford 3D Library, Artec 3D Scans, and Thingi10k datasets. Figs. 1 and 8 illustrate that our curve skeletonization is topologically correct and faithfully follows the object geometry. In comparison to ROSA and MCF—which often miss key features—our method (and LS) retains more details. Notably, our skeletons are smoother and yield better-centered nodes without additional smoothing; we suspect this is due to weighing centroid computations with vertex and face areas. In the case of Copper Key, our method uniquely captures the intricate design (See Fig. 1 (b)).

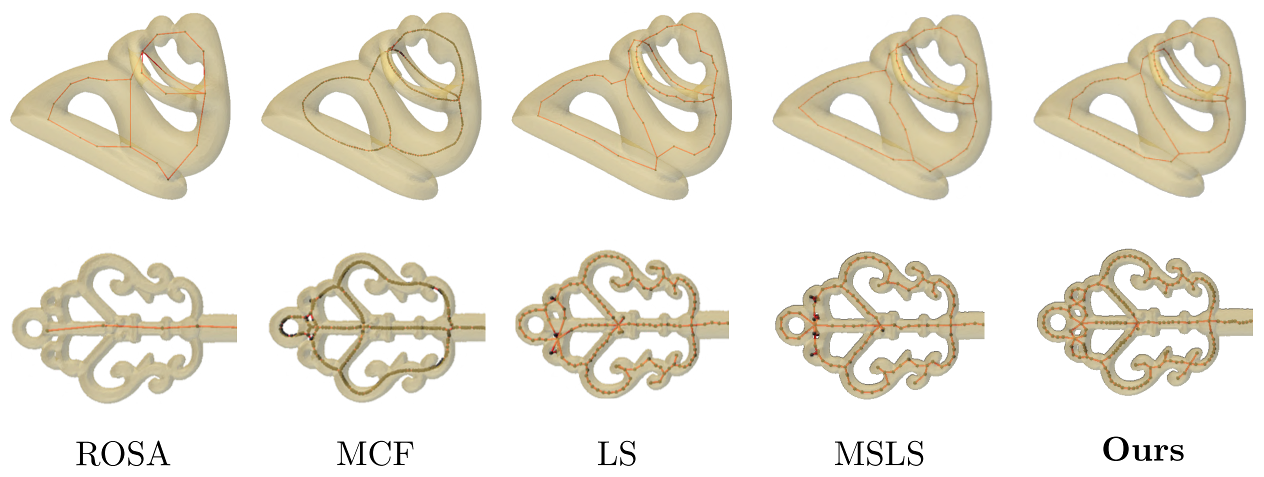


Figure 8. **Qualitative results of our method:** ROSA [45] struggles to capture mesh details, while MCF [46] produces overly smooth skeletons. `CSCD`-M, LS [2] and MSLS [3] yield comparable results on meshes; however, our method correctly captures details, as seen on the copper key. In `fertility`, our approach results in smoother skeletons compared to LS and MSLS.

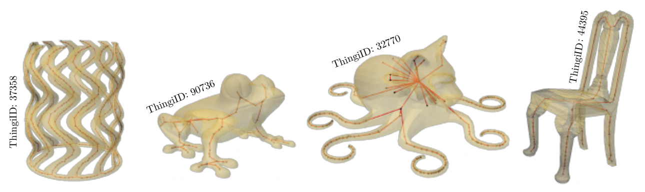


Figure 9. **Qualitative results of `CSCD`-M on Thingi10k.** `CSCD`-M performs well even on poorly triangulated meshes.

Performance on poorly triangulated meshes: Our intrinsic triangulation scheme, based on the Integer coordinate system [16], uses intrinsic edge flips to enforce Delaunay conditions. As shown in Fig. 9, our method reconstructs skeletons effectively on such meshes; Tab. 5. Also see Fig. 18 for results on noisy meshes. Fig. 19.

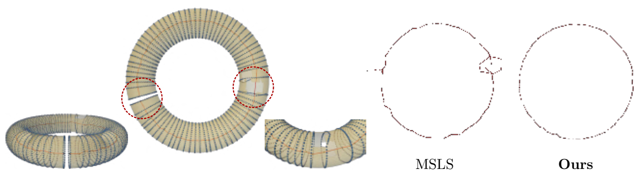


Figure 10. `CSCD`-M and MSLS on a **torus with holes**. Left panel, **`CSCD`-M local separators on the torus** – highlighting how the separators go around the holes. Right panel, **comparison of the curve skeleton** obtained by MSLS and Ours (`CSCD`-M).

Performance on meshes with holes: We evaluate on a torus mesh with three holes—two partial and one fully disconnecting the shape—as a controlled test case. Our method is the only one to perform reliably (Fig. 10); LS fails to produce output; MSLS recovers incorrect topology.

Quantitative performance of `CSCD`-M: Due to the lack of a standard quantitative metric for curve skeletonization, we propose a reconstruction loss based on convolutional surfaces [42], where the error at vertex i is $\epsilon_i = \min_{\hat{p}\in\hat{\mathcal{M}}} \|\mathbf{p}_i - \hat{p}\|^2$, with $\hat{p}$ any point on the reconstructed mesh $\hat{\mathcal{M}}$ (see App. D). As shown in Table 2, our method outperforms LS, likely due to smoother, more centered skeletons.

Inference times: Our method on average performs faster than LS on our subset of objects ranging in different complexities (see Table 3). For additional details see App. H.6.

4.3. Curve Skeletonization on Point Clouds

We demonstrate a proof-of-concept realization of `CSCD` on point clouds (`CSCD`-PC), which captures detailed, well-centered skeletons (Figs.11,12), outperforming ROSA, CA++, and EPCS, which yield coarser skeletons with fewer nodes.

Table 2. *(Truncated)* Convolutional Surfaces reconstruction error ($\times 10^{-3}$) for objects using skeletons from `CSCD`-M (Ours), MCF [46], LS [2] and MSLS [3]. Average is computed over the subset here. Complete Table 5.

Object	LS	MSLS	MCF	`CSCD`-M **(Ours)**
`Copper-key`	06.13	**04.60**	6.02	04.70
`rocker-arm`	26.30	**24.40**	25.30	24.90
`neptune`	04.16	04.82	10.62	**03.80**
`TID:44395`	**09.56**	10.70	16.80	10.07
`TID:40987`	09.29	09.81	11.36	**08.32**
`TID:133568`	**04.75**	05.49	05.51	04.99
Average	10.03	9.97	12.60	**9.46**

Table 3. *(Truncated)* Runtime analysis (in secs.) for CSCD-M (Ours), LS [2] (TOG'21) and MSLS [3]. Results are based on $N = 3K$ (with three cases using $N = 4K$ due to mesh complexity). LS times for gorilla are omitted from averaging due to excessive computation time. Averaging is computed over the complete table. Complete Tab. 6

Object	$\|V\|$	$\|F\|$	LS	MSLS	CSCD-M (Ours)
TID:44395	2948	5900	6.78	2.14	37.65
fertility	4494	9000	9.39	2.79	41.43
TID:32770	20125	40246	1889.70	13.20	225.17*
gorilla	48762	97520	$\geq 2400^{\dagger}$	32.46	532.16
armadillo	49990	99976	994.75	33.70	495.33
garuda-vishnu	49972	100084	292.43	30.13	609.30*
neptune	50000	100008	886.68	31.53	611.66*
Average			471.17	16.14	277.99

Table 4. **Application I: Shape Classification:** For subset of classes from the Princeton Shape Benchmark dataset [41].

Metric	LS	MSLS	CSCD-M
Accuracy	0.63	0.74	**0.79**
F1 Score	0.59	0.76	**0.80**

4.4. Downstream Applications

4.4.1. Object Classification

We compare curve skeletonization methods for object classification on the Princeton Shape Benchmark [41]. A global shape embedding is constructed via a histogram of the Shape Diameter Function (SDF) [37], and the 1D Wasserstein distance between histograms is used for comparison, ensuring robustness against discretization. Our method outperforms contemporary approaches (see, Table 4).

4.4.2. Object Segmentation

We also evaluate our outputs for unsupervised object segmentation using SDF [37], yielding consistent results robust to pose variations (Fig. 21).

Note: App. for more results, ablations, downstream task.

5. Conclusion and Future Work

We introduced CSCD, a general framework for curve skeletonization on continuous manifolds that generalizes LS. Its effectiveness is demonstrated through implementations on meshes (CSCD-M) and point clouds (CSCD-PC). CSCD-M is the first intrinsic curve skeletonization method and shows robust performance across diverse meshes, with results that are comparable or superior to the state-of-the-art LS. Meanwhile, CSCD-PC provides a compelling proof-of-concept for the framework's generalizability. Our realizations are intended as starting points, with results on meshes and point clouds. Future work could improve individual modules for improved speed, robustness, and performance, or extend the framework to other representations. We plan to release the source code post acceptance.

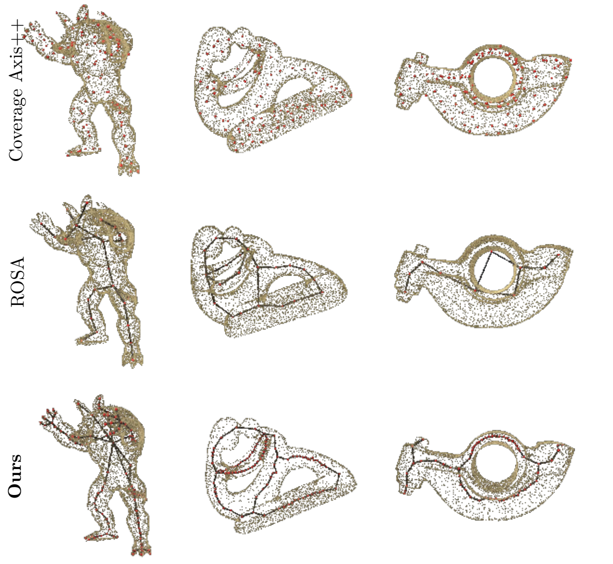


Figure 11. **CSCD-PC vs ROSA [45] vs CA++ [Eurographics'24][53]:** CA++ fails to generate a valid skeleton (since it's a MAT inspired algorithm). Our method captures object details better, yielding more nodes and centered skeletons compared to ROSA.

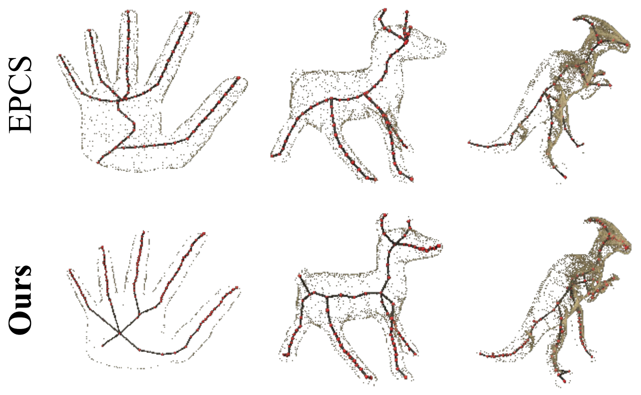


Figure 12. **CSCD-PC vs EPCS [CAG'23]:** Our method captures object details better, compared to EPCS [23]. In hand (left), our skeleton is centered in the palm and shows skeleton consistent with square shape. In deer (middle), we capture the snout and the tail. In dino (right), EPCS fails to capture the curvature of the arm completely while our skeleton follows the arm.

6. Acknowledgments

We would like to thank Rahul Narain for the insightful initial discussions on shape representation, which laid the foundation for this work. We would also like to thank Cyrin Neeraj for his help on results presentation.

References

[1] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Trans. Graph.*, 27(3):1–10, 2008. 2, 20

[2] Andreas Bærentzen and Eva Rotenberg. Skeletonization via local separators. *ACM Trans. Graph.*, 40(5), 2021. 1, 2, 5, 6, 7, 8, 10, 20

[3] J. Andreas Bærentzen, Rasmus Emil Christensen, Emil Toftegaard Gæde, and Eva Rotenberg. Multilevel skeletonization using local separators. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG 2023)*, 2023. 2, 6, 7, 8, 5, 10, 20

[4] Justin L Brown, Takuya Furuta, and Wesley E Bolch. A robust algorithm for voxel-to-polygon mesh phantom conversion. *Brain and Human Body Modeling: Computational Human Modeling at EMBC 2018*, pages 317–327, 2019. 20

[5] Jingliang Cheng, Xinyu Zheng, Shuangmin Chen, Guozhu Liu, Shiqing Xin, Lin Lu, Yuanfeng Zhou, and Changhe Tu. Skeletonization via dual of shape segmentation. *Computer Aided Geometric Design*, 80:101856, 2020. 2, 20

[6] Y. Chu, W. Wang, and L. Li. Robustly extracting concise 3d curve skeletons by enhancing the capture of prominent features. *IEEE Trans Vis Comput Graph*, 29(8):3472–3488, 2023. Epub 2023 Jun 29, PMID: 35324442. 2

[7] David Coeurjolly and Jacques-Olivier Lachaud. A simple discrete calculus for digital surfaces. In *Discrete Geometry and Mathematical Morphology*, pages 341–353, Cham, 2022. Springer International Publishing. 12

[8] Nicu D Cornea, Deborah Silver, and Patrick Min. Curve-skeleton applications. In *VIS 05. IEEE Visualization, 2005.*, pages 95–102. IEEE, 2005. 2, 20

[9] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. The heat method for distance computation. *Commun. ACM*, 60(11):90–99, 2017. 4, 2, 19

[10] Keenan Crane, Marco Livesu, Enrico Puppo, and Yipeng Qin. A survey of algorithms for geodesic paths and distances. *ArXiv*, abs/2007.10430, 2020. 19

[11] Zhiyang Dou, Cheng Lin, Rui Xu, Lei Yang, Shiqing Xin, Taku Komura, and Wenping Wang. Coverage axis: Inner point selection for 3d shape skeletonization. *Computer Graphics Forum*, 41(2):419–432, 2022. 2, 20

[12] Zhiyang Dou, Cheng Lin, Rui Xu, Lei Yang, Shiqing Xin, Taku Komura, and Wenping Wang. Coverage axis: Inner point selection for 3d shape skeletonization. In *Computer Graphics Forum*, pages 419–432. Wiley Online Library, 2022. 2, 20

[13] Bastein Durix, Géraldine Morin, Sylvie Chambon, Céline Roudet, and Lionel Garnier. Towards skeleton based reconstruction: From projective skeletonization to canal surface estimation. In *2015 International Conference on 3D Vision*, pages 545–553, 2015. 2

[14] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM SIGGRAPH 2005 Papers*, 2005. 4

[15] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. 1

[16] Mark Gillespie, Nicholas Sharp, and Keenan Crane. Integer coordinates for intrinsic geometry processing. *arXiv preprint arXiv:2106.00220*, 2021. 7

[17] Wooi-Boon Goh. Strategies for shape matching using skeletons. *Computer Vision and Image Understanding*, 110(3): 326–345, 2008. Similarity Matching in Computer Vision and Multimedia. 2

[18] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Trans. Graph.*, 32(4), 2013. 2, 20

[19] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 2

[20] M.W. Jones, J.A. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12 (4):581–599, 2006. 1

[21] Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences*, 95(15):8431–8435, 1998. 2

[22] David Kordalewski. New greedy heuristics for set cover and set packing. *ArXiv*, abs/1305.3584, 2013. 4

[23] Chunhui Li, Mingquan Zhou, Guohua Geng, Yifei Xie, Yuhe Zhang, and Yangyang Liu. Epcs: Endpoint-based part-aware curve skeleton extraction for low-quality point clouds. *Computers & Graphics*, 117:209–221, 2023. 2, 8, 20

[24] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG)*, 35(1):1–16, 2015. 2

[25] Chu-Hsing Lin, Changjian Li, Yuan Liu, Nenglun Chen, Yi-King Choi, and Wenping Wang. Point2skeleton: Learning skeletal representations from point clouds. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4275–4284, 2020. 2, 20

[26] Bangquan Liu, Shuangmin Chen, Shi-Qing Xin, Ying He, Zhen Liu, and Jieyu Zhao. An optimization-driven approach for computing geodesic paths on triangle meshes. *Computer-Aided Design*, 90:105–112, 2017. SI:SPM2017. 20

[27] Marco Livesu, Fabio Guggeri, and Riccardo Scateni. Reconstructing the curve-skeletons of 3d shapes using the visual hull. *IEEE Transactions on Visualization and Computer Graphics*, 18:1891–1901, 2012. 2, 20

[28] Chenlei Lv, Weisi Lin, and Baoquan Zhao. Voxel structure-based mesh reconstruction from a 3d point cloud. *IEEE Transactions on Multimedia*, 24:1815–1829, 2022. 20

[29] Shubh Maheshwari, Rahul Narain, and Ramya Hebbalaguppe. Transfer4d: A framework for frugal motion capture and deformation transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12836–12846, 2023. 2

[30] C. Mancinelli, M. Livesu, and E. Puppo. Practical computation of the cut locus on discrete surfaces. *Computer Graphics Forum*, 40(5):261–273, 2021. 4, 6, 2, 5

[31] Dimas Martínez, Luiz Velho, and Paulo C. Carvalho. Computing geodesics on triangular meshes. *Computers and Graphics*, 29(5):667–675, 2005. 20

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65:99–106, 2020. 1

[33] D Nicu, C Silver, and D Silver. Curve-skeleton properties, applications and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007. 2, 20

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 2

[35] Dennie Reniers, Jarke van Wijk, and Alexandru Telea. Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):355–368, 2008. 2, 20

[36] Punam K. Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters*, 76:3–12, 2016. Special Issue on Skeletonization and its Application. 2, 20

[37] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24:249–259, 2008. 8

[38] Nicholas Sharp and Keenan Crane. A laplacian for nonmanifold triangle meshes. *Computer Graphics Forum*, 39(5):69–80, 2020. 12

[39] Nicholas Sharp and Keenan Crane. You can find geodesic paths in triangle meshes by just flipping edges. *ACM Trans. Graph.*, 39(6), 2020. 5, 2, 4, 20

[40] Nicholas Sharp, Keenan Crane, et al. Geometrycentral: A modern c++ library of data structures and algorithms for geometry processing. *github*, 2019. 2

[41] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Proceedings Shape Modeling Applications, 2004.*, pages 167–178. IEEE, 2004. 8

[42] Alvaro Javier Fuentes Suarez. *Modeling shapes with skeletons: scaffolds & anisotropic convolution*. PhD thesis, COMUE Université Côte d'Azur (2015-2019), 2019. 7, 3

[43] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *2003 Shape Modeling International.*, pages 130–139, 2003. 2

[44] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J Gortler, and Hugues Hoppe. Fast exact and approximate geodesics on meshes. *ACM transactions on graphics (TOG)*, 24(3):553–560, 2005. 2

[45] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph.*, 28(3), 2009. 2, 6, 7, 8, 10, 13, 20

[46] Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. Mean curvature skeletons. *Computer Graphics Forum*, 31(5):1735–1744, 2012. 2, 6, 7, 5, 20

[47] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3d skeletons: A state-of-the-art report. *Computer Graphics Forum*, 35(2):573–597, 2016. 2, 20

[48] G. Taubin. Geometric Signal Processing on Polygonal Meshes. In *Eurographics 2000 - STARs*. Eurographics Association, 2000. 1

[49] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. 3D Mesh Skeleton Extraction Using Topological and Geometrical Analyses. In *14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006)*, page s1poster, Tapei, Taiwan, 2006. 2, 20

[50] S. R. S. Varadhan. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20(2):431–455, 1967. 19

[51] He Wang and Juyong Zhang. A survey of deep learning-based mesh processing. *Communications in Mathematics and Statistics*, 10, 2022. 1

[52] Ningna Wang, Bin Wang, Wenping Wang, and Xiaohu Guo. Computing medial axis transform with feature preservation via restricted power diagram. *ACM Transactions on Graphics (TOG)*, 41(6):1–18, 2022. 2

[53] Zimeng Wang, Zhiyang Dou, Rui Xu, Cheng Lin, Yuan Liu, Xiaoxiao Long, Shiqing Xin, Taku Komura, Xiaoming Yuan, and Wenping Wang. Coverage axis++: Efficient inner point selection for 3d shape skeletonization. In *Computer Graphics Forum*, page e15143. Wiley Online Library, 2024. 2, 8, 20

[54] Udaranga Wickramasinghe, Edoardo Remelli, Graham Knott, and Pascal Fua. Voxel2mesh: 3d mesh model generation from volumetric data. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part IV 23*, pages 299–308. Springer, 2020. 20

[55] Shi-Qing Xin and Guo-Jin Wang. Efficiently determining a locally exact shortest path on polyhedral surfaces. *Computer-Aided Design*, 39(12):1081–1090, 2007. 20

[56] Shi-Qing Xin, Ying He, and Chi-Wing Fu. Efficiently computing exact geodesic loops within finite steps. *IEEE Transactions on Visualization and Computer Graphics*, 18(6): 879–889, 2012. 20

[57] Baorong Yang, Junfeng Yao, Bin Wang, Jianwei Hu, Yiling Pan, Tianxiang Pan, Wenping Wang, and Xiaohu Guo. P2mat-net: Learning medial axis transform from sparse point clouds. *Computer Aided Geometric Design*, 80: 101874, 2020. 2, 20

[58] Zipeng Ye, Yong-Jin Liu, Jianmin Zheng, Kai Hormann, and Ying He. De-path: A differential-evolution-based method for computing energy-minimizing paths on surfaces. *Computer-Aided Design*, 114:73–81, 2019. 20

[59] Na Yuan, Peihui Wang, Wenlong Meng, Shuangmin Chen, Jian Xu, Shiqing Xin, Ying He, and Wenping Wang. A variational framework for curve shortening in various geometric domains. *IEEE Transactions on Visualization and Computer Graphics*, 29(4):1951–1963, 2023. 4

[60] N. Yuan, P. Wang, W. Meng, S. Chen, J. Xu, S. Xin, Y. He, and W. Wang. A variational framework for curve shortening

in various geometric domains. *IEEE Transactions on Visualization and; Computer Graphics*, 29(04):1951–1963, 2023. 5, 2, 20

[61] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 2