


## Article

# Mixed 1D/2D Simplicial Approximation of Volumetric Medial Axis by Direct Palpation of Shape Diameter Function

Andres F. Puentes-Atencio <sup>1,\*</sup> , Daniel Mejia-Parra <sup>1</sup>, Ander Arbelaiz <sup>1</sup>, Carlos Cadavid <sup>2</sup> and Oscar Ruiz-Salguero <sup>1</sup>

<sup>1</sup> Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009 Donostia-San Sebastián, Spain; dmp2627@gmail.com (D.M.-P.); aarbelaiz@vicomtech.org (A.A.); oruiz@eafit.edu.co (O.R.-S.)

<sup>2</sup> School of Applied Sciences and Engineering, Universidad EAFIT, Cra. 49 No. 7 Sur-50, Medellín 050022, Colombia; ccadavid@eafit.edu.co

\* Correspondence: afpuentes@vicomtech.org

## Abstract

In the domain of Shape Encoding, the approximation of the Medial Axis of a solid region in  $\mathbb{R}^3$  with Boundary Representation  $M$ , is relevant because the Medial Axis is an efficient encoding for  $M$  in Design, Manufacturing, and Shape Learning. Existing Medial Axis approximations include (a) full Voronoi and (b) and partial Shape Diameter Function (SDF)-based ones. Methods (a) produce large high-frequency data, which must then be pruned. Methods (b) reduce computing expenses at the price of not handling some shapes (e.g., prismatic), and currently, they only synthesize 1D Medial Axes. To partially overcome these limitations, this investigation performs a direct synthesis of a 1D and 2D simplex-based Medial Axis approximation by a combination of stochastic geometric reasoning and graph operations on the SDF-originated point cloud. Our method covers one- and two-dimensional Simplicial Complex Medial Axes, thus improving on 1D Medial Axes approximation methods. Our approach avoids the expensive full computing plus pruning of Medial Axis based on Voronoi methods. Future work is needed in the synthesis of Medial Axis approximation for high-frequency neighborhoods of mesh  $M$ .

**Keywords:** 1D/2D medial axis; simplicial complex; direct palpation; shape diameter; point cloud; manifold learning



Academic Editor: Frank Werner

Received: 2 July 2025

Revised: 14 August 2025

Accepted: 28 August 2025

Published: 31 August 2025

**Citation:** Puentes-Atencio, A.F.; Mejia-Parra, D.; Arbelaiz, A.; Cadavid, C.; Ruiz-Salguero, O. Mixed 1D/2D Simplicial Approximation of Volumetric Medial Axis by Direct Palpation of Shape Diameter Function. *Algorithms* **2025**, *18*, 546. <https://doi.org/10.3390/a18090546>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Skeletal representations allow for the synthesis of the underlying geometry and topology properties of a 3D object. They have been useful in applications such as shape reconstruction [1–3], segmentation [4], abstraction [5], recognition [6,7], and animation [8], among others.

The Medial Axis (MA) is a commonly used skeletal representation. The MA of a 3D body  $\mathcal{B}$  is defined as the set of centers and their respective radii of all the maximally inscribed balls (medial balls) in  $\mathcal{B}$  [9]. This MA generally contains intermixed 1D and 2D elements. Therefore, a Piecewise Linear (PL) representation of MA is generally a non-manifold simplicial complex. A full MA of  $\mathcal{B}$  normally contains highly fibrous (small hair-like or sheet-like) structures. Hence, a simplification is usually applied to the Medial Axis after its computation in order to obtain a less noisy result (which in this manuscript we call *Skeleton*).

Notice that each point of the skin  $\partial\mathcal{B}$  of  $\mathcal{B}$  is related to the ball centered in a point of the Medial Axis, with radius  $r$ , which is tangent to that point of the skin  $\partial\mathcal{B}$ . This theoretical

and exhaustive Medial Axis Radius scalar field is approximated by a more economical version, called the Shape Diameter Function (SDF [10]). Obviously, the Shape Diameter Function approximates twice the corresponding radius  $r$  ( $SDF \approx 2r$ ).

The Shape Diameter Function (SDF [10]) is an economical intermediate scalar field, useful to approximate the Medial Axis for particular classes of solids. SDF is defined as the “thickness” of the solid at each point on its surface, and it is a scalar function  $SDF : M \rightarrow \mathbb{R}^+$ . The SDF-based point cloud is the locus of the “midpoints” of the thickness segment for all points of  $M$  (see Section 3.5 for formal definition). Therefore, the point cloud derived from the SDF scalar field is an approximation of the Medial Axis.

This investigation executes a direct palpation of the SDF-based point cloud. The term “palpation” is analogous to the actions executed by a blind person, who seeks to apprehend the nature of a point cloud: the person would directly touch the point cloud neighborhoods, seeking to identify 1D (PL curve) or 2D (PL surface) regions. This direct palpation executes the synthesis of the 1D/2D Simplicial Complex, which approximates the Medial Axis of the solid, and, unlike previous SDF approaches, ours includes the *mixture of 1D and 2D simplicial Medial Axes*. Additionally, unlike other approximations of Medial Axis by Voronoi methods, which require extensive pruning of MA fibrosities, ours computes a Skeleton directly, without generating fibrosities.

Our method inherits the advantages of SDF of being an economical estimation of the Medial Axis Radius Scalar Field. However, it also inherits the disadvantage of SDF of failing to approximate the Medial Axis Radius scalar field in the cases of prismatic regular solids.

This manuscript is divided as follows: Section 2 reviews the existing literature and draws the conclusions about the current status. Section 3 explains the applied methodology. Section 4 displays the results of our implementation, and Section 5 concludes the article and sketches possible future research directions.

A glossary of terms is provided at the end of the manuscript, before the references, to help readers clarify terminology as needed.

## 2. Literature Review

We review some of the most relevant existing 3D shape skeletonization methods. For more exhaustive information on different methods, we refer the reader to Tagliasacchi et al. [11].

### 2.1. Mesh Contraction

The Mesh Contraction method [12] iteratively performs geometry collapses on an input surface mesh until the shape volume is close to zero. Constrained Laplacian smoothing is used for the geometry collapses and then the contracted mesh is converted into a curve Skeleton. Cao et al. [13] extends the Mesh Contraction method by allowing point cloud inputs and improving on the conversion from the contracted shape to a curve Skeleton.

### 2.2. Sphere-Meshes

Thiery et al. [14] proposes a different shape approximation structure called Sphere-Meshes. Given a surface mesh input, their algorithm produces a user-specified number of connected spheres that minimize a distance function with respect to the input mesh.

### 2.3. Voxel Thinning

Voxel thinning methods [15,16] use an image model of the input shape and the output Skeleton. These methods work mainly by removing boundary voxels according to different criteria until a *thin* model is obtained. Moreno et al. [17] produces a Graph Skeleton from the result of the thinning process. In both [17,18], it is shown that when trying to force a

curve Skeleton representation for a general 3D shape, the thinning algorithm produces more fibrosities.

#### 2.4. Shape Diameter Function-Based Methods

Shapira et al. [10] present a method that uses Moving Least Squares projections on an SDF-based point cloud to build a Medial Axis approximation. However, the obtained Skeleton consists only of 1D segments, failing to accurately represent shapes with 2D structures. We have not found references in addition to [10] for MA using SDF-based point clouds. This absence has a relation to our decision to work in this direction.

#### 2.5. Voronoi Medial Axis Simplification

The Voronoi Medial Axis simplification methods first extract an initial, large, noisy Medial Axis using Voronoi diagrams. Later, according to different criteria, pruning or mesh simplification is iteratively performed. Medial Meshes [19] simplify the initial Medial Axis by iteratively performing edge contractions until a shape approximation error reaches a threshold.

In  $\lambda$ -MAT [20], medial balls are discarded so that only those that have a radius of at least  $\lambda$  are preserved.  $\lambda$ -MAT can lead to an important loss of features at different scale structures of the input shape.

Scale Axis Transform (SAT) [21] scales the medial balls by some factor  $s > 1$  and deletes those that end up contained in another ball, i.e., are no longer maximal. Then, it scales the medial balls back by a factor  $1/s$ . Progressive Medial Axis [22] extends SAT by performing edge collapses based on ball absorption. SAT computes multiple Voronoi diagrams, which result in low computational efficiency.

Mean Curvature Skeletons [23] perform iterative Mesh Contraction based on the Mean Curvature Flow due to its area-reducing properties. However, it requires the input mesh to be significantly dense.

Q-MAT [24] iteratively performs edge collapses on the initial Voronoi extracted MA based on the QEM framework [25]. Q-MAT final result depends on the quality of the initial MA used for simplification. To yield a better quality initial MA, Q-MAT+ [26] uses the SDF to identify thin parts and increases the sampling density in those thin regions.

Coverage Axis [27] uses the Set Cover Problem to select a minimum number of inner points whose dilated balls cover a set of sampled points that lie on the object's surface. Then, Coverage Axis follows the approach of Q-MAT [24] to obtain a connected MAT, with the addition of preserving the previously identified inner points. However, Coverage Axis [27] suffers from a high computational burden for shapes with planar structures. Coverage Axis++ [28] formulates a different inner point selection algorithm that runs more efficiently.

#### 2.6. Point Cloud Skeletons

Point cloud skeletonization methods allow for inputs lacking manifold information about the shape's underlying surface. Least Squares MAT [29] computes a Skeleton using information of a Signed Distance Function from an oriented point cloud.

The Iterative Medial Axis Transform [30], iteratively identifies a set of maximally inscribed balls that minimizes a shape disparity function.

Ma et al. [31] compute a set of Medial Axis points from an oriented point cloud sampled from a surface. For each sampled point  $p$ , a ball tangent to  $p$  is iteratively shrunk until it is maximal. The ball has the restriction of having its center located along the line that passes through  $p$  in the direction of the normal of point  $p$ . The aforementioned methods, however, lack topology information in the Medial Axis.

Jalba et al. [32] improve Ma et al. [31] performance and proposes methods to obtain a surface Skeleton from the computed Medial Axis points by applying Delaunay triangulation or a Ball Pivoting Algorithm on said points.

Coverage Axis [27] also allows for point cloud inputs, in which case it uses the Power Diagram following Amenta et. al. [2].

## 2.7. Learning-Based Medial Axis

Point2MM [33] learns a geometric transformation to predict the medial spheres of a point cloud. Later, it predicts the connectivity of the medial spheres to form a medial mesh. Neural Skeleton [34] uses Implicit Neural Representation training on a point cloud to extract skeletal points. Then, it follows Coverage Axis [27] to retrieve the Skeleton mesh. These methods suffer mainly from requiring training time and data in order to achieve satisfactory results.

The present manuscript does not intend to explore Artificial Intelligence learning methods for Shape Encoding. Our work strictly lies in the domain of Applied Computational Geometry.

## 2.8. Conclusions of Literature Review

We present the conclusions of the Literature Review in Table 1, where we mention relevant Medial Axis approximation approaches and some of their main advantages and disadvantages.

**Table 1.** Conclusions of the literature review. Our approach is compared against relevant Medial Axis approximation methods. First column: name of the approach. Second column: references for the corresponding approach. Third and fourth columns: main advantages and disadvantages of each corresponding approach, respectively.

Approach	Refs.	Main Advantages	Main Disadvantages
Mesh Contraction	[12,13]	(1) Can handle noisy input. (2) The Skeletons produced are pose invariant.	(1) Computationally expensive. (2) Only produces 1D Skeletons.
Sphere-Meshes	[14]	(1) Can handle incomplete datasets. (2) Represents shapes using very few primitives.	(1) The produced Skeleton can lie outside the input mesh. (2) Topology is not always preserved.
Voxel Thinning	[15–18]	(1) Direct compatibility with volumetric data. (2) Can soften the effect of noise on surfaces.	(1) The Skeleton is constrained to the fixed Voxel grid. (2) Skeleton centeredness can be lost.
Voronoi Medial Axis Simplification	[20–24,26,27]	(1) Can be error-controllable. (2) Have topological information. (3) Allows for a high accurate approximation of the input shape.	(1) Highly dependent on the initial Medial Axis quality. (2) Large noise can yield unstable results.
Point Cloud Skeletons	[27,29–31]	(1) Can handle both mesh and point cloud inputs. (2) Can handle noisy input.	(1) Lacks connectivity information.
Learning-based Medial Axis	[33–35]	(1) Can handle input with missing parts for trained data.	(1) Poor results for new data. (2) Requires training time for satisfactory results.
Direct Palpation on the SDF-based Point Cloud	Our approach	(1) Does not require the full computation of Voronoi-based Medial Axis with all fibrillations, which then must be eliminated. (2) Delivers mixed 1D/2D Medial Axes, unlike competitor SDF-based methods ([10]).	(1) The SDF-based point cloud does not always approximate the Medial Axis. (2) It is vulnerable in high-frequency neighborhoods of $M$ (creases, pits, joints).

In this paper, we present a method for the computation of the Skeleton of a mesh  $M$  in an economical and faster way than existing approaches. We do not deal with the reconstruction based on Skeleton or object skin segmentation, which are downstream applications of the Medial Axes.

### 3. Methodology

#### 3.1. Scope

We want to reiterate that Medial Axes or Skeletons are essentially filters. Therefore, according to the characteristics of the filter, some parts of the original solid may be lost. It is not within the scope of this paper to quantify the effect of the filter on the reconstruction of the object based on the Skeleton.

It is also important to note that, in our approach, we frequently use local Principal Component Analysis (PCA) in order to identify Piecewise Linear approximations for curves and surfaces within a point cloud. We do not try to approximate a curve or a surface with only *one* straight edge or only *one* planar surface. In any case, the initial triangular meshes must be very dense in high frequency neighborhoods in order to respect the Whittaker–Nyquist–Shannon theorem [36], regardless of any ulterior computation (e.g., Medial Axis, Resampling, Finite Element, etc.).

A glossary is presented at the end of this manuscript, before the references, providing definitions for terms used throughout the text.

#### 3.2. Manifold Learning by Direct Palpation

Our algorithm probes local neighborhoods of the SDF-based point cloud, seeking locally planar surfaces or locally straight segments. This means our algorithm tests the local neighborhoods of the SDF-based point cloud for compliance with the definition of a 2-manifold with border or a 1-manifold with border. Such definitions follow [37].

**Definition 1** ( $k$ -manifold with border.  $k = 1$  for 1-manifolds or curves,  $k = 2$  for 2-manifolds for surfaces). A set  $M \subset \mathbb{R}^3$  is a  $k$ -manifold with Border if for each point  $p \in M$  there exists a  $\delta \in \mathbb{R}^+$  such that for all radius  $r$  with  $0 < r < \delta$ ,  $B(p, r) \cap M$  is isomorphic to either (1) the disk  $S^k$  or (2) the half-disk  $S^k/2$ .

This definition informally means that, locally, the neighborhoods look like (a) flat surfaces (2-manifolds) or (b) straight line segments (1-manifolds). Therefore, they can be identified with Principal Component Analysis (PCA).

Our algorithm attempts to identify all point cloud regions that are locally homeomorphic to a plane or to a straight segment. This is an instance of Manifold Learning. This identification is executed through PCA. The regions in which this probe fails (i.e., are neither planes or lines) are graded as “Gray” and processed differently, as they indicate high frequency regions/junctions (Section 3.7).

PCA has been extensively used in processing and characterizing point cloud data. For example, in [38,39] PCA is used for plane fitting on local point cloud regions.

### 3.3. Simplices and Simplicial Complexes

We use the following definitions of simplices and simplicial complexes [40]:

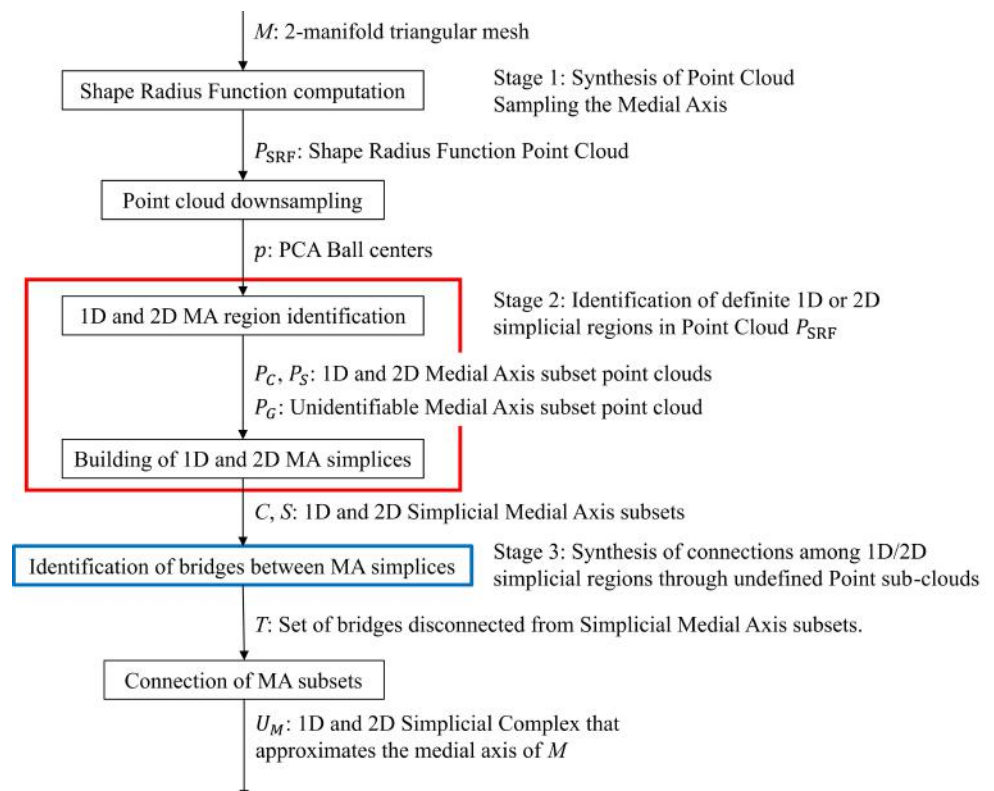
The simplices considered in this manuscript are 0-simplex (vertex), 1-simplex (straight edge), 2-simplex (triangular area), and 3-simplex (solid tetrahedron). The face of a 3-simplex is a 2-simplex (triangular area). A face of a 2-simplex is a 1-simplex (straight edge). The face of a 1-simplex is a 0-simplex (vertex). The faces of a  $k$ -simplex include all its constitutive  $k-1, k-2, \dots$  simplices. For example, the faces of a tetrahedron are all its triangles, edges, and vertices.

**Definition 2** (Simplicial Complex). A simplicial complex  $\mathcal{S}$  is a set of simplices, and their faces, which satisfy that for all  $s_k, s_w \in \mathcal{S}$ , either (1)  $s_k \cap s_w = \emptyset$ , or (2)  $s_k \cap s_w \in \mathcal{S}$ .

### 3.4. Overview of Direct Palpation Skeleton Identification

The proposed method consists of first identifying subsets of the SDF-based point cloud that exhibit 1D or 2D nature. Then, build piecewise linear curves  $C$  [41] from the 1D point cloud subsets  $P_C$ , and surface meshes  $S$  from the 2D point cloud subsets  $P_S$ . This yields a set of disconnected Skeleton regions.

To obtain a fully connected Skeleton, we use data from unclassified SDF-based point cloud subsets (neither 1D nor 2D) to help establish connections in the regions where the Skeleton is missing. The pipeline of our approach is shown in Figure 1.



**Figure 1.** Synthesis of 1D/2D simplicial complex approximation of Medial Axis via direct palpation of SRF Point Cloud. Red and blue boxes are expanded in Algorithms 1 and 2, respectively.



**Algorithm 1:** One-dimensional and 2D MA region building

---

**Input** :Shape Radius Function point cloud  $P_{\text{SRF}}$ , PCA Ball centers  $\rho$ , PCA Ball radii  $R_{\text{PCA}}$ , eigenvalue ratios for 1D and 2D point subset identification  $\alpha$ ,  $\beta$

**Output**: 1D and 2D Simplicial Skeleton subsets  $C$  and  $S$ , unidentifiable Skeleton point cloud subsets  $P_G$ .

```

1 for  $p_i \in \rho$  do
2    $P = \text{PointsInBall}(P_{\text{SRF}}, p_i, R_{\text{PCA}})$ ;
3    $Q = \text{PrincipalComponentAnalysis}(P)$ ;
4    $\lambda_1, \lambda_2, \lambda_3 = Q.\text{LargestEigenvalues}$ ;
5   Ensure that  $\lambda_1 > \lambda_2 > \lambda_3$ 
6   if  $\lambda_1/\lambda_2 > \alpha$  then
7      $P_C.\text{Append}(P)$ ;
8   else if  $\lambda_2/\lambda_3 > \beta$  then
9      $P_S.\text{Append}(P)$ ;
10  else
11     $P_G.\text{Append}(P)$ ;
12  end
13 end
14  $P_C^* = \text{Cluster}(P_C)$ ;
15  $P_S^* = \text{Cluster}(P_S)$ ;
16 for  $x_i \in P_C^*$  do
17    $c = \text{PiecewiseLinearApproximation}(x_i)$ ;
18    $C.\text{Append}(c)$ ;
19 end
20 for  $y_i \in P_S^*$  do
21    $s = \text{PointCloudTriangulation}(y_i)$ ;
22    $S.\text{Append}(s)$ ;
23 end

```

---

**Algorithm 2:** Identification of Bridges between MA subsets

---

**Input** :1D and 2D Skeleton subsets  $C$  and  $S$ , Mesh  $M$ , Unidentifiable Skeleton subset  $P_G$ .

**Output**: Collection of joining bridges  $T$ .

```

1  $P_G^* = \text{Cluster}(P_G)$ 
2 for  $x_i \in P_G^*$  do
3    $b = \text{OptimalBoundingBox}(x_i)$ ;
4    $w = b.\text{Dimensions} * s$ ;
5    $L = \text{CreateJoiningLines}(C, S, M, w)$ ;           // Refer to algorithm 3
6   for  $l_i \in L$  do
7      $p = \text{SampleLine}(l_i)$ ;
8      $P.\text{Append}(p)$ ;
9   end
10   $t = \text{PointCloudTriangulation}(P)$ ;
11   $T.\text{Append}(t)$ ;
12 end

```

---

---

**Algorithm 3:** Line Generation between Skeleton subsets through a Bounding Box
 

---

**Input** : 1D and 2D Skeleton subsets  $C$  and  $S$ , Mesh  $M$ , Bounding box  $w$ .

**Output**: Collection of joining lines  $L$ .

```

1  $G = \text{union}(C, S);$ 
2 for  $g_i \in G$  do
3    $q = \text{PointsInBox}(g_i.\text{Points}, w)$ 
4    $P.\text{append}(q)$ 
5 end
6 for  $p_i \in P$  do
7   for  $p_j \in P$  do
8     if  $\text{NotInSameSubset}(p_i, p_j)$  then
9        $l = \text{join}(p_i, p_j)$ 
10      if  $\text{NotIntersects}(l, M)$  then
11         $L.\text{append}(l)$ 
12      end
13    end
14  end
15 end
  
```

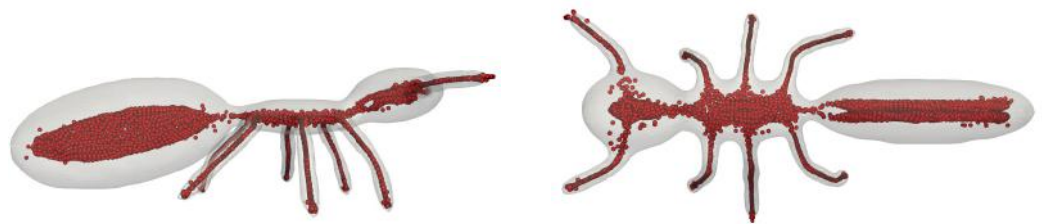
---

### 3.5. Shape Radius Function

The Shape Diameter Function is defined as a scalar function  $\text{SDF} : M \rightarrow \mathbb{R}^+$  representing the neighborhood diameter of  $M$  at each point  $p \in M$  [10]. We obtain an SDF-based point cloud following Shapira et al.'s [10] procedure. Each SDF-based point is generated as follows:

$$\mathbf{p}_i = \mathbf{z}_i - (\text{SDF}_i/2)\mathbf{n}_i \quad (1)$$

where  $\mathbf{z}_i$  and  $\text{SDF}_i$  are the incenter and the SDF value of the  $i$ th triangle, respectively.  $\mathbf{n}_i$  corresponds to the  $i$ th triangle normal. In this manuscript, to compute the SDF, instead of multi-ray tracing using multiple oscillations of the normal of each triangle [10] we use 1-ray tracing using an averaged local normal vector, since this helps to speed up the computation of the SDF while maintaining its accuracy. From now on, we refer to the SDF-based point cloud as the Shape Radius Function (SRF) point cloud  $P_{\text{SRF}}$ . An example of the SRF point cloud is shown in Figure 2.



**Figure 2.** Shape Radius Function point cloud (red)  $P_{\text{SRF}}$ . Ant dataset. Mesh  $M$  in gray.

### 3.6. Dimension Identification of Skeleton Subregions

Our approach first identifies points on  $P_{\text{SRF}}$  that will serve as centers to perform Principal Component Analysis (PCA). To obtain said centers,  $P_{\text{SRF}}$  is downsampled by applying a 3D box grid filter to it.

On each center, a PCA ball is attached, which is an open ball  $B(p, R_{\text{PCA}})$  centered in  $p$  with radius  $R_{\text{PCA}}$ . The PCA Ball radius  $R_{\text{PCA}}$  is computed as  $R_{\text{PCA}} = \varepsilon d$ , where  $d$  is the average edge length of  $M$  and  $\varepsilon$  is a safety factor, required to be input by the user.



The procedure for the construction of the 1D and 2D Skeleton subsets ( $C$  and  $S$ , respectively) from  $P_{\text{SRF}}$  is presented in Algorithm 1.

In the first 13 lines, we iterate over each PCA ball and check what points of  $P_{\text{SRF}}$  it encloses. PCA is performed with this enclosed point subset, and its dimension is established by checking the proportions of the magnitude of the three largest eigenvalues obtained by PCA. If there is no eigenvalue significantly larger or smaller than the others, the enclosed point subset is marked as neither 1D or 2D. By the end of the first 13 lines, we obtain the following  $P_{\text{SRF}}$  subsets: (1) 1D identified points  $P_C$ , (2) 2D identified points  $P_S$  and (3) unidentified points  $P_G$ , also called Gray Zones.

The values for the 1st-to-2nd and 2nd-to-3rd eigenvalue ratios  $\alpha$  and  $\beta$  are manually input. The higher these values, the more the enclosed point subset needs to strictly resemble a curve or a surface. Figure 8 shows an overview of how these parameters are used in the overall pipeline (more details in Section 3.8).

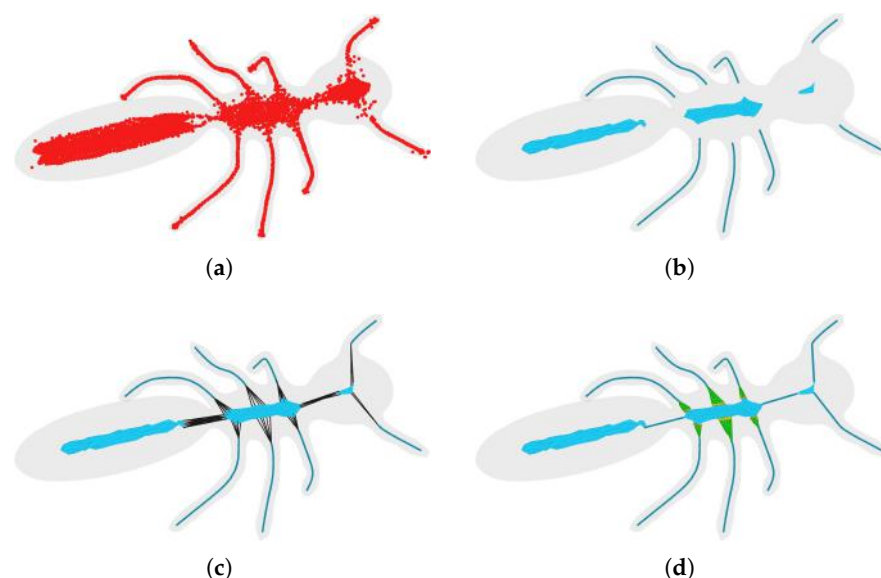
From lines 14 to 23, we cluster the  $P_C$  and  $P_S$  point clouds, in order to later identify separate curves and surfaces.

**One-dimensional clusters  $P_C^*$ .** For each  $P_C^*$  point set cluster, we fit a sequence of linear segments (PL approximation [41]). Notice that if a set of points resembles a line, PCA will associate the eigenvector (local line vector) with the largest eigenvalue of the point set auto-covariance matrix.

**Two-dimensional clusters  $P_S^*$ .** For each  $P_S^*$  point set cluster, we apply ball-based PCA identifications, which render (point, normal) pairs. We use this point and normal information to perform point cloud triangulation (Point Cloud Library PCL [42]). Notice that if a set of points resembles a surface, PCA will associate the eigenvector (local surface normal) with the smallest eigenvalue of the point set auto-covariance matrix.

### 3.7. Junction of Disconnected Subregions

After obtaining all the sets of definite 1D and 2D Skeleton subsets  $C$  and  $S$ , we still need to achieve a fully connected Skeleton, because all  $C$  and  $S$  are disjointed, as shown in Figure 3b.

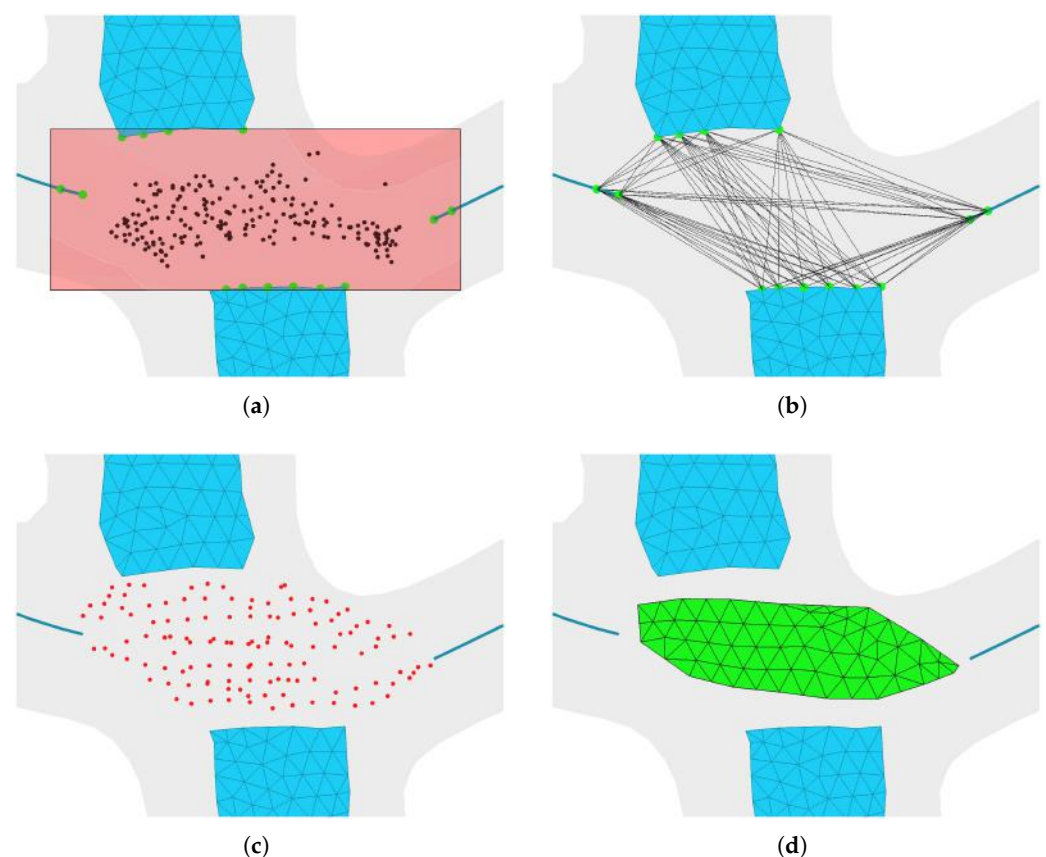


**Figure 3.** Direct palpation of Skeleton regions from subsets of  $P_{\text{SRF}}$  point cloud with clear 1D or 2D nature. Mesh  $M$  in gray. Refer to Figure 1. (a) Stage 1: Synthesis of point cloud sampling the Skeleton. (b) Stage 2: Building of definite 1D or 2D simplicial regions in point cloud  $P_{\text{SRF}}$ . (c) Stage 3: Synthesis of connections among 1D/2D simplicial regions through undefined point sub-clouds. (d) Stage 3 final result.

We use information from the Gray Zones point cloud  $P_G$  to help establish connections between Skeleton subsets. In the SRF point cloud, subsets lacking a clear 1D or 2D character correspond to neighborhoods in the Skeleton where transitions between 1D and 2D structures appear. These regions are called “gray” in the manuscript. Thus,  $P_G$  points are located in the missing Skeleton regions that bridge the various curves  $C$  and surfaces  $S$ .

Our objective is to create bridges or connections located in these Gray Zones. Bridges are surface meshes that serve as supports for connecting definite 1D/2D Skeleton subsets. The bridges do not reach the subsets that they communicate. Instead, a blend surface is later used to fill the remaining gap between the bridge and the Skeleton subsets, as described in Section Mesh Stitching.

As shown in Algorithm 2, we first cluster  $P_G$ , and for every cluster, an optimal bounding box is computed and scaled. We scale the bounding box to identify what points from all the Skeleton subsets are inside and thus involved in a joining process (Figure 4a). The disposition of these identified points will dictate the type of union to be performed.



**Figure 4.** Algorithm steps for junction of disconnected regions. The bridge (green) connects more than two Skeleton subsets (blue): (a) Unidentified point cloud  $P_G$  in black.  $P_G$  Bounding box in red, scaled by a factor  $s > 1$ . Identified points (green) inside bounding box. (b) Lines generated (black) between candidate points. (c) In red, point cloud generated by sampling the joining lines. (d) Final bridge mesh that will connect the Skeleton subsets.

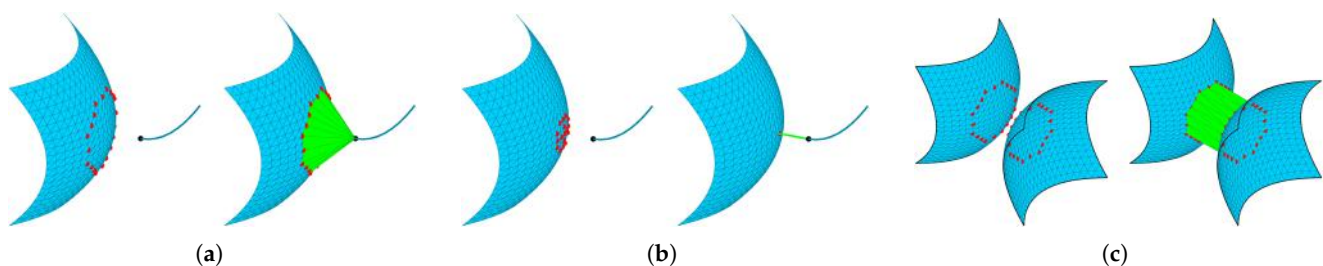
**Case 1. Boundary-based bridges among MA subsets.** In the first case,  $N$  Skeleton subsets are to be connected through their boundaries. Joining lines are established between all the points enclosed by the bounding box (see Algorithm 3), except between those belonging to the same subset, as shown in Figure 4b.

After joining the points, the joining lines are sampled, as shown in Figure 4c. We want to later triangulate these sampled points to form a surface mesh. Thus, points too close to the line endpoints are discarded to avoid intersections with the Skeleton subsets. Finally,

the resulting sampled point cloud is triangulated (Figure 4d), following the same approach as the 2D Skeleton subset triangulation in Section 3.6.

If only two Skeleton subsets are to be joined, we still create joining lines and sample them as in Case 1: Boundary-based bridges among MA subsets. However, we check if the sampled point cloud resembles a line by doing PCA with said point cloud [41]. If the point cloud was identified as 1D, we choose the joining line that is closer to the centroid of said point cloud and thus skip the process of creating a bridge mesh.

**Case 2. Interior-based connections among MA subsets.** When joining two Skeleton subsets and not all the candidate points lie on the boundaries, we do not create bridges and instead produce connections according to two main scenarios: (1) Candidate points resemble a polyline, and (2) candidate points do not resemble a polyline. The polylines do not need to be closed; however, we do not handle self-intersecting polylines. A combination of possible scenarios can be seen in Figure 5. In Section Mesh Stitching, we explain the algorithm to create the connections for said scenarios.



**Figure 5.** Possible scenario encounters when candidate points do not lie on boundaries: (a) Candidate points (red) resemble a polyline only in a Skeleton subset. The connection (green) resembles a conical shape. (b) In neither Skeleton subset, candidate points resemble a polyline. The connection is a straight line. (c) Candidate points in both Skeleton subsets resemble a polyline. The connection resembles a cylindrical shape.

### Mesh Stitching

Once a bridge is created, the next step is to fill the gap between the Skeleton subsets and the bridges in order to obtain a fully connected Skeleton. We first address the case where the bridge connects Skeleton subsets by its boundary.

We begin by identifying pairs of vertices between the Skeleton subset and the bridge that are within a specified proximity threshold ( $d_{avg}$ ). See Section 3.8. Using these matched vertices, we extract a sequence of connected points that forms a closed path spanning both the Skeleton subset and the bridge mesh.

This path contains points from both structures. To construct a blending structure in it, we apply Algorithm 4 as follows:

We first take the sequence of points from the Skeleton subset that lie on the path and iterate over them. For each pair of consecutive points, we attempt to form a triangle using those two points and an optimal third point selected from the bridge-side points along the path (lines 2–6). We then repeat the process symmetrically: iterating over the bridge-side points and selecting the third point from the Skeleton subset.

In other words, we extract a sequence of points  $P$  from the Skeleton subset and a sequence of points  $Q$  from the bridge, and pass these as inputs to Algorithm 4.

**Algorithm 4:** Blend Creation via Simple Loop Traversing

---

**Input** : Point sequences  $P = \{p_0, \dots, p_{m-1}\}$  and  $Q = \{q_0, \dots, q_{n-1}\}$  with  $p_k, q_k \in \mathbb{R}^3$

**Output**: Blend triangular mesh  $M$

```

1 for  $i = 0$  to  $m - 2$  do
2    $v_0 = P.\text{pointAtIndex}(i)$ ;
3    $v_1 = P.\text{pointAtIndex}(i + 1)$ ;
4    $p = (v_0 + v_1)/2$ ; // midpoint between  $v_0$  and  $v_1$ 
5    $v_2 = \text{getClosestPoint}(p, Q)$ ;
6    $t = \text{Triangle}(v_0, v_1, v_2)$ ;
7   if  $t \notin M$  and  $t$  is not too skinny then
8      $M.\text{AddTriangle}(t)$ ;
9   end
10 end
11 for  $j = 0$  to  $n - 2$  do
12    $v_0 = Q.\text{pointAtIndex}(j)$ ;
13    $v_1 = Q.\text{pointAtIndex}(j + 1)$ ;
14    $p = (v_0 + v_1)/2$ ;
15    $v_2 = \text{getClosestPoint}(p, P)$ ;
16    $t = \text{Triangle}(v_0, v_1, v_2)$ ;
17   if  $t \notin M$  and  $t$  is not too skinny then
18      $M.\text{AddTriangle}(t)$ ;
19   end
20 end

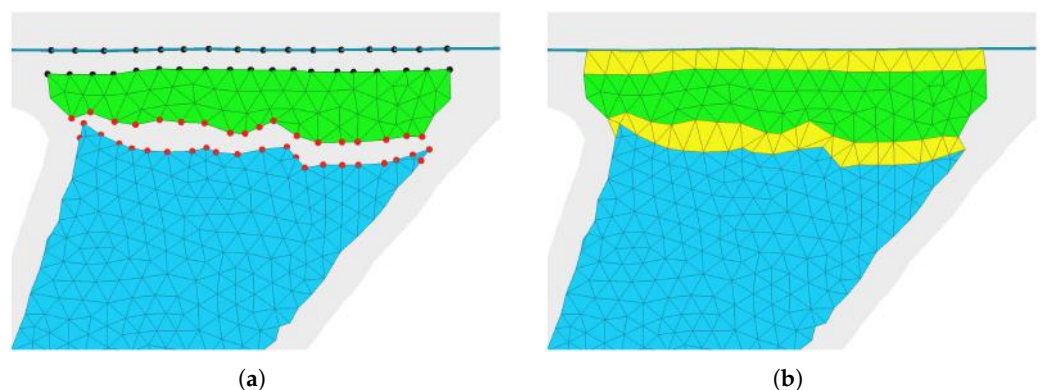
```

---

Figure 6 illustrates an example in which the algorithm is applied twice to connect a bridge with two different Skeleton subsets.

In the first step, the black point cloud is partitioned into  $P$  and  $Q$ , where  $P$  consists of a sequence of points on the 1D Skeleton subset, and  $Q$  consists of the corresponding point sequence on the bridge. The blending mesh is computed using these inputs. In the second step, the red point cloud is similarly divided into  $P$  and  $Q$ , and the algorithm is applied again.

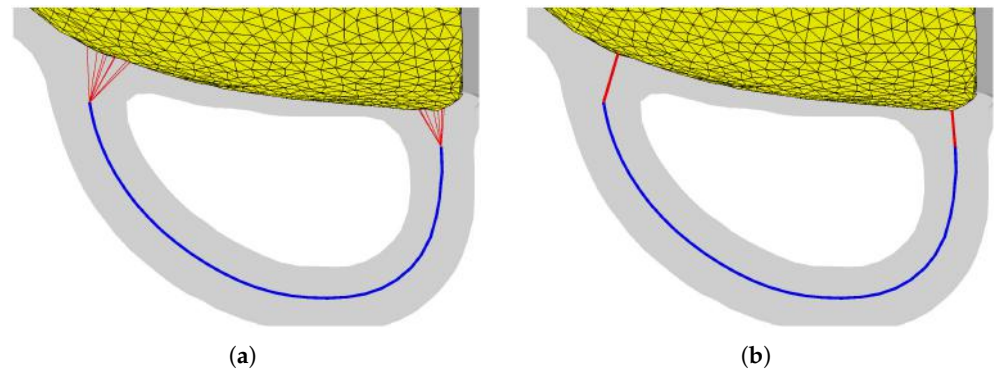
The resulting blending meshes connect the bridge to the Skeleton subsets, as shown in Figure 6b.



**Figure 6.** Bridge–Skeleton blending on a wing of the Airplane dataset: (a) In black and red: point sets used for the joining of a bridge (green) with a 1D Skeleton subset (top) and a 2D Skeleton subset (bottom), respectively. (b) Resulting blend meshes in yellow.

The bottom line of the described Algorithm 4 is the creation of a blend between two polylines. Therefore, the same algorithm can be applied to join the identified points in Case 2: Interior based connections among MA subsets, where the candidate points do not lie on the boundary but do resemble a polyline (Figure 5a,b).

In the case where candidates in neither Skeleton subset resemble a polyline (Figure 5c), we choose the joining line that is closer to the centroid of the candidate points. An example of this type of union is presented in Figure 7.



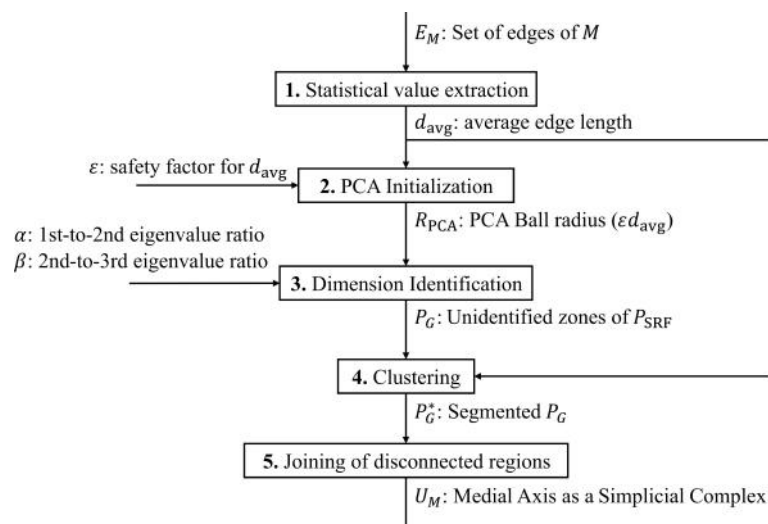
**Figure 7.** Junction of 1D and 2D Skeleton disconnected subregions. Cup dataset. Mesh  $M$  in gray. Cup handle section view: (a) Stage 3: Synthesis of connections among 1D/2D simplicial regions through undefined point sub-clouds. Refer to Figure 1. (b) Stage 3 final result.

### 3.8. Parameter Tuning

Currently, our approach requires the manual input of the parameters  $\varepsilon$ ,  $\alpha$  and  $\beta$ . To compute the PCA ball radius we opt for making the radius a function of an statistical value based on the edge lengths of the input shape  $M$ , in this case  $R_{PCA} = \varepsilon d_{avg}$ .

Since the parameters  $\alpha$  and  $\beta$  dictate the identified dimension of  $P_{SRF}$  subsets, they can be tuned to obtain Medial Axes that are fully 1D or 2D. In our experiments, we found that a value of  $\varepsilon = 3$  allows for a good dimension identification process, along with  $\alpha = 4$ ,  $\beta = 4$ .

An overview of the parameter tuning process is shown in Figure 8, and it goes as follows: we first extract an average length value  $d_{avg}$  from the set of edges  $E_M$  of the input mesh. We use this value to compute each PCA ball radius  $R_{PCA}$ . We then use  $\alpha$  and  $\beta$  to identify Curves, Surfaces, and a Gray Zone point cloud  $P_G$ . The point cloud is clustered by grouping points that are closer than  $d_{avg}$  to each other. Finally, with the clustered Gray Zones  $P_G^*$ , we carry out the joining process of the Skeleton.



**Figure 8.** Synthesis for the parameter tuning procedure in our approach for Medial Axis approximation.



### 3.9. Parameter Influence

The parameter  $\varepsilon$  directly influences the size of the PCA Balls used to classify local neighborhoods of the SRF point cloud. We compute the radii of the PCA Balls as  $R_{PCA} = \varepsilon d_{avg}$ , where  $d_{avg}$  is the average edge length of the input mesh. This means that bigger  $\varepsilon$  values will generate bigger PCA Balls.

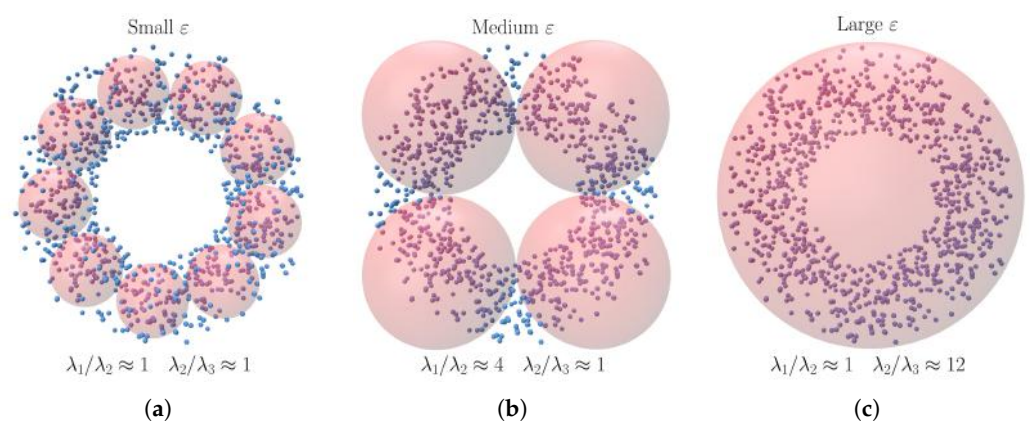
Ideally, the PCA Balls must be large enough to correctly capture the underlying dimension of each local neighborhood of the SRF Point Cloud  $P_{SRF}$ , but not too large that a single PCA Ball encloses unrelated  $P_{SRF}$  regions. In Figure 9, a  $P_{SRF}$  that samples (with noise) a circumference is used to show how the value of  $\varepsilon$  may change the underlying dimension of the enclosed point cloud subsets.

The values for  $\varepsilon$  that we provide in the following paragraphs correspond to results from our experiments, where  $d_{avg}$  was, on average, 1% of the length of the diagonal bounding box of the input mesh.

If  $\varepsilon$  is too small (e.g.,  $1 < \varepsilon < 2$ ), the enclosed point cloud subsets of the  $P_{SRF}$  will lack a clear tendency towards a curve or a surface. This case is shown in Figure 9a, where each enclosed point cloud subset resembles a solid sphere, causing eigenvalue ratios of  $\lambda_1/\lambda_2 \approx 1$  and  $\lambda_2/\lambda_3 \approx 1$ , i.e., they resemble a solid.

Moderate  $\varepsilon$  values (e.g.,  $3 < \varepsilon < 6$ ) create PCA Balls that enclose point cloud subsets with clearer curve or surface tendencies. This case is shown in Figure 9b, where each enclosed point cloud subset resembles a solid but tubular shape, causing the eigenvalue ratios  $\lambda_1/\lambda_2 \approx 4$  and  $\lambda_2/\lambda_3 \approx 1$ , i.e., each point cloud subset potentially samples a curve. This is the desired behavior in this particular example, given that  $P_{SRF}$  originally samples a 1D structure.

Too large  $\varepsilon$  values (e.g.,  $\varepsilon > 7$ ) can create PCA Balls that enclose very large portions of the  $P_{SRF}$ , which could cause loss of detail or misidentification of the underlying Skeleton structure. In Figure 9c, the resulting enclosed point subset is the entire  $P_{SRF}$ , causing the eigenvalue ratios to be  $\lambda_1/\lambda_2 \approx 1$  and  $\lambda_2/\lambda_3 \approx 12$ , i.e., the points potentially sample a surface. This is an undesired behavior, since  $P_{SRF}$  originally samples a 1D structure.



**Figure 9.** Influence of parameter  $\varepsilon$  on the eigenvalue ratios of local neighborhoods within the Shape Radius Function point cloud  $P_{SRF}$  (blue points).  $P_{SRF}$  samples (with noise) a circumference. In this figure,  $\lambda_1 > \lambda_2 > \lambda_3$  are the eigenvalues returned by every local Principal Component Analysis (PCA) on average: (a) Small  $\varepsilon$ . PCA Balls (red) yield solid-like eigenvalue ratios. (b) Moderate  $\varepsilon$ . PCA Balls yield curve-like eigenvalue ratios. (c) Too large  $\varepsilon$ . PCA Balls yield surface-like eigenvalue ratios.

Let  $\lambda_1 > \lambda_2 > \lambda_3$  be the eigenvalues obtained from applying PCA on a local neighborhood of the SRF point cloud (Algorithm 1 line 4).  $\alpha$  is the minimum ratio  $\lambda_1/\lambda_2 > \alpha$  that must be satisfied in order for the enclosed point cloud subset to be classified as belonging to a curve.  $\beta$  is the minimum ratio  $\lambda_2/\lambda_3 > \beta$  that must be satisfied in order for the enclosed point cloud subset to be classified as belonging to a surface.



Setting  $\alpha = 1$  means that all local neighborhoods of the SRF point cloud  $P_{\text{SRF}}$  will pass the check of curve tendency, since the eigenvalues from each PCA Ball will always hold that  $\lambda_1/\lambda_2 \geq 1$ .

In the same way, setting  $\beta = 1$  means that all point subsets of  $P_{\text{SRF}}$  will pass the check of surface tendency, since the eigenvalues from each PCA Ball will always hold that  $\lambda_2/\lambda_3 \geq 1$ .

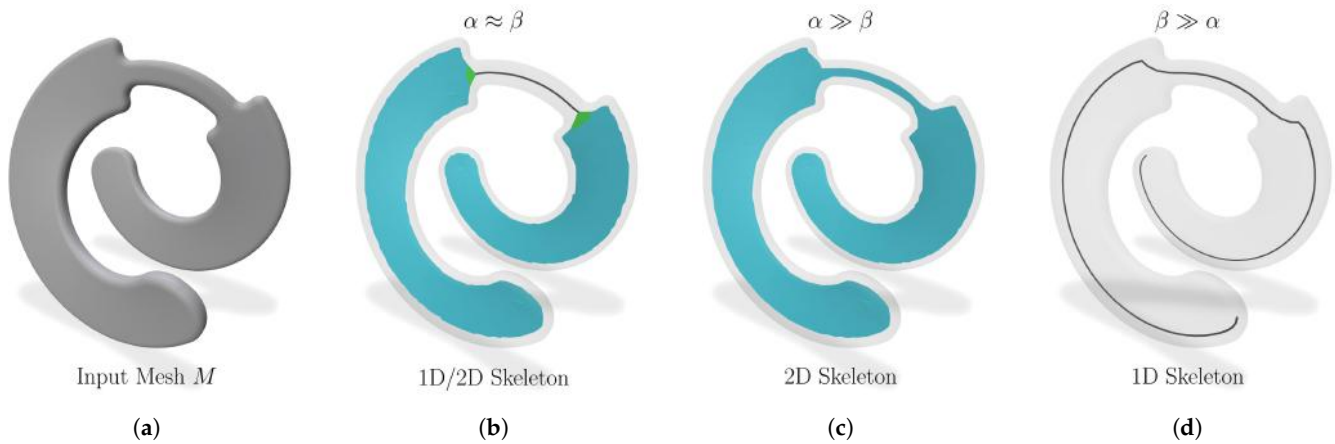
This means that, if  $\alpha = \beta = 1$ , all the local neighborhoods within the SRF point cloud will be classified exclusively as 1D or 2D, depending on the order in which the eigenvalue ratios are checked (see Algorithm 1 lines 4 to 12).

Figure 10 shows how the final dimension of the Skeleton of an input mesh  $M$  changes depending on the values set for  $\alpha$  and  $\beta$ . If  $\alpha \approx \beta$  (e.g.,  $\alpha = \beta = 3$ ) then the Skeleton will likely have a mixture of 1D and 2D elements. This behavior can be seen in Figure 10b.

If  $\alpha \gg \beta$  (e.g.,  $\alpha = 100$  and  $\beta = 1$ ) then all local neighborhoods of  $P_{\text{SRF}}$  will likely fail the curve tendency test and will pass the surface tendency test. This means that all the local neighborhoods of  $P_{\text{SRF}}$  can be used only to build 2D structures, leading to pure 2D Skeletons (Figure 10c).

If  $\beta \gg \alpha$  (e.g.,  $\alpha = 1$  and  $\beta = 100$ ) then all local neighborhoods of  $P_{\text{SRF}}$  will likely fail the surface tendency test and will pass the curve tendency test. In this setting, all the local neighborhoods of  $P_{\text{SRF}}$  will be used to build 1D structures, leading to pure 1D Skeletons (Figure 10d).

Figure 10d also shows that forcing apparent 2D Skeleton regions to be 1D can lead to artifacts in the final Skeleton, such as loss of centrality.



**Figure 10.** Influence of parameters  $\alpha$  and  $\beta$  on the dimension of the computed Skeleton of a triangular mesh  $M$ : (a) The input mesh  $M$ .  $M$  contains both planar-like and curve-like regions. (b)  $\alpha = \beta = 3$ . A mixed 1D/2D Skeleton is computed. (c)  $\alpha = 100, \beta = 1$ . A 2D Skeleton is computed. (d)  $\alpha = 1, \beta = 100$ . A 1D Skeleton is computed.

#### Scale Invariance

Let  $P = \{p_0, p_1, \dots, p_n\}$ ,  $p_i \in \mathbb{R}^3$  be an arbitrary 3D point cloud, and let  $\bar{p}$  denote its centroid. The covariance matrix  $K$  can be computed as

$$K = \frac{1}{n} \sum_{i=0}^n (p_i - \bar{p})(p_i - \bar{p})^T \quad (2)$$

If the point cloud is uniformly scaled by a factor  $s$ , the new covariance matrix  $K'$  becomes

$$K' = \frac{1}{n} \sum_{i=0}^n (sp_i - s\bar{p})(sp_i - s\bar{p})^T = s^2 \sum_{i=0}^n (p_i - \bar{p})(p_i - \bar{p})^T = s^2 K \quad (3)$$

This shows that uniformly scaling the input by a factor  $s$  results in the covariance matrix being scaled by a factor of  $s^2$ . Moreover, if  $\lambda$  is an eigenvalue of  $K$ , with corresponding eigenvector  $v$ , then

$$K'v = (s^2K)v = s^2(Kv) = s^2\lambda v \quad (4)$$

Thus, if  $\lambda$  is an eigenvalue of the original covariance matrix  $K$ , then  $s^2\lambda$  is an eigenvalue of the scaled covariance matrix  $K' = s^2K$ . Additionally, the eigenvectors remain unchanged. In other words, scaling a point cloud by a factor  $s$  scales the eigenvalues produced by PCA by  $s^2$ , while preserving the eigenvectors.

In our method, the local structure of the SRF point cloud  $P_{\text{SRF}}$  is classified using the ratios of the eigenvalues  $\lambda_1, \lambda_2, \lambda_3$  obtained from local PCA. Specifically, a region is classified as curve-like if  $\lambda_1/\lambda_2 > \alpha$ , and as surface-like if  $\lambda_2/\lambda_3 > \beta$ . When the input mesh is scaled by a factor  $s$ , the point cloud  $P_{\text{SRF}}$  is also scaled accordingly. The new eigenvalue ratios become:

$$\frac{s^2\lambda_1}{s^2\lambda_2} = \frac{\lambda_1}{\lambda_2}, \quad \frac{s^2\lambda_2}{s^2\lambda_3} = \frac{\lambda_2}{\lambda_3} \quad (5)$$

This demonstrates that the eigenvalue ratios are invariant to the scale of the input data. Consequently, the parameters  $\alpha$  and  $\beta$  are independent of the scale of the point cloud. Similarly, the parameter  $\varepsilon$  is also scale-invariant, since it appears in the expression  $R_{\text{PCA}} = \varepsilon d_{\text{avg}}$ , where  $d_{\text{avg}}$  is the average edge length of the input mesh, and therefore scales as  $s d_{\text{avg}}$ .

This scale invariance makes our method robust across datasets with different units or scale conventions (e.g., meters vs millimeters).

## 4. Results

In this section, we present the results, timing, and complexity comparisons of our algorithm with the other two relevant Medial Axis approximation approaches, which have publicly available code: QMAT [24] and Mean Curvature Skeletons (MCS) [23].

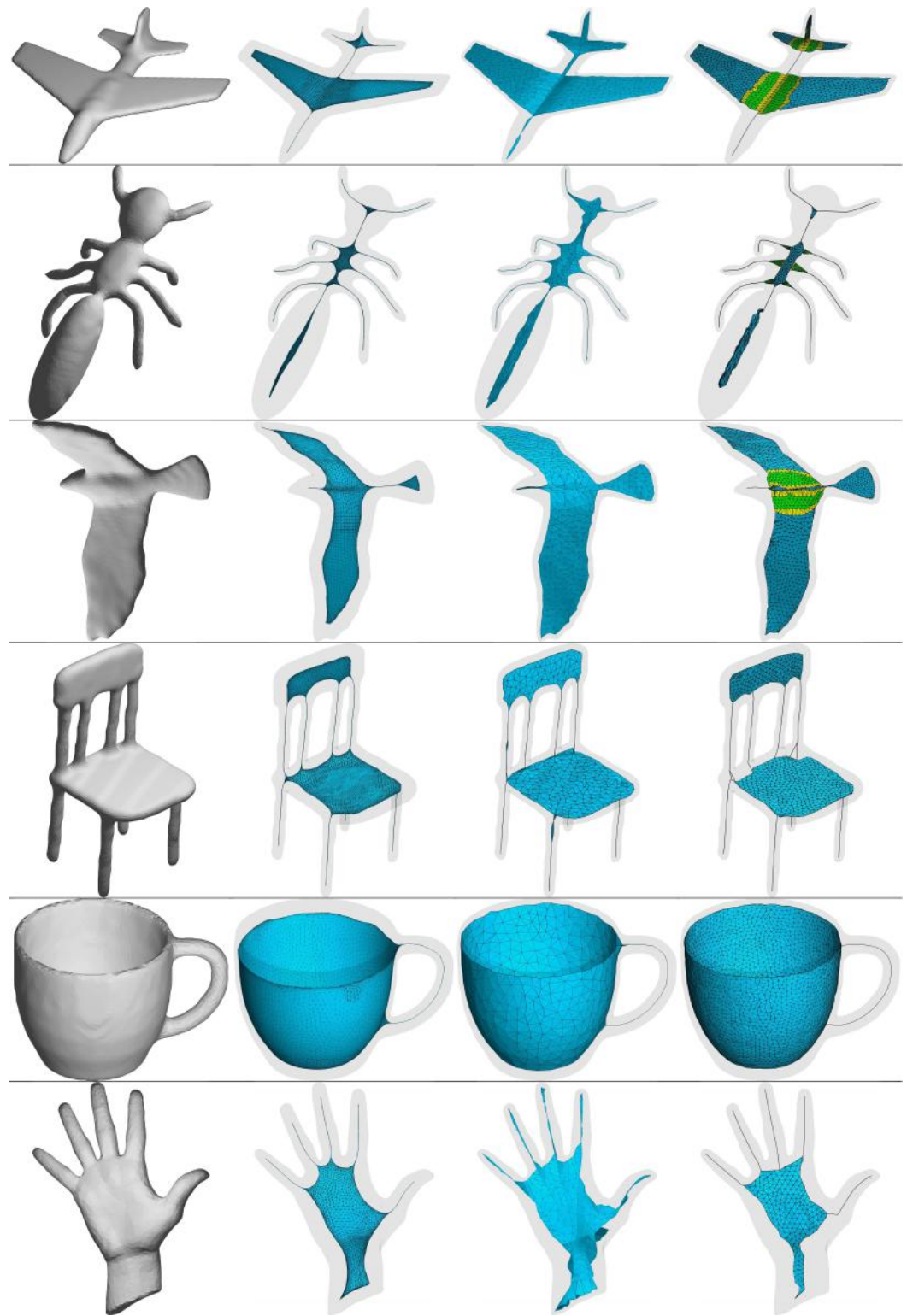
In Figure 11, we show the Skeletons obtained with QMAT, MCS, and our approach for several datasets. For our method, Green and yellow triangles indicate the bridges and connections, respectively, previously explained in Section 3.7.

For clarity, abbreviations and terms used in this manuscript appear in a glossary, which is presented at the end of this manuscript, before the references.

### 4.1. Experimental Setup

All programs were run on a Windows 10 PC with an Intel Core i5-7500 CPU 3.4 GHz and 8 GB of RAM. All the reported timings correspond to a value averaged over 5 runs.

For QMAT, the number of vertices that the final Skeleton will have must be specified. We used a value of 500 vertices, since it generally yields a good enough approximation for the Medial Axis, as can be seen in their work [24]. For MCS, we used the default parameter values that their software provided, with an iteration number of 5. For our algorithm, we used values of  $\varepsilon = 3$ ,  $\alpha = 4$  and  $\beta = 4$  for all the presented datasets.



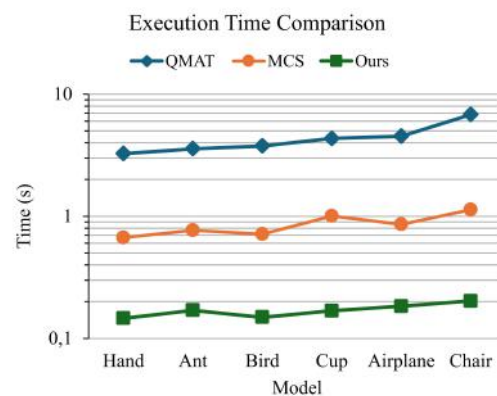
**Figure 11.** Contrast with other Medial Axis approximation strategies. Our method vs. [23,24]. Columns: 1 = Input mesh  $M$ , 2 = MCS [23], 3 = QMAT [24], 4 = our method.

#### 4.2. Performance Analysis

The time taken for the Medial Axis approximation algorithms to run is presented in Table 2 and Figure 12, where it can be seen that QMAT is the approach that took longer to run, followed by MCS and finally by our approach, which was the fastest in all the evaluated datasets.

**Table 2.** Execution Processor time comparison. Our method vs. [23,24]. First column: model name. Second and 3rd columns: number of vertices and faces of the model, respectively. Fourth column: QMAT [24]. **Displayed** times (seconds) in software QMAT<sup>TM</sup> [43]. 5th column: MCS [23]. **Displayed** times (seconds) in software STARLAB<sup>TM</sup> [44]. 6th column: Our approach. **Processor Usage** times (seconds). Compilable version of QMAT [24] was not found.

Dataset	Verts	Faces	QMAT [24]	MCS [23]	Ours
Airplane	7739	15474	4.52	0.856	0.183
Ant	5867	11730	3.57	0.769	0.17
Bird	6351	12698	3.76	0.716	0.149
Chair	10500	21008	6.81	1.128	0.202
Cup	7652	15304	4.33	1.004	0.168
Hand	5468	10932	3.26	0.667	0.146



**Figure 12.** Execution Processor times. Our method vs. QMAT [24] vs. MCS [23]. QMAT [24]: **Displayed** times in software QMAT<sup>TM</sup> [43]. MCS [23]: **Displayed** times in software STARLAB<sup>TM</sup> [44]. Our approach: **Processor Usage** times. Vertical axis is logarithmic. Compilable version of QMAT [24] was not found.

Notice that it is unclear if the displayed time of our competitors refers to wall clock time, Processor Usage time, or another kind of metric.

In addition to the timing table, we present an algorithm time complexity analysis in Table 3 based on the number of vertices of the input mesh. The complexity was calculated by using the standard costs of each different task.

**Table 3.** Time complexities of different Medial Axis approximation methods. Our method vs. [10,23,24].  $n$  is the number of points in the input model. The complexity was calculated based on the standard costs of each different task.

Shape Diameter Function Approaches					
Approach	Shape Diameter Function	Synthesis of $P_{\text{SRF}}$	Dimension Identification	Medial Axis Building	Overall
Shapira et al. [10]	$O(n^2)$	$O(n)$	—	$O(n^2)$	$O(n^2)$
Ours	$O(n^2)$	$O(n)$	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Voronoi Approaches					
Approach	Preprocessing	Voronoi Medial Axis	Parameter Initialization	Medial Axis Simplification	Overall
QMAT [24]	$O(n)$	$O(n^2)$	$O(n)$	$O(n \log n)$	$O(n^2)$
MCS [23]	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(n^3)$

In Table 3, it is shown that our algorithm maintains the overall time complexity of Shapira et al. [10] but with the advantage of being able to handle mixed 1D/2D Skeletons, unlike Shapira's which only produces 1D Skeletons. Besides, in comparison with QMAT and MCS, our approach does not increase complexity.

The Shape Diameter Function has a complexity of  $O(n^2)$ , as we cast a ray for each triangle in the mesh and check for intersections. Notice that this operation can be optimized using data structures like Bounding Volume Hierarchies (BVH) or by leveraging the GPU, both of which are efficient for ray-triangle queries and can significantly reduce the complexity. However, to ensure fair comparisons in the complexity analysis, we use the standard costs for the core steps across all methods.

The SRF point cloud  $P_{\text{SRF}}$  is computed in  $O(n)$  time, since for each triangle of the input mesh we generate a point (see Section 3.5).

In the dimension identification step of our method, we downsample  $P_{\text{SRF}}$  to obtain center points for the PCA balls. This downsampling can be performed using a voxel grid filter, with linear complexity  $O(n)$ . For each center, we identify the point subset from  $P_{\text{SRF}}$  that are contained in a sphere centered at the current point. These neighborhood queries are executed using a kd tree, which has a build complexity of  $O(n \log n)$  and a per-query complexity of  $O(\log n + k)$ , where  $k$  is the number of points returned. Each PCA computation is  $O(k)$ . Since  $k \ll n$ , and the number of centers is much less than  $n$ , the total complexity is dominated by the kd tree construction, resulting in an overall complexity of  $O(n \log n)$  for the dimension identification step.

In the Medial Axis building step, we cluster point clouds using PCL's Euclidean Cluster Extraction [42], with complexity  $O(n \log n)$ . Piecewise linear approximations involve PCA to estimate directions (for curves) and normals (for surfaces), along with  $k$ -d tree queries, also at  $O(n \log n)$ . Bounding box computation and point-in-box checks each run in  $O(n)$ . Assuming the Skeleton includes bridges and connections, the overall complexity of the building step is  $O(n \log n)$ . Consequently, the total time complexity of our method (including all steps) is  $O(n^2)$ .

Shapira et al. [10] do not perform a dimension identification step, as their method assumes that Skeletons are always 1D. Neighborhood queries are performed for all  $n$  input points in order to estimate local 1D structures. Each query may involve comparing against all other points, leading to a complexity of  $O(n^2)$ . Additional steps, such as grouping and connecting these local structures, do not introduce higher asymptotic costs. Therefore, the overall complexity of the method is  $O(n^2)$ .

In QMAT [24], the input mesh must first be sampled to enable the initial computation of the Medial Axis. Assuming uniform sampling, this step has a linear complexity of  $O(n)$ . The computation of the initial Medial Axis involves constructing a Delaunay triangulation, which can have a complexity of  $O(n^2)$  in 3D. Before simplifying the Medial Axis, Q-MAT computes both the optimal contraction target and the collapse cost for each edge, which is a linear operation  $O(n)$ . The subsequent simplification of the Medial Axis adopts the edge collapse strategy introduced by Garland and Heckbert [25], which has a complexity of  $O(n \log n)$ . The overall time complexity of the QMAT is then  $O(n^2)$ .

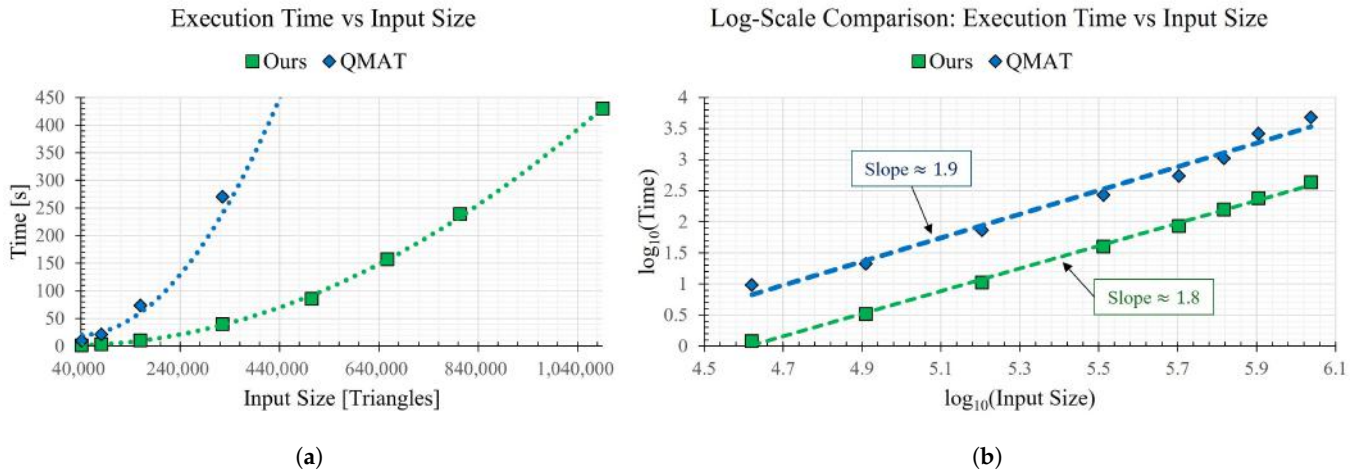
In MCS [23], the input mesh is finely remeshed and sampled to obtain high-quality Voronoi poles. This preprocessing step can take up to  $O(n^3)$ , depending on the desired quality. Computing the initial Voronoi diagram takes  $O(n^2)$ . Medial Axis simplification is performed iteratively, with a cost of  $O(n^2)$  for structure preparation (e.g., Laplacian computation, weight assignment, remeshing), followed by linear system solving at up to  $O(n^3)$ . Thus, the overall time complexity of MCS is  $O(n^3)$ .

We present a comparison of the evolution of the execution times with larger input sizes (Figure 13), between our method and QMAT. The MCS software did not admit the computation of the Skeleton for meshes with a large amount of triangles (more than 600,000), which is why it is excluded from this analysis.

Figure 13a shows that both methods (Ours and QMAT) exhibit quadratic growth in execution time as input size increases. However, our method has a smaller constant factor



and demonstrates better practical scalability. The log–log plot in Figure 13b supports this observation: both methods have slopes near 2, indicating near-quadratic time complexity. Specifically, our method has a slope of 1.8, compared with QMAT’s 1.9, and our fitted line lies below QMAT’s, reflecting lower execution times across all input sizes. The fact that the slopes deviate slightly from 2 may be due to implementation-specific optimizations, deviations from worst-case behavior, or measurement variability.



**Figure 13.** Execution time comparison between our method and QMAT [24] as a function of input size: (a) Execution time vs Input Size. Both methods exhibit quadratic growth. Our method consistently achieves lower times. (b) Log–log plot of Execution time vs Input Size. Fitted slopes of 1.8 with  $R^2 = 0.99$  (ours) and 1.9 with  $R^2 = 0.98$  (QMAT) indicate near-quadratic time complexity and better scalability for our method.

#### 4.3. Robustness to Noise

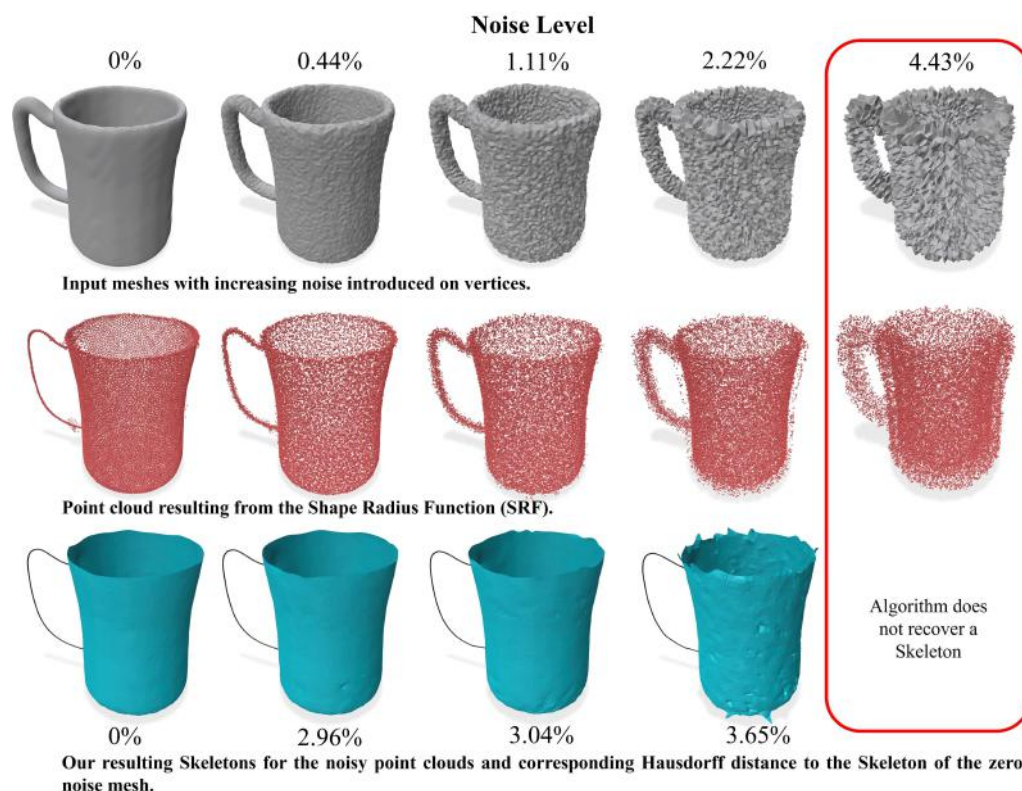
In this section, we evaluate the effect of noise of the input mesh  $M$  vertices on our computed Skeleton. Noise is created by displacing the vertices of  $M$  along their normals by a random value between 0 and  $\epsilon_{\max}$  (Noise Level). Noise Level is measured in percentage with respect to the length of the diagonal of the bounding box of  $M$ . Notice that the Noise Level is conditioned by the fact that  $M$  must preserve its manifoldness.

Figure 14 shows the effect of noise on a Cup dataset. For this analysis, we used  $\epsilon = 4$ ,  $\alpha = 6$  and  $\beta = 4$ . The Hausdorff distance metric was used to measure the error between the computed Skeletons and the Skeleton corresponding to the zero noise mesh.

Naturally, as the Noise Level increases, so does the error in the resulting Skeletons. As the SRF point cloud  $P_{\text{SRF}}$  becomes more scattered, the quality of local geometric estimates deteriorates, resulting in approximated structures that are increasingly irregular and less smooth. This behavior is expected, since it is well known that, in manifold learning, the density and uniformity of data sampling critically influence the accuracy of local neighborhood approximations.

At noise levels too high, all the local neighborhoods of  $P_{\text{SRF}}$  fail the test for curve or surface tendency, causing our algorithm to classify all local neighborhoods as Gray Zones. Since no Curve or Surface could be identified, no Skeleton can be built.



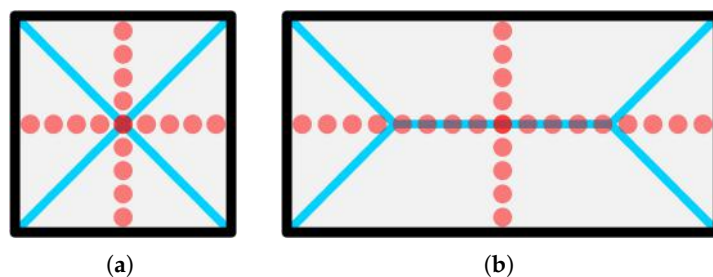


**Figure 14.** Effect of noise of the mesh vertices upon the quality of our Skeletons. Upper limit (column 5) in the noise level corresponds to mesh becoming non-manifold (by self-intersections). Percentages are computed with respect to the length of the diagonal in the input dataset bounding box.

#### 4.4. Limitations

Our method inherits the advantages and disadvantages of the Shape Diameter Function, which means that we benefit from qualities such as pose invariance and fast computation [10]. However, for some solids (e.g., symmetric prismatic shapes), the SDF is not able to generate a sampling of the Medial Axis and thus its computation can not be performed.

Figure 15 illustrates this limitation using a 2D square and rectangle. In both cases, the SRF point cloud  $P_{\text{SRF}}$  does not capture parts of the Medial Axis, especially near the corners, and includes points where the Medial Axis does not exist. In the square, only one point from  $P_{\text{SRF}}$  lies on the true Medial Axis. In the rectangle, more points fall along the central segment, but the diagonal branches near the corners remain unsampled. Several points still appear in regions unrelated to the Medial Axis.



**Figure 15.** Two-dimensional illustration of the limitation of the Shape Diameter Function (SDF)-based point cloud  $P_{\text{SRF}}$  in sampling the Medial Axis of prismatic shapes. True Medial Axis is shown in blue.  $P_{\text{SRF}}$  is shown in red: (a) Square shape. Only one point from  $P_{\text{SRF}}$  lies on the Medial Axis. (b) Rectangle shape. Only the central segment of the Medial Axis is sampled by  $P_{\text{SRF}}$ .

This behavior is not caused by errors in our method. Instead, it follows directly from the definition of the  $P_{\text{SRF}}$ : points projected inward from triangle incenters (or edge mid-points in 2D) at half the local thickness, given by the Shape Diameter Function (SDF). The limitation is therefore inherent to the use of SDF-based point clouds, not to the algorithm itself.

## 5. Conclusions and Future Work

We present a method to compute a Medial Axis approximation of a triangular mesh  $M$  by doing Direct Palpation on a point cloud derived from the Shape Diameter Function of  $M$ . Unlike existing Voronoi-based methods that rely on the computation of an initial fibrous Medial Axis and then simplify it, ours do not require performing simplification and instead build a Skeleton directly. In comparison with the SDF-based method presented by Shapira et al. [10], ours is capable of producing mixed 1D and 2D Skeletons.

Future work includes a generalization for the problem of joining disconnected Medial Axes subsets, as our current method only addresses specific scenarios that depend on the disposition of identified points on the MA subsets (Figure 5). Future work also includes reducing the amount of manually input parameters for the method so that it is completely self-tuning.

The geometric cues present in the SRF point cloud also open the door to learning-based strategies. These might assist in estimating the optimal value for parameters ( $\epsilon$ ,  $\alpha$ ,  $\beta$ ) or eliminating their need at all. Learning methods could also be explored to assist in deciding how to join disconnected Skeleton components or in predicting the complete Skeleton structure from the SRF point cloud.

To better handle high-frequency neighborhoods, our method might benefit from incorporating local surface descriptors or adaptive sampling to better capture fine details. However, prismatic shapes and sharp corners remain challenging because the SRF point cloud does not represent them well for the purpose of Skeleton extraction. This limitation might be alleviated by combining SRF data with other geometric representations suited to such cases.

**Author Contributions:** Conceptualization, D.M.-P., A.A., C.C., and O.R.-S.; methodology, D.M.-P., A.A., and O.R.-S.; software, A.F.P.-A.; formal analysis, C.C.; investigation, A.F.P.-A., D.M.-P., and O.R.-S.; writing—original draft preparation, A.F.P.-A., D.M.-P., A.A., C.C., and O.R.-S.; writing—review and editing, A.F.P.-A.; supervision, D.M.-P. and A.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding

**Data Availability Statement:** The datasets used in this paper are public and available at the Princeton Segmentation Benchmark repository at Princeton University, USA (Chen et al. [45]).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Glossary

The following terms are used in this manuscript:

PL	Piecewise linear.
MA	Medial Axis. The Medial Axis of a solid region $\mathcal{B} \subset \mathbb{R}^3$ is the set of points $(x, y, z)$ which are centers of spheres simultaneously touching the boundary of $\mathcal{B}$ at 2 or more points. The MA of $\mathcal{B} \subset \mathbb{R}^3$ is generally made up of multiple surface and curve neighborhoods connected in non-manifold configurations.
MAT	Medial Axis Transform.
Skeleton	Approximation of the Medial Axis obtained by ignoring high frequency fibrillations of the Medial Axis.

PCA	Principal Component Analysis.
PCA Ball	Open ball $B(p, r)$ centered at $p$ with radius $r$ . Used for local PCA of a subset (enclosed by $B$ ) of a point cloud.
$R_{PCA}$	Radius of the balls used for local PCA.
$\epsilon$	Safety factor for $d$ to compute $R_{PCA}$ , i. e. $R_{PCA} = \epsilon d$ .
$\lambda_1, \lambda_2, \lambda_3$	Principal component eigenvalues obtained from a PCA ball.
$\alpha$	1st-to-2nd eigenvalue ratio. Used for point subset 1D identification.
$\beta$	2nd-to-3rd eigenvalue ratio. Used for point subset 2D identification.
$\mathcal{B}$	3D body. Compact subset of $\mathbb{R}^3$ with boundary or border $M = \partial\mathcal{B}$ .
$M$	2-manifold triangular mesh $M = (\mathcal{X}, \mathcal{T})$ . Represented as a set of points $\mathcal{X} : \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}$ and a set of triangles $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$ .
SDF	Shape Diameter Function. Scalar function $f : M \rightarrow \mathbb{R}^+$ representing the “width” (neighborhood-diameter [10]) of $M$ at each point $p \in M$ .
SRF	Shape Radius Function. Scalar function corresponding to half the values of the SDF.
$P_{SRF}$	SRF Point Cloud. Obtained by projecting points $p \in M$ along their inward normal at half their SDF value.
Clusters	Subsets of point cloud $P_{SRF}$ defined by their euclidean distance proximity.
$P_C$	Subsets of $P_{SRF}$ point cloud, identified as 1-dimensional (1D).
$P_S$	Subsets of $P_{SRF}$ point cloud, identified as 2-dimensional (2D).
$P_G$	Subsets of $P_{SRF}$ point cloud, identified as neither 1D or 2D, also called <i>Gray Zones</i> .
$k$ -simplex	A 0-simplex, 1-simplex, 2-simplex, 3-simplex is a vertex, straight segment, triangular area and solid tetrahedron, respectively.
Face	A face of a 2-simplex is a 1-simplex (straight edge). A face of a 1-simplex is a 0-simplex (vertex). The faces of a $k$ -simplex include all its constitutive $k - 1, k - 2, \dots$ simplexes.
$\mathcal{S}$	Simplicial Complex. Set of simplexes, and their faces, which satisfy that for all $s_k, s_w \in \mathcal{S}$ , either (1) $s_k \cap s_w = \emptyset$ , or (2) $s_k \cap s_w \in \mathcal{S}$ .
$\mathcal{C}$	PL curves. One-dimensional simplicial subsets of the Skeleton.
$\mathcal{S}$	PL surfaces. Two-dimensional simplicial subsets of the Skeleton.
$P_C^*, P_S^*, P_G^*$	Segmented versions of point clouds $P_C, P_S$ and $P_G$ , respectively.
Bridge	2D triangular mesh that quasi communicates 1D and 2D Skeleton subsets. In order to avoid self-intersections, bridges do not reach the MA subsets that they communicate with. Instead, a blended surface is used to fill the remaining gap between the bridge and the MA subsets.
$\mathcal{T}$	Set of bridges. Bridges $\mathcal{T}$ cover point cloud neighborhoods with neither 1D nor 2D statistical character.
BL	Blend (triangle) neighborhoods which smoothly join bridges $\mathcal{T}$ with $\mathcal{C}$ and $\mathcal{S}$ . PL surfaces.
CN	Connections. One-dimensional/2D simplex sets which join interior and/or border neighborhoods of 1D/2D Skeleton subsets.
$U_M$	Final Result of the implemented algorithm. One-dimensional/2D Simplicial Complex that approximates the Medial Axis of $M$ . $U_M = \mathcal{C} \cup \mathcal{S} \cup \mathcal{T} \cup \text{BL} \cup \text{CN}$ .

## References

1. Saeed, H.; Skalski, A. Vessel Geometry Estimation for Patients with Peripheral Artery Disease. *Sensors* **2024**, *24*, 6441. <https://doi.org/10.3390/s24196441>.
2. Amenta, N.; Choi, S.; Kolluri, R.K. The power crust. In Proceedings of the SMA '01, Ann Arbor, MI, USA, 4–8 June 2001; pp. 249–266. <https://doi.org/10.1145/376957.376986>.
3. Wu, S.; Huang, H.; Gong, M.; Zwicker, M.; Cohen-Or, D. Deep points consolidation. *ACM Trans. Graph.* **2015**, *34*, 1–13. <https://doi.org/10.1145/2816795.2818073>.
4. Lin, C.; Liu, L.; Li, C.; Kobbelt, L.; Wang, B.; Xin, S.; Wang, W. SEG-MAT: 3D Shape Segmentation Using Medial Axis Transform. *IEEE Trans. Vis. Comput. Graph.* **2022**, *28*, 2430–2444. <https://doi.org/10.1109/TVCG.2020.3032566>.
5. Dou, Z.; Xin, S.; Xu, R.; Xu, J.; Zhou, Y.; Chen, S.; Wang, W.; Zhao, X.; Tu, C. Top-Down Shape Abstraction Based on Greedy Pole Selection. *IEEE Trans. Vis. Comput. Graph.* **2021**, *27*, 3982–3993. <https://doi.org/10.1109/TVCG.2020.2995495>.

6. Boujou, M.; Iguernaissi, R.; Nicod, L.; Merad, D.; Dubuisson, S. In-Depth Analysis of GAF-Net: Comparative Fusion Approaches in Video-Based Person Re-Identification. *Algorithms* **2024**, *17*, 352. <https://doi.org/10.3390/a17080352>.
7. Hu, J.; Wang, B.; Qian, L.; Pan, Y.; Guo, X.; Liu, L.; Wang, W. MAT-Net: Medial Axis Transform Network for 3D Object Recognition. In Proceedings of the IJCAI '19, Macao, China, 10–16 August 2019; pp. 774–781. <https://doi.org/10.24963/ijcai.2019/109>.
8. Yang, B.; Yao, J.; Guo, X. DMAT: Deformable Medial Axis Transform for Animated Mesh Approximation. *Comput. Graph. Forum* **2018**, *37*, 301–311. <https://doi.org/10.1111/cgf.13569>.
9. Blum, H. A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*; Wathen-Dunn, W., Ed.; MIT Press: Cambridge, MA, USA, 1967; pp. 362–381.
10. Shapira, L.; Shamir, A.; Cohen-Or, D. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.* **2008**, *24*, 249–259. <https://doi.org/10.1007/s00371-007-0197-5>.
11. Tagliasacchi, A.; Delame, T.; Spagnuolo, M.; Amenta, N.; Telea, A. 3D Skeletons: A State-of-the-Art Report. *Comput. Graph. Forum* **2016**, *35*, 573–597. <https://doi.org/10.1111/cgf.12865>.
12. Au, O.K.C.; Tai, C.L.; Chu, H.K.; Cohen-Or, D.; Lee, T.Y. Skeleton extraction by mesh contraction. *ACM Trans. Graph.* **2008**, *27*, 1–10. <https://doi.org/10.1145/1360612.1360643>.
13. Cao, J.; Tagliasacchi, A.; Olson, M.; Zhang, H.; Su, Z. Point Cloud Skeletons via Laplacian Based Contraction. In Proceedings of the SMI '10, Aix-en-Provence, France, 21–23 June 2010; pp. 187–197. <https://doi.org/10.1109/SMI.2010.25>.
14. Thiery, J.M.; Guy, E.; Boubekur, T. Sphere-Meshes: Shape approximation using spherical quadric error metrics. *ACM Trans. Graph.* **2013**, *32*. <https://doi.org/10.1145/2508363.2508384>.
15. Palágyi, K.; Kuba, A. A Parallel 3D 12-Subiteration Thinning Algorithm. *Graph. Model. Image Process.* **1999**, *61*, 199–221. <https://doi.org/10.1006/gmip.1999.0498>.
16. Németh, G.; Kardos, P.; Palágyi, K. Topology Preserving 3D Thinning Algorithms Using Four and Eight Subfields. In Proceedings of the Image Analysis and Recognition, Varzim, Portugal, 21–23 June 2010; pp. 316–325. [https://doi.org/10.1007/978-3-642-13772-3\\_32](https://doi.org/10.1007/978-3-642-13772-3_32).
17. Moreno-Avendano, S.; Mejia-Parra, D.; Ruiz-Salguero, O. Triangle mesh skeletonization using non-deterministic voxel thinning and graph spectrum segmentation. *MATEC Web Conf.* **2021**, *336*, 02030. <https://doi.org/10.1051/mateconf/202133602030>.
18. Németh, G.; Kardos, P.; Palágyi, K. Thinning combined with iteration-by-iteration smoothing for 3D binary images. *Graph. Models* **2011**, *73*, 335–345. <https://doi.org/10.1016/j.gmod.2011.02.001>.
19. Sun, F.; Choi, Y.K.; Yu, Y.; Wang, W. Medial Meshes – A Compact and Accurate Representation of Medial Axis Transform. *IEEE Trans. Vis. Comput. Graph.* **2016**, *22*, 1278–1290. <https://doi.org/10.1109/TVCG.2015.2448080>.
20. Chazal, F.; Lieutier, A. The ‘ $\lambda$ -medial axis’. *Graph. Models* **2005**, *67*, 304–331. <https://doi.org/10.1016/j.gmod.2005.01.002>.
21. Miklos, B.; Giesen, J.; Pauly, M. Discrete scale axis representations for 3D geometry. In Proceedings of the ACM SIGGRAPH 2010 Papers, Los Angeles, CA, USA, 26–30 July 2010; pp. 1–10. <https://doi.org/10.1145/1833349.1778838>.
22. Faraj, N.; Thiery, J.M.; Boubekur, T. Progressive medial axis filtration. In Proceedings of the SIGGRAPH Asia 2013 Technical Briefs, Hong Kong, China, 19–22 November 2013; pp. 1–4. <https://doi.org/10.1145/2542355.2542359>.
23. Tagliasacchi, A.; Alhashim, I.; Olson, M.; Zhang, H. Mean Curvature Skeletons. *Comput. Graph. Forum* **2012**, *31*, 1735–1744. <https://doi.org/10.1111/j.1467-8659.2012.03178.x>.
24. Li, P.; Wang, B.; Sun, F.; Guo, X.; Zhang, C.; Wang, W. Q-MAT: Computing Medial Axis Transform By Quadratic Error Minimization. *ACM Trans. Graph.* **2016**, *35*, 1–16. <https://doi.org/10.1145/2753755>.
25. Garland, M.; Heckbert, P.S. Surface simplification using quadric error metrics. In Proceedings of the SIGGRAPH '97, Los Angeles, CA, USA, 3–8 August 1997; pp. 209–216. <https://doi.org/10.1145/258734.258849>.
26. Pan, Y.; Wang, B.; Guo, X.; Zeng, H.; Ma, Y.; Wang, W. Q-MAT+: An error-controllable and feature-sensitive simplification algorithm for medial axis transform. *Comput. Aided Geom. Des.* **2019**, *71*, 16–29. <https://doi.org/10.1016/j.cagd.2019.04.007>.
27. Dou, Z.; Lin, C.; Xu, R.; Yang, L.; Xin, S.; Komura, T.; Wang, W. Coverage Axis: Inner Point Selection for 3D Shape Skeletonization. *Comput. Graph. Forum* **2022**, *41*, 419–432. <https://doi.org/10.1111/cgf.14484>.
28. Wang, Z.; Dou, Z.; Xu, R.; Lin, C.; Liu, Y.; Long, X.; Xin, S.; Komura, T.; Yuan, X.; Wang, W. Coverage Axis++: Efficient Inner Point Selection for 3D Shape Skeletonization. *Comput. Graph. Forum* **2024**, *43*, e15143. <https://doi.org/10.1111/cgf.15143>.
29. Rebain, D.; Angles, B.; Valentin, J.; Vining, N.; Peethambaran, J.; Izadi, S.; Tagliasacchi, A. LSMAT Least Squares Medial Axis Transform. *Comput. Graph. Forum* **2019**, *38*, 5–18. <https://doi.org/10.1111/cgf.13599>.
30. Lee, Y.; Baek, J.; Kim, Y.M.; Park, F.C. IMAT: The Iterative Medial Axis Transform. *Comput. Graph. Forum* **2021**, *40*, 162–181. <https://doi.org/10.1111/cgf.14266>.
31. Ma, J.; Bae, S.W.; Choi, S. 3D medial axis point approximation using nearest neighbors and the normal field. *Vis. Comput.* **2012**, *28*, 7–19. <https://doi.org/10.1007/s00371-011-0594-7>.
32. Jalba, A.C.; Kustra, J.; Telea, A.C. Surface and Curve Skeletonization of Large 3D Models on the GPU. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1495–1508. <https://doi.org/10.1109/TPAMI.2012.212>.



33. Ge, M.; Yao, J.; Yang, B.; Wang, N.; Chen, Z.; Guo, X. Point2MM: Learning medial mesh from point clouds. *Comput. Graph.* **2023**, *115*, 511–521. <https://doi.org/10.1016/j.cag.2023.07.020>.
34. Clémot, M.; Digne, J. Neural skeleton: Implicit neural representation away from the surface. *Comput. Graph.* **2023**, *114*, 368–378. <https://doi.org/10.1016/j.cag.2023.06.012>.
35. Lin, C.; Li, C.; Liu, Y.; Chen, N.; Choi, Y.K.; Wang, W. Point2Skeleton: Learning Skeletal Representations from Point Clouds. In Proceedings of the CVPR '21, Nashville, TN, USA, 20–25 June 2021; pp. 4275–4284. <https://doi.org/10.1109/CVPR46437.2021.00426>.
36. Whittaker, E.T. On the Functions which are represented by the Expansions of the Interpolation Theory. *Proc. R. Soc. Edinb.* **1915**, *35*, 181–194. <https://doi.org/10.1017/S0370164600017806>.
37. Spivak, M.D. *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*; Westview Press: Boulder, CO, USA, 1971. ISBN-13: 978-0-8053-9021-6, ISBN: 0-8053-9021-9.
38. Ge, Y.; Li, Z.; Tang, H.; Chen, Q.; Wen, Z. Efficient rock joint detection from large-scale 3D point clouds using vectorization and parallel computing approaches. *Geosci. Front.* **2025**, *16*, 102085. <https://doi.org/10.1016/j.gsf.2025.102085>.
39. Chen, Q.; Ge, Y.; Tang, H. Rock discontinuities characterization from large-scale point clouds using a point-based deep learning method. *Eng. Geol.* **2024**, *337*, 107585. <https://doi.org/10.1016/j.enggeo.2024.107585>.
40. Croom, F.H., Geometric Complexes and Polyhedra. In *Basic Concepts of Algebraic Topology*; Springer: New York, NY, USA, 1978; pp. 1–15. [https://doi.org/10.1007/978-1-4684-9475-4\\_1](https://doi.org/10.1007/978-1-4684-9475-4_1).
41. Ruiz, O.; Vanegas, C.; Cadavid, C. Principal component and Voronoi skeleton alternatives for curve reconstruction from noisy point sets. *J. Eng. Des.* **2007**, *18*, 437–457. <https://doi.org/10.1080/09544820701403771>.
42. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). In Proceedings of the ICRA '11, Shanghai, China, 9–13 May 2011; pp. 1–4. <https://doi.org/10.1109/ICRA.2011.5980567>.
43. Li, P.; Wang, B.; Sun, F.; Guo, X.; Zhang, C.; Wang, W. Q-MAT Source Code. 2016. Available online: <https://binwangthss.github.io/qmat/qmat.html> (accessed on 23 December 2024).
44. Tagliasacchi, A.; Alhashim, I.; Olson, M.; Zhang, H. Mean Curvature Skeletons Source Code. 2021. Available online: <https://github.com/taiya/starlab-mcfskel> (accessed on 23 December 2024).
45. Chen, X.; Golovinskiy, A.; Funkhouser, T. A benchmark for 3D mesh segmentation. *ACM Trans. Graph.* **2009**, *28*, 1–12. <https://doi.org/10.1145/1531326.1531379>.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.