DOI: 10.1111/cgf.14652
Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Out-of-core Extraction of Curve Skeletons for Large Volumetric Models

Yiyao Chu[1,2] and Wencheng Wang[†1,2]

[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China

Abstract

Existing methods for skeleton extraction have limitations in terms of the amount of memory space available, as the model must be allocated to the random access memory. This challenges the treatment of out-of-core models. Although applying out-of-core simplification methods to the model can fit in memory, this would induce distortion of the model surface, and so causing the skeleton to be off-centered or changing the topological structure. In this paper, we propose an efficient out-of-core method for extracting skeletons from large volumetric models. The method takes a volumetric model as input and first computes an out-of-core distance transform. With the distance transform, we generate a medial mesh to capture the prominent features for skeleton extraction, which significantly reduces the data size and facilitates the process of large models. At last, we contract the medial mesh in an out-of-core fashion to generate the skeleton. Experimental results show that our method can efficiently extract high-quality curve skeletons from large volumetric models with small memory usage.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;*** *Shape analysis;*

1. Introduction

Skeletons are shape descriptors that concisely capture the shapes' topology and geometry. 3D shapes admit two types of skeletons: surface skeletons and curve skeletons. *Surface skeletons*, or medial surfaces, are surfaces that consist of the centers of all maximally inscribed balls inside the shapes. *Curve skeletons* are 1D curves that are locally centered inside the shapes. The curve skeletons have been extensively used in various applications, such as 3D vessel tracking [AB02], virtual colonoscopies and virtual endoscopies, computer animation, action or gesture recognition and retargeting [TTL*18, ALL*20], 3D shape matching, retrieval, and reconstruction [WCLB09, LH11, JCZ19].

With the advance in data acquisition techniques, such as laser range and CT/MRI scanners, we have witnessed a boom in high-resolution 3D data in recent years. For example, the human body models in the Visible Human Project [Ack98] can consist of over 10 billion voxels, and the range scans of Michelangelo's sculptures [LPC*00] can contain up to two billion triangles. The growing data size poses great challenges in memory and efficiency for existing methods to extract curve skeletons. Since most existing methods extract the curve skeleton by contracting the shape [PK98, ATC*08, ClO*10, LHW*13, QHL*20] or contracting the medial surface for results with better centrality [LB07, CB16, LW17, LW18, JST16, YSC*16, LWC20, CWL22], the model or the medial surface must be allocated to the random access memory (RAM). This means that the size of models that existing methods can handle is limited by the amount of memory space available. Hence, applying existing methods to the out-of-core models often requires simplifying the model to fit the model in memory. Nevertheless, out-of-core simplification might induce distortion of the surface, which may cause the skeleton to be off-centered or change the topology. Another possible solution is to decompose the model into in-core blocks to fit in memory. However, since the contraction is global, each time a block is loaded only a small part of the block can be contracted, and so each block would be repeatedly loaded until all the contents in the block have been contracted, resulting in extremely inefficient data access. Besides, existing methods also suffer from insufficient speed for large models even if the memory space is enough, as the computation cost grows by geometric progression as the grid resolution of the volumetric model increases.

In this paper, we address the challenges posed by large models based on two observations. First of all, inspired by the work [CWL22] which generates a medial mesh to capture prominent features from the medial surface for high-quality skeleton extraction, we observe that capturing the prominent features considerably reduces the data size and facilitates the process of large models. Besides, since the medial surface is also defined as the singularities of the distance field, we find that the prominent features can be cap-

† Corresponding author: whn@ios.ac.cn (Wencheng Wang)

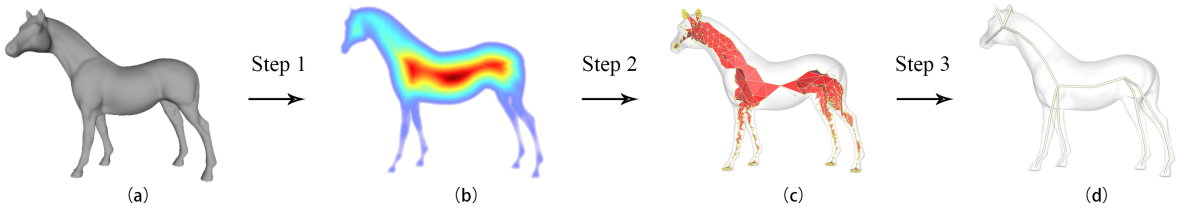


Figure 1: *The pipeline of the proposed out-of-core curve skeleton extraction method. Firstly, an out-of-core distance field (b) is constructed from the input volumetric shape (a). Then, a medial mesh (c) is extracted from the distance field to capture the prominent features and reduce the data size. Finally, the medial mesh is contracted in an out-of-core fashion to generate the curve skeleton (d).*

tured from the distance field instead of the medial surface, which saves the trouble of extracting the medial surface from the distance field and so significantly promote the efficiency for handling large models. Hence, we propose to generate a medial mesh from the distance field to efficiently reduce the data size and capture the prominent features for skeleton extraction. Secondly, we observe that existing works always contract the boundary of the shape or medial surface with some priority guidance to achieve skeletons with better centrality. Therefore, we can evolve the contraction process by keeping in-core storage of the most prior contraction boundary, which also synchronizes the data access with the contraction process and avoids repeatedly loading the same data. In this way, we can realize an efficient out-of-core contraction of the medial mesh to extract the curve skeleton. Accordingly, we develop a novel out-of-core curve skeleton extraction method (abbreviated as *OoCCS*) for large 3D models.

Our method takes a 3D volumetric shape, denoted as $\mathbf{V}$, as input. As for shapes of other representations, they can be voxelized to feed in our method, where the grid resolutions should be high enough for the volumes to accurately capture the shapes' topology and geometry. As shown in Figure 1, the pipeline of our method consists of three steps. The first step is to compute the distance field inside the shape in an out-of-core fashion similar to existing work [MTFS07]. The second step is to generate a medial mesh $\mathcal{M}$ from the distance field in an out-of-core fashion to reduce the data size and capture the prominent features for high-quality skeleton extraction. The third step is to contract the medial mesh $\mathcal{M}$ in an out-of-core fashion to generate the curve skeleton. In the first step which constructs the distance field by propagation between local blocks, we also develop a novel and interpretable ordering measure that improves the performance by reducing the number of block propagations. In the last two steps, we also develop novel out-of-core measures to generate and contract a medial mesh.

In sum, to solve the challenges posed by the large data size, we present an out-of-core method to extract curve skeletons from large 3D models. Experimental results show that our method can efficiently extract high-quality curve skeletons from large 3D models with small memory usage, and the resulted curve skeletons are of high quality, comparable with those from in-core models with existing methods. We have contributions as follows:

- We propose a novel ordering measure for the out-of-core distance transform framework [MTFS07], which can improve efficiency by reducing the number of block propagations.
- We define the set cover of the distance field to extend the method used in [CWL22] to capture prominent features for skeleton extraction and save the trouble of extracting the medial surface from the distance field.
- We propose an out-of-core contraction framework for the out-of-core skeleton extraction, where efficient data access can be realized.

2. Related Work

In this section, we briefly review the related works on medial surface extraction, curve skeleton extraction, and distance field computation.

2.1. Medial surface extraction

The medial surface $\mathcal{M}$ of a 3D shape, also called the *medial axis*, can be defined as the set of the centers of all maximally inscribed spheres. Although the definition of the medial surface is simple, methods for extracting exact medial surfaces are available only for rather simple shapes, such as polyhedra [Man04] and unions of balls [AK01a]. For ordinary shapes, the medial surface can only be approximately extracted by existing methods. According to the representation of the medial surface, the existing methods can be divided into three categories: Point-based methods, Mesh-based methods, and Voxel-based methods.

Point-based methods [MBC12, RAV*19] take samples on the shape boundary and generate the medial surface by computing the centers of inscribed balls that are tangent to the shape boundary at the sample points. However, their medial surfaces are represented as point clouds, which is not effective for representing the topological structure of the shape and so is limited for further applications.

Mesh-based methods extract the medial surface by computing the Voronoi diagram of points sampled on the shape boundary [AM96, AK01b, GRS06, JKT13]. These methods provide theoretical guarantees of geometric accuracy and topology preservation when the boundary samples are dense enough. Yan et al. [YLJ18] propose to use voxel boundary vertices as samples to reduce the requirement for the sample density with reasonable approximation. Nevertheless, the out-of-core implementation of these methods remains an open challenge.

Voxel-based methods generally compute a distance field of the

volumetric shape and extract the medial surface by locating the singularities of the distance field with some local significance measure [SBTZ02] or contracting the shape under the guidance of the distance field [Pud98, LB07] or some centrality measure [JST16]. Unfortunately, these methods suffer from expensive memory and computational costs for large volumetric models. Michikawa et al. [MNS09] try to reduce the memory cost by proposing a simplified geodesic measure for extracting medial surfaces from out-of-core distance transforms. However, the expensive computational cost for large models remains a great challenge.

2.2. Skeleton extraction

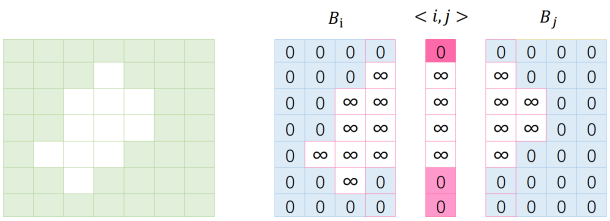
For 3D curve skeleton extraction, most methods extract the curve skeleton by contracting the 3D shape or its medial surface. Some methods directly handle the 3D shape by iterative thinning [PK98] or contraction with different local operators [ATC*08, TAOH12, THCO09, ClO*10, LHW*13]. However, since the speeds of contraction depend on the local boundary geometry, the speeds for different parts of a shape differ. Therefore, these methods often generate off-centered skeletons.

Hence, for generating centered curve skeletons, many methods attempt to contract the medial surface of a 3D shape to leverage the centrality of the medial surface. They extract curve skeletons by removing the boundary of medial surfaces layer by layer [LB07, ABS11, CB16], using local contraction operators with carefully controlled directions and speeds [TJ12, LW17, LW18], or performing significance measurements on the medial surface [DS06, RvWT08, JST16, YSC*16]. Unfortunately, as these methods adopt an outside-in evolution scheme, they are unavoidably sensitive to boundary perturbations and generally require tedious manual parameter adjustment to prune noisy branches. Besides, the contraction scheme of these methods may result in terminal-contracted skeletons, which hinders the completeness in representing the shape. As an alternative, Li et al. [LWC20] adopt an inside-out evolution scheme to measure centrality on the medial surface, by which the influences of boundary perturbations can be suppressed. Nevertheless, its mechanisms for detecting ridge curves of the centrality field to extract skeletons may fail to capture curvature features and terminal features. Recently, Chu et al. [CWL22] propose to enhance the capture of prominent features for skeleton extraction by computing a set cover of the medial surface. The set cover they used can not only eliminate the influence of boundary perturbations but also reduces redundant prominent features, which provides a solid base for fast extraction of high-quality skeletons. But they still follow a contraction framework to generate curve skeletons from the medial mesh they extracted for capturing prominent features. In sum, though existing methods have achieved great improvements in the quality of the skeleton, the size of the model they can handle is still limited by the amount of memory space available as the shape or the medial surface must be loaded completely into memory.

There are also some non-contraction methods for curve skeleton extraction, such as generating curve skeletons from rendered images of 3D shapes from many views [LGS12, LS13, THP*19], or using deep learning to extract skeletons [XZKS19, XZK*20]. However, the former ones either generate spurious branches or fail to handle complex shapes with severe self-occlusion. The latter ones are limited to simple articulated shapes. Besides, the deep learning methods have strict limitations in terms of memory space available as the model generally needs to be loaded to GPU memory.

2.3. Distance field construction

Distance field construction is a widely studied topic in the field of computer graphics and image processing. Some surveys can be found in [JBS06, FCTB08]. Here, we mainly focus on volumetric distance fields. Early methods try to approximate the distance field with different definitions of distance [Bor96, RP66, BMA*98] so that the distance field can be sequentially computed in a time linear to the number of voxels. Maurer et al. [MQR03] first compute the exact Euclidean Distance Transform (EDT) for a binary image of arbitrary dimension with a time linear to the number of voxels using dimensionality reduction and partial Voronoi diagram construction. Researchers have also proposed many methods for parallel acceleration with GPU [KMH11, CTMT10, YXM*15]. However, these methods require that all voxels are stored in the RAM, meaning that these methods are limited by the amount of memory available. To solve this problem, Michikawa et al. [MTFS07] proposed a method (OOCDT) for out-of-core distance field construction. This method decomposes an input model into sub-blocks and computes the distance transform for each block. Inconsistency in distances between blocks can be resolved by inter-block propagation. This method provides a framework to compute large distance fields with small memory usage. Thus, we adopt a similar framework as OOCDT to compute a distance field in our first step, but propose a novel ordering measure for the local block distance transform to improve the efficiency.



(a) Input shape volume (b) Neighboring blocks and the interface

Figure 2: *The Data structure for out-of-core computation of the distance field. (a) Input shape volume* **V**. *The voxels occupied by the shape is colored in white. (b) The two blocks decomposed from the input volume and their shared unit thick interface are stored as three separate files. The initialization of the distance field set the foreground voxels to* ∞ *and the background voxels to* 0.

3. Out-of-core Distance Field Construction

In this section, we first briefly introduce the out-of-core framework for distance field computation [MTFS07] and the efficiency problem in the process ordering of blocks. Then, we discuss our improvement in efficiency by proposing a novel priority measure called *boundary index* for block ordering. At last, we give the algorithm for the out-of-core distance transform using the proposed priority measure.

3.1. Out-of-core framework for distance field computation

The framework for computing out-of-core distance transform [MTFS07] first decomposes the input volumetric shape V into $l \times m \times n$ blocks. Neighboring blocks B_i and B_j share an interface $< i, j >$ with unit thickness, as shown by the 2D example in Figure 2. Then, an inside-block distance transform (InsDT) is applied to each block to compute the local distance field inside the block. Afterward, inconsistency in distances between blocks is resolved by inter-block propagation (IntBP), and the distance transforms of the propagated blocks are updated with InsDT after propagation. In this paper, we use $\mathbf{v}_k^i$ and $\mathbf{v}_k^{<i,j>}$ to denote the voxel in the block B_i and the interface $< i, j >$. For each voxel $\mathbf{v}_k$, we store both the distance v_k and the most closest boundary voxel $DT(\mathbf{v}_k)$. Notice that $\mathbf{v}_k^i, \mathbf{v}_k^j$, and $\mathbf{v}_k^{<i,j>}$ refer the same voxel, and we always have $v_k^{<i,j>} = \min(v_k^i, v_k^j)$.

Inside-Block distance transform (InsDT) computes the distance transform inside the target block. Different distance transforms algorithms can be used for InsDT. In our method, we implement InsDT by adapting the exact Euclidean distance transform proposed in [MQR03].

Inter-Block propagation (IntBP) is the propagation between the target block B_i and its neighboring interface $< i, j >$. The propagation contains two directions: propagating from B_i to $< i, j >$ and propagating from $< i, j >$ to B_i. On the one hand, after applying InsDT to the block B_i, the propagation from B_i to interface $< i, j >$ should be launched to propagate inconsistency information to outside, where the distances at each voxel $\mathbf{v}_k^{<i,j>}$ in $< i, j >$ and the corresponding voxel $\mathbf{v}_k^i$ in B_i are compared. If $v_k^i < v_k^{<i,j>}$, then $v_k^{<i,j>}$ is replaced with v_k^i. Such inconsistent voxels are called *collision voxels*. On the other hand, before applying InsDT to the block B_i, the propagation from interface $< i, j >$ to B_i should be launched to ensure that the inconsistency information from outside can be eliminated after InsDT, where the distances at $\mathbf{v}_k^{<i,j>}$ and $\mathbf{v}_k^i$ are compared. If $v_k^{<i,j>} < v_k^i$, then v_k^i is replaced with $v_k^{<i,j>}$.

Block ordering: Notice that in such a framework, the processing ordering of blocks has a great influence on the performance. Fig 3 shows a 1D simple example. Since the block B_1 has more background voxels (marked as 0 in Figure 3) and the block B_2 has more foreground voxels (marked as m in Figure 3), B_1 is closer to the boundary than B_2. Therefore, if we apply InsDT to B_1 first like the ordering (b), the block B_1 has no need to be updated. But if we apply InsDT to B_2 first like the ordering (a), we need one more InsDT for B_2 to incorporate the distance information propagated from B_1. When the number of blocks increases, a good ordering can significantly reduce the computational cost. Existing work [MTFS07] uses an empirical ordering method for the blocks. We propose a novel ordering measure, called *boundary Index*, for each block to improve the efficiency as discussed in the following.

3.2. Boundary index

With InsDT and IntBP, we can generate distance fields in an out-of-core fashion, but the performance is affected by the ordering of blocks. We observe that when InsDT is applied on a block near the shape boundary, it is likely that this block requires no more InsDT

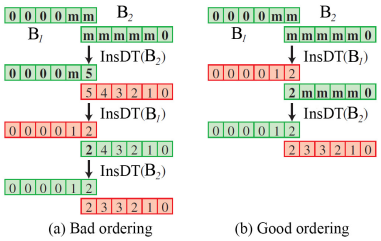


Figure 3: *A good ordering can reduce the computational cost. The ordering (a) requires 3 InsDT to construct the global distance field while the ordering (b) only requires 2 InsDT. [MTFS07]*

in the further iterations. This means that we can construct the global distance field with less number of InsDT if we apply InsDT along the shape boundary with an inward ordering. Hence, we propose a priority measure, called *boundary index*, for each block to measure the possibility of the block being near the shape boundary. In this way, by iteratively applying InsDT to the block with the largest boundary index, we can intuitively reduce the number of InsDT that is required and improve efficiency.

Clearly, if a block contains more foreground voxels, it is closer to the shape boundary and should be applied with InsDT earlier. Thus, the boundary index c_i of each block B_i is initialized as σf_i, where σ is a constant weight and f_i is the number of foreground voxels in B_i. Following the inward ordering idea, when a block is applied with the InsDT, the collision voxels between the block and its interfaces can be regarded as a virtual shape boundary. Therefore, when propagating to $< i, j >$ from B_i, we update the boundary index c_j of B_j as $c_j + T$, where T is the number of collision cells in $< i, j >$. In this way, we can find the block that is closer to the shape boundary by selecting the block with the maximized boundary index. Notice that the boundary index is initialized with the number of foreground voxels and updated by the number of collision voxels. Apparently, the foreground voxels are prior to the collision voxels in finding the blocks near the shape boundary. Hence, we set $\sigma = 1000$ in the experiments to ensure the priority of foreground voxels in finding the blocks near the shape boundary.

3.3. Algorithm

The flow of the out-of-core distance transform is detailed in Algorithm 1. The algorithm first decomposes the input volumetric shape into blocks. Then, we initialize the boundary index and the distance field, where we set the distance of background voxels as 0, and the distance of foreground voxels as ∞. Afterward, we iteratively select the block with the largest boundary index to apply the InsDT and propagate the distance information to the neighboring interfaces. Each time we apply InsDT to a block, the boundary index for this block will be updated as 0, meaning that there is no inconsistency between the block and its neighboring interfaces. The iterations stop when there is no inconsistency between the blocks and all interfaces. The framework can compute large and exact distance fields with small memory usage.

Algorithm 1: Out-of-Core Distance Transform (OocDT)

Input: **V**, the input volumetric shape ;
Divide **V** into blocks;
Priority Queue $Q = \emptyset$;
for *each* $B_i \in V$ **do**
  Initialize B_i and c_i, push c_i to Q;
end
while $\max(Q) > 0$ **do**
  $t = \arg\max(Q)$, Q.pop() ;
  /* Load B_t and $< t, j >$ from storage */
  Propagate from each interface $< t, j >$ to B_t;
  Apply local distance transform InsDT(B_t), $c_t = 0$;
  Propagate from B_t to each interface $< t, j >$;
  update c_j for each neighbor block B_j of B_t;
  /* Save B_t and $< t, j >$ to storage */
end

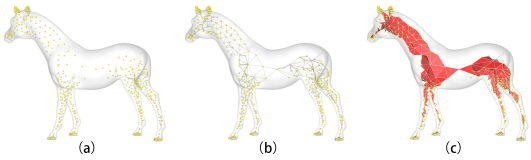


Figure 4: *To generate the medial mesh, we first extract a set of prominent vertices (a) from the distance field, and then construct edge connections (b) and face connections (c).*

4. Medial mesh extraction

Similar to the framework in [CWL22] to capture prominent features for high-quality skeleton extraction with a medial mesh, we also first extract a set of vertices, called *prominent vertices*, which significantly reduces the data size, and then generate the medial mesh by constructing connections among the vertices to capture the topological structure of the shape, as shown in Figure 4. Instead of extracting prominent vertices from the mesh-represented medial surface [CWL22], we propose novel measures for extracting prominent vertices from the volumetric distance field, which saves the trouble of extracting the medial surface. We also develop specialized procedures to leverage the volumetric representation for connection construction. In the following, we first discuss our developed procedures for vertex extraction and connection construction and then introduce our out-of-core implementation.

4.1. Vertex extraction

Inspired by the work [CWL22] which computes a set cover of the medial surface as prominent vertices for high-quality skeleton extraction to reduce the data size, we propose to extract a set of prominent vertices from the set cover of the distance field to save the trouble of extracting medial surface from the distance field, which requires local or global computations for each voxel and is computationally expensive for large volumetric models as discussed in subsection 2.1. In the following, we first briefly review the set cover of a medial surface defined in [CWL22]. Then, we extend the definition of the set cover for distance fields and discuss the rationality and procedures to extract prominent vertices by the set cover of the distance field.

Chu et al. define the set cover of a mesh-represented medial surface $\mathcal{M}$ in [CWL22] as follows:

Definition 1 Given a medial surface $\mathcal{M} = \{\mathcal{V} = \{\mathbf{v}\}, \mathcal{E} = \{\mathbf{e}\}, \mathcal{F} = \{\mathbf{f}\}\}$, where $\mathcal{V}, \mathcal{E}$, and $\mathcal{F}$ are the sets of medial vertices, edges, and facets, respectively, denote the radius function defined on $\mathcal{V}$ as $R(\mathbf{v})$, which decodes the distance of vertex $\mathbf{v}$ to the boundary. We call a set $\mathcal{G} = \{\mathbf{v}\}$ a **set cover** of $\mathcal{M}$ if:

$$\mathcal{G} \subset \mathcal{V}$$
$$\forall \mathbf{v} \in \mathcal{V}, \exists \mathbf{v}_i \in \mathcal{G} \; s.t. \; \mathbf{v} \in \mathcal{B}(\mathbf{v}_i, R(\mathbf{v}_i))$$

where $\mathcal{B}(\mathbf{v}_i, R(\mathbf{v}_i))$ is the inscribed ball centered at $\mathbf{v}_i$ and with radius $R(\mathbf{v}_i)$.

Chu et al. [CWL22] compute a greedy set cover by iteratively selecting the point on the medial surface that corresponds to the largest inscribed ball and removing the medial points that are covered by the selected ball. When no more points can be removed, the remained points on the medial surface form the greedy set cover. The greedy set cover eliminates most influences of boundary perturbations and considerably reduces redundant prominent features, which motivates us to leverage this to improve the efficiency of extracting skeletons from large volumetric shapes.

Inspired by [CWL22], we extend the definition of set cover for a volumetric distance field $\mathcal{D}$ as follows:

Definition 2 Given a distance field $\mathcal{D}$ of a shape volume $\mathbf{V} = \{\mathbf{v}\}$, where we use $\mathcal{D}(\mathbf{v})$ to represent the distance value for a voxel $\mathbf{v}$ of the volume, we call a set $\mathcal{C} = \{\mathbf{v}\}$ a **set cover** of $\mathcal{D}$ if:

$$\mathcal{C} \subset \mathbf{V}$$
$$\forall \mathbf{v} \in \mathbf{V}, \exists \mathbf{v}_i \in \mathcal{C} \; s.t. \; \mathbf{v} \in \mathcal{B}(\mathbf{v}_i, \mathcal{D}(\mathbf{v}_i))$$

where $\mathcal{B}(\mathbf{v}_i, \mathcal{D}(\mathbf{v}_i))$ is the inscribed ball centered at $\mathbf{v}_i$ and with radius $\mathcal{D}(\mathbf{v}_i)$.

We observe that the preserved points in the greedy set cover $\mathcal{G}$ of the medial surface will also be preserved in the greedy set cover $\mathcal{C}$ of the distance field, which is formed by selecting the voxel with the largest distance value because the medial surface can also be regarded as the ridges of the distance field. A 2D example is shown in Figure 5. Hence, we can use the set cover of the distance field as a replacement to capture prominent features. The replacement can save the expensive computational cost of extracting the medial surface from the distance field, which significantly promotes the efficiency of handling large volumetric models. However, we notice that $\mathcal{C}$ also preserves many points which do not exist on the medial surface, as shown by the points near the shape boundary in Figure 5(d). This is because, except for the medial voxels on the medial surface, the volumetric distance field also contains non-medial voxels inside the shape. These non-medial points are redundant and useless for extracting centered curve skeletons. Since they are not on the medial surface and the data size of the set cover is rather small, we can easily remove them from the set cover using existing methods for detecting medial voxels from the distance field. In our implementation, we use the method in [SBTZ02] by removing the non-medial points which have zero divergences.

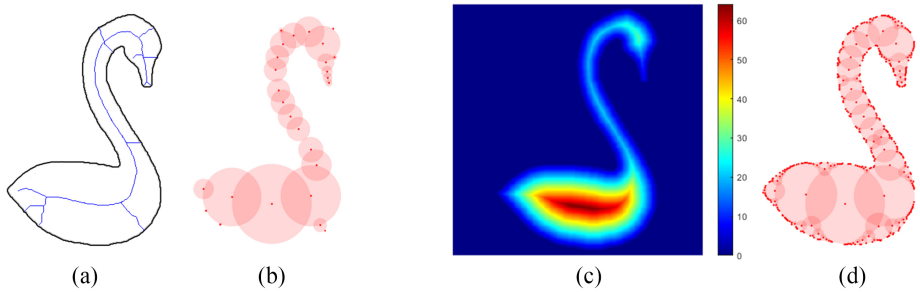


Figure 5: *Using the set cover to capture prominent features. (a) the input shape swan and its medial surface (blue curves); (b) the preserved points (red dots) in the set cover of the medial surface, and the corresponding inscribed balls (red disks); (c) the distance field of the shape swan; (d) the preserved points (red dots) in the set cover of the distance field, and the corresponding inscribed balls (red disks);*

In sum, we extract a set of vertices that compactly captures the prominent features of the shape. It is by first computing a set cover of the distance field, and then removing the voxels whose divergences are smaller than $\varepsilon_d = 0.01$, as suggested in [SBTZ02].

4.2. Connection construction

With the extracted prominent vertices, we construct edge and face connections to generate a medial mesh. As discussed in [CWL22], we can generate a topology-preserving medial mesh if we can keep the connectivity and genus of the original model with the constructed connections.

Edge connection

Since the vertices are extracted from the set cover of a volumetric distance field, no crevice can exist between the neighboring inscribed balls corresponding to the vertices. Therefore, to keep the connectivity, we generate an edge for every two vertices whose corresponding inscribed balls have an intersection.

Face connection

To keep the genus of the original shape, we shall avoid creating or closing a hole in the medial mesh. As discussed above, since no new hole can exist between intersected inscribed balls, we can avoid creating a new hole by generating faces for every three vertices that are connected to each other. To avoid closing a hole, we only generate a face when there is no hole inside the triangle that is formed by the three vertices. This can be easily determined by checking whether there exists a background voxel in the triangle.

4.3. Out-of-core generation of the medial mesh

We propose an out-of-core framework to generate the medial mesh (OocMMG). Similar to OocDT [MTFS07], this framework also works by inside-block computation and inter-block propagation. The inside-block computation selects the voxel $\mathbf{v}_i^*$ inside the block B_i with the largest distance to the boundary and performs covered voxel removal and connection construction inside the block. The inter-block propagation conducts inter-block voxel removal and connection construction. The flow of the framework is detailed in Algorithm 2. The algorithm first initializes the priority measure d_i for each block B_i as the largest distance of the voxels in the block to the boundary and initializes the number of remaining voxels R as the number of voxels inside the shape. Afterward, we iteratively select the block with the largest d_i to perform inside-block computation and inter-block propagation. The iterations stop when $R = 0$ meaning there is no voxel to be selected or removed. As we always update d_i as the largest distances of the remaining voxels in B_i, we can ensure that, in each inside-block computation, the selected voxel $\mathbf{v}_i^*$ in B_i also has the largest distance among the distances of all remaining voxels inside the shape. This means the proposed framework can effectively generate a set cover of the global distance field.

Algorithm 2: Out-of-Core Medial Mesh Generation

```
Input: V, the input distance field;
Priority Queue Q = ∅;
Initialize the remaining voxels R = 0;
for each block B_i ∈ V do
    Initialize priority queue S_i;
    Initialize d_i, push d_i to Q;
    Add R with the number of non-zeros voxels in B_i;
end
while R > 0 do
    t = arg max(Q), Q.pop() ;
    /* Load B_t and <t, j> from storage   */
    Propagate from each interface <t, j> to B_t, update R;
    Apply inside-block computation, update d_t and R;
    Propagate from B_t to each interface <t, j>, update R;
    update d_j for each neighbor block B_j of B_t;
    /* Save B_t and <t, j> to storage     */
end
```

4.3.1. Inside-block computation

Inside-block computation performs vertex extraction and connections construction inside the target block B_i. First, we select the voxel $\mathbf{v}_k^i$ with the largest distance d_k among the remaining voxels inside B_i and compute its divergence to check whether it is on the medial surface. If it is not, we mark $\mathbf{v}_k^i$ as removed and re-select a new $\mathbf{v}_k^i$ among the remaining voxels till it is on the medial surface. Then, we preserve the selected medial voxel $\mathbf{v}_k^i$ as a vertex in

the medial mesh and mark the covered voxels by the inscribed ball centered at $\mathbf{v}_k^i$ with radius d_k as removed by vertex $\mathbf{v}_k^i$. Hence, when removing voxels covered by the ball corresponding to $\mathbf{v}_k^i$, if we find any covered voxel, which has already been marked as removed by some other vertex $\mathbf{v}_m^j$, we construct an edge to connect $\mathbf{v}_k^i$ and $\mathbf{v}_m^j$. In this way, we can save the trouble of checking whether there exists an edge for each pair of two vertices. Then, we generate a face if the two vertices $\mathbf{v}_k^i$ and $\mathbf{v}_m^j$ are connected to the same vertex and there is no hole inside the triangle formed by these three vertices.

4.3.2. Inter-block propagation

The inter-block propagation for generating medial mesh also contains two directions: propagating from block B_i to the interface $< i, j >$ and propagating from interface $< i, j >$ to the block B_j. On the one hand, after the inside-block computation for B_i, if the ball corresponding to the preserved vertex $\mathbf{v}_k^i$ has an intersection with a neighboring block B_j, the propagation from B_i to interface $< i, j >$ should be launched to remove the covered voxels outside B_i. In this propagation, we store the preserved vertex $\mathbf{v}_k^i$ and its corresponding distance and connections to the interface $< i, j >$ to propagate the information for voxel removal and connection construction. On the other hand, before the inside-block computation of block B_i, a propagation from interface $< i, j >$ to B_i should be launched to ensure that the selected voxel in B_i should not be removed by other selected voxels outside B_i. In this propagation, we load the stored information in the interface $< i, j >$ and remove the voxels which should have been removed by other voxels outside B_i, and construct connections with the same procedures in the inside-block computation.

5. Skeleton Extraction

Similar to the procedures in [CWL22], we extract curve skeletons from medial meshes in two steps: spike pruning and face contraction. For spike pruning, we adopt the same method in [CWL22] to cull the spikes in the form of isolated edges, each of which has one endpoint that is connected to no other vertex. For face contraction, in order to extract a centered skeleton, we follow the grassfire analogy [Blu67] and develop novel out-of-core measures to contract the faces towards the center. In the following, we first introduce the data structure of the medial mesh we used and then discuss our out-of-core spike pruning and face contraction of the medial mesh.

5.1. Data Structure

The data size of the medial mesh has been considerably reduced compared to the data size of the volumetric shape. If the medial mesh is small enough to be in-core storage, we can directly perform the in-core computation to extract the curve skeleton. In case the medial mesh is still too large, we adopt a new data structure for the medial mesh to facilitate the out-of-core contraction of the medial mesh. Inspired by the data structure used in [SOW20], we merge the extracted vertices and constructed connections to three separate files to store the medial mesh: vertex.bin, edge.bin, and face.bin. The vertex.bin file stores for each vertex its coordinates, distance to the boundary, other attribute information used in the contraction process, and the location and size of the corresponding information in the edge.bin and face.bin file. The edge.bin and face.bin file store for each vertex its neighboring edges and neighboring faces. With this data structure, we can quickly load the corresponding connection or attribute information of any given vertex, which provides a solid basis for synchronizing the data loading with contraction evolution.

5.2. Spike pruning

We adopt the corresponding measure used in [CWL22] to prune spikes using the stability ratio proposed in [LWS*15]. The stability ratio of an edge (i, j) connecting two vertex $\mathbf{v}_i$ and $\mathbf{v}_j$ is defined as follows:

$$SR(i,j) = \frac{\max\{0, ||\mathbf{v}_i - \mathbf{v}_j|| - |R(\mathbf{v}_i) - R(\mathbf{v}_j)|\}}{||\mathbf{v}_i - \mathbf{v}_j||} \quad (1)$$

where $R(\mathbf{v}_i)$ is the distance of $\mathbf{v}_i$ to the boundary and is also the radius of the inscribed ball centered at $\mathbf{v}_i$. As discussed in [CWL22], we can prune the spikes by culling isolated edges whose stability ratio are smaller than $\varepsilon_s = 0.6$. With the convenience for the connection query provided by the data structure above, our spike pruning can be easily implemented in an out-of-core fashion by a single pass over the edges of the medial mesh.

5.3. Face contraction

To generate a centered skeleton, we perform an out-of-core centrality measurement on the medial mesh and contract the faces towards the center during the measurement. Our centrality measurement follows the grassfire analogy [Blu67] to set fires at all the boundaries of the medial mesh. Clearly, vertices that are more centered will take more time to be burned out, which means that the burning time can be used to measure the centrality of the vertices. As the fire propagates, each triangular face is contracted to the vertex with the longest burning time among the three vertices of the face.

In the centrality measurement, the fire front propagation can also be regarded as a contraction of the boundary of the medial mesh with the guidance of burning time. As discussed in Section 1, when the medial mesh is too large to fit in memory, the fire propagation is unpractical using the in-core block decomposition. Because each time a block is loaded, the fire propagation can only burn out the vertices with the shortest burning time and will have to switch to other blocks under the guidance of burning time, resulting in each block being repeatedly loaded and off-loaded until all the vertices in the block have been burned out. Clearly, this is extremely inefficient. With the convenience of the information query provided by the new data structure, we can solve this problem by synchronizing the data access with the fire propagation with in-core storage of the part of the firefront that has a smaller burning time. In this way, a vertex will be loaded only once when it is on the firefront, and will be off-loaded only once after it is burned out.

In the implementation, we first detect the border of the medial mesh to form the initial fire front and initialize the burning time for the vertices on the initial fire front as the distance to the boundary. The burning times of the other vertices are initialized to infinity. A vertex is determined to be on the border of $\mathcal{M}_s$ if it is related to

only one edge in $\mathcal{M}_s$ or if one of its related edges has only one associated face in $\mathcal{M}_s$. With a single pass over the vertex, we can detect the initial firefront and offload it to the disk. Then, we perform an external sort of the firefront according to the burning time using rsort [AV99]. Afterward, we load a part of the initial firefront with a shorter burning time to start its fire propagation.

In the fire propagation, we find the vertex $\mathbf{v}_b$ with the shortest burning time to be discarded and label it as burned, simulating the fire burning out this vertex. Each time a vertex $\mathbf{v}_b$ is burned, its unburned neighboring vertices on the new border of the burning medial mesh are added to the firefront and have their burning times updated via the following equation:

$$\Theta(v) = \max_{\mathbf{v}_k \in BN(\mathbf{v})} (\Theta(\mathbf{v}_k) + dist(\mathbf{v}, \mathbf{v}_k))$$

where $BN(\mathbf{v})$ is the burned neighborhood of $\mathbf{v}$ and $dist(\mathbf{v}, \mathbf{v}_k)$ is the distance between $\mathbf{v}$ and one of its burned neighbors $\mathbf{v}_k$. Each time a vertex is burned, we also check for each of its neighboring faces whether the face should be contracted. If two of the three vertices constructing the face have been burned out, we contract the face to the remaining vertex. Notice that we also record the most prior burning time T_o of the out-of-core initial firefront, and will continue to load the remaining initial firefront when T_o is smaller than all the burning time of the vertices in the in-core firefront. We continue this process iteratively until all the faces have been contracted.

6. Results

We evaluate our method on volumetric shapes in different resolutions to show its effectiveness in handling out-of-core models and compare it with existing methods. We implemented our method in C++ and run all the experiments on a PC with an Intel i7 CPU @3.20GHz and 16GB RAM.

6.1. Skeleton quality

To evaluate the quality of our extracted skeletons, we picked a set of smooth shapes (Dog, Ant, Bird, and Fertility), and compare our results with those of three state-of-the-art methods, the MCF method [TAOH12], the ET method [YSC*16] and the PF method [CWL22]. In experiments, we feed our method with voxelizations of smooth shapes with resolution 128^3 and divide the volume into 2^3 blocks for out-of-core computation. To generate the results of other methods, we use the software implemented in [MGP10] for the method in [AK01b] to generate the mesh-represented medial surfaces as input for the ET method and the PF method. Besides, we use the default parameter setting of the MCF method and the PF method, and carefully adjust the threshold parameter of the ET method to prune the spikes.

For quantitative comparison, we use three evaluation metrics: the center deviation (CD), the contraction error (CE), and the number of skeleton nodes (vertices). The center deviation proposed in [LW17] measures the centrality of the skeleton by the center deviation on the cross-sections at the skeleton nodes. A smaller center deviation value indicates better centrality of the obtained skeleton. The contraction error is introduced in [CWL22] to evaluate the completeness of the skeleton by measuring the terminal contraction of the skeleton. A smaller contraction error means that the skeleton exhibits less contraction at terminations and represents the shape more completely. The number of the skeleton nodes reveals the compactness of the skeleton.

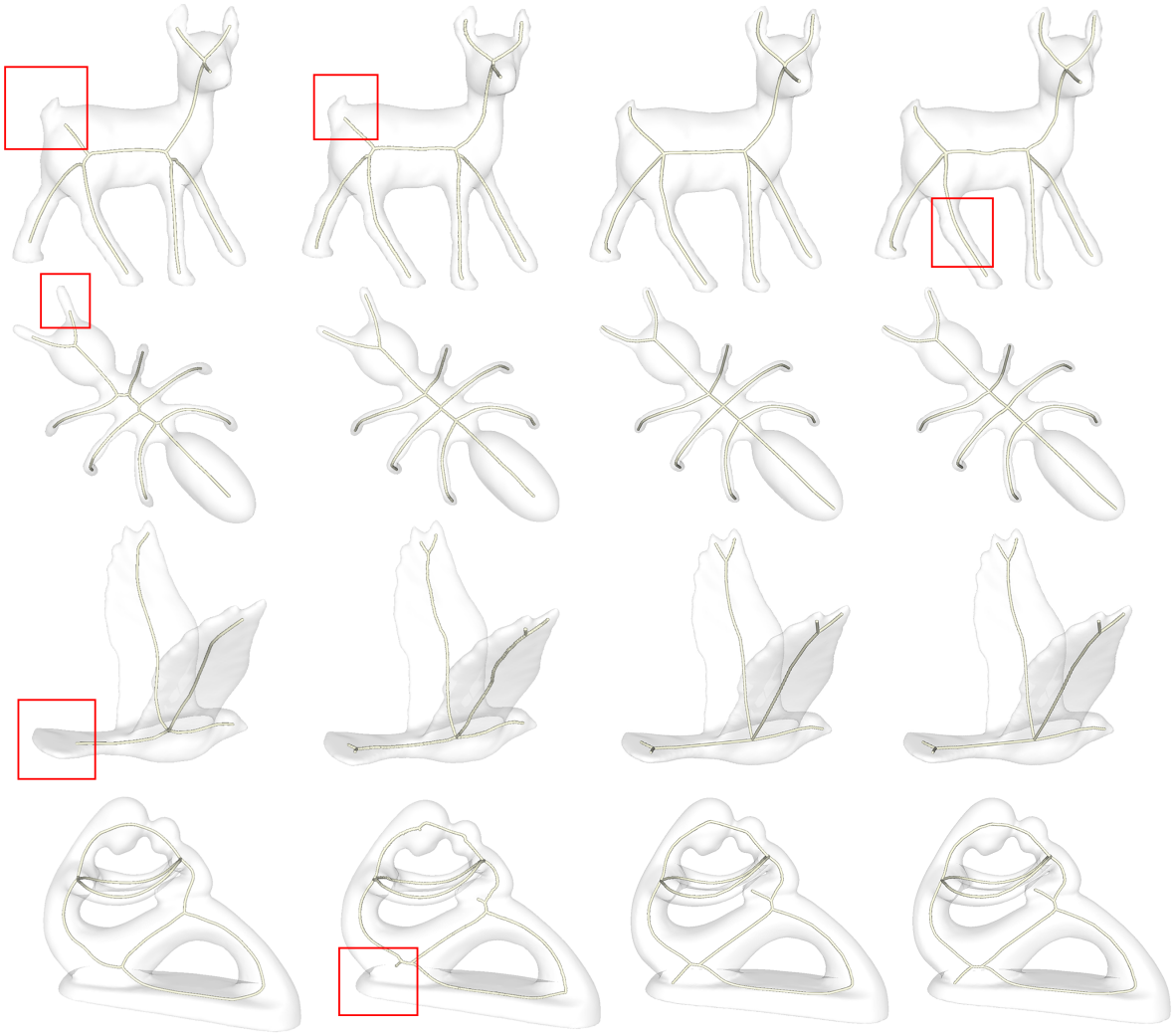
Qualitative Comparison: As shown in Figure 6, our method can generate results that are visually similar to the best results of existing works. The defective parts of the resulted skeletons are highlighted with red rectangles. The results of the MCF method tend to be off-centered and have significant contraction at the terminal of the skeleton. The ET method also contracts the terminal of the skeleton, which prevents the completeness of the skeleton, and its results are unsmooth. The PF method and our method can both generate smooth and complete skeletons, but the results of the PF method is slightly more centered in some part of the shapes. This is because the PF method stores for each skeleton node the corresponding medial surface that is contracted to the node and leverages this information to perform centrality refinement, which is unpractical for large volumetric shapes in terms of efficiency.

Quantitative Comparison: As shown in Table 1, our results have comparable performance with the best results of existing methods under the three quantitative metrics. Our method achieves significantly smaller center deviations than the MCF method and achieves similar levels of center deviation to the ET method and the PF method. As for the Contraction error and the number of skeleton nodes, our method demonstrates comparable performance with the PF method and is superior to the other two methods. Hence, in general, our method can generate centered, complete, and compact skeletons that are comparable to the results of state-of-the-art methods.

6.2. Memory

To evaluate the memory efficiency of our method, we generate two volumetric shapes with resolution $512 \times 512 \times 2048$ from three classic out-of-core models (Buddha, Lucy, and Statue) and evaluate the performance of our method. Table 2 illustrates the running time and memory cost for our method to extract the skeletons. Clearly, our method can extract skeletons from large volumetric shapes with small memory usage. This means that our method can effectively address the memory challenge confronted by existing methods.

In our experiments, existing methods exhaust the memory and crash when extracting skeletons directly from the three out-of-core meshes. Therefore, for a comprehensive comparison, we select the Buddha mesh which has a complex topology structure to conduct an out-of-core simplification [Lin00], and compare the skeleton extracted by the PF method [CWL22] from the simplified mesh with the skeleton extracted by our method from the high-resolution volume. As shown in Figure 7, the resulted skeletons of the PF method have severely topological errors. This may be caused by two reasons. First, the out-of-core simplification introduces severe distortion of the mesh surface and causes topological errors. Secondly, as the PF method use the voronoi-based method to generate the medial surface as its input, it is difficult for existing methods to generate appropriate samples to guarantee a topology-correct medial surface for shapes with many surface details. Hence, our low memory requirements bring another advantage that enables us to process volumes in higher resolutions to acquire skeletons with better quality.



(a) MCF method (b) ET method (c) PF method (d) Our method

Figure 6: *The comparison of our results with the skeletons generated by the MCF method, the ET method and the PF method on different shapes.*

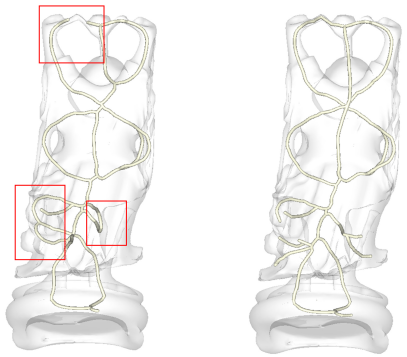
Table 1: *Quantitative evaluations of the skeletons extracted using our method and the compared methods for the models in Fig. 6.*

Shapes	*Center Deviation (CD)*				*Contraction Error(CE)%*				*#Vertices*			
	MCF	ET	PF	Ours	MCF	ET	PF	Ours	MCF	ET	PF	Ours
Dog	0.0856	0.0594	**0.0538**	0.617	7.34	4.25	1.58	**1.56**	615	9037	114	120
Ant	0.0634	0.0215	**0.0198**	0.0219	6.67	5.32	**1.38**	1.43	517	7769	68	59
Bird	0.1428	**0.1207**	0.1233	0.1319	9.63	6.11	**1.82**	1.89	402	5503	39	42
Fertility	0.0776	0.0535	**0.0529**	0.0546	-	10.12	**7.15**	7.36	747	13783	145	141

The *Center deviation* measures the centrality of the skeletons. The *contraction error* measures the completeness of the extracted skeletons in representing the shape. *#Vertices* is the number of skeleton nodes, which reflects the compactness of the skeleton. Here, the bold numbers indicate the best values for the corresponding evaluation metrics. "-": MCF method have no CE value for fertility because it contracts the feature and generates a skeleton with no endpoints.

Table 2: *The performance of our method when processing large volumetric models.*

Model	Reconstruction Error			Running Time(min)				Memory(MB)
	Facets Size	Voxel Resolution	Block	OocDT	OocMMG	Contraction	Total	
Buddha	1055726	$512 \times 512 \times 2048$	$4 \times 4 \times 16$	6.63	0.89	0.13	7.65	69.34
Lucy	14027872	$512 \times 512 \times 2048$	$4 \times 4 \times 16$	7.18	1.01	0.12	8.32	73.27
Statue	386488573	$512 \times 512 \times 2048$	$4 \times 4 \times 16$	9.46	1.19	0.16	10.81	74.64



(a) the PF method (b) our method

Figure 7: *Comparison of our skeleton generated from the high-resolution volume with the skeleton generated from the simplified Buddha mesh by the PF method.*

6.3. Efficiency

We propose two strategies to improve the efficiency of extracting skeletons from large volumetric models: a novel ordering measure in the out-of-core distance transform based on boundary index and using the set cover of the distance field to replace the set cover of the medial surface. We validate the effectiveness of these two strategies through two experiments.

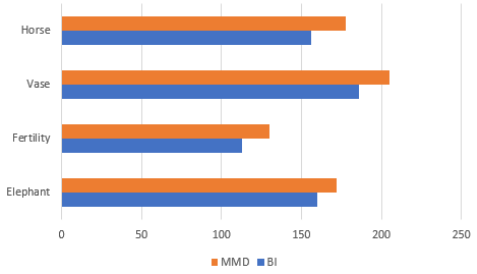


Figure 8: *Running time (s) for computing OocDT of four models using two different ordering measures BI and MMD.*

To evaluate the proposed ordering measure for OocDT, We pick a set of shapes (Elephant, Fertility, Vase, and Horse) to compute the out-of-core distance transform, and compare our ordering based on boundary index (BI) to the ordering based on MMD adopted in [MTFS07]. In experiments, we used voxelizations with resolution 512^3 as the input for the OocDT algorithm and divided the volume into 4^3 blocks. Figure 8 illustrates the running time for OocDT using the two different ordering measures. Clearly, our proposed ordering measure achieves better performance.

Next, we evaluate the replacement of the set cover. We pick two shapes (Vase and Horse) to extract skeletons for voxelizations in 10 different resolutions n^3 ($n = 256, 512, \ldots, 2560$) and compare the running time by using or not using the replacement. In the experiment, we divided the volumes into 4^3 blocks for the first five resolutions and 8^3 blocks for the last five resolutions. As shown in Figure 9, the replacement of the set cover significantly improves the efficiency, especially for the large volumetric models.

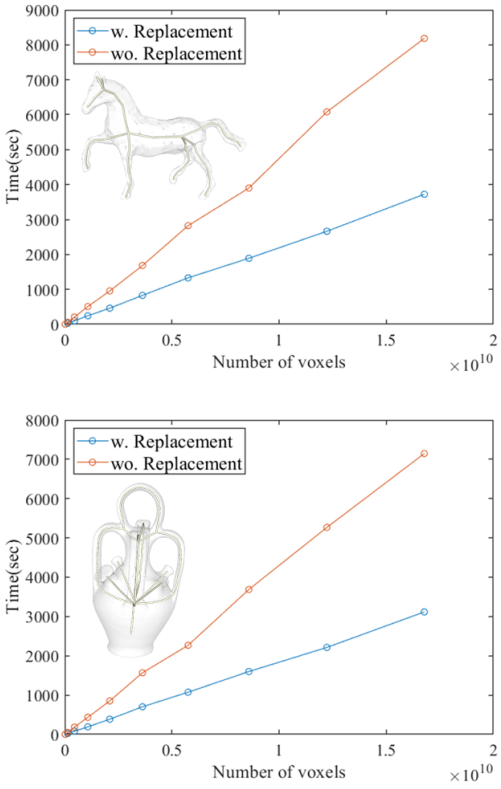


Figure 9: *Running time (s) for extracting curve skeletons of models in different resolutions using or not the replacement of the set cover. w. Replacement/ wo. Replacement: with/without replacement.*

7. Conclusion

To solve the challenges in memory and computational cost posed by the increasing data size, we present a method for out-of-core curve skeleton extraction of large volumetric 3D models. The method first computes an out-of-core distance transform, where we propose a novel ordering measure to improve the efficiency. Then, we propose novel out-of-core measures to generate a medial mesh from the distance transform to reduce the data size and capture the prominent features for skeleton extraction. Finally, the method extracts the curve skeleton by contracting the medial mesh, where we propose an efficient out-of-core contraction framework by synchronizing the data loading with the contraction evolution. Experimental results show that our outcomes have comparable quality with those of the state-of-the-art methods. More importantly, our method has lower limits in terms of the amount of memory space available and performs efficiently for high-resolution models.

Limitation and future work. As shown in Table 1, our results are slightly inferior to those of the state-of-the-art methods in terms of centrality. This is because our center-oriented face contraction only propagates the fire along the edges of the medial mesh and omits the propagation on the faces to reduce the computational cost. The PF method [CWL22] solves this problem by performing centrality refinement for each skeleton node on the cross-section. However, such a centrality refinement requires global intersection computations, which are computationally expensive for out-of-core models. Hence, enhancing the centrality of extracted skeletons by the out-of-core method is an important future work.

Acknowledgements: This work is supported in part by the National Natural Science Foundation of China under Grant 62072446. The authors would like to thank the Princeton Shape Benchmark and Aim@Shape for providing the models used in this paper. The authors also want to thank the anonymous reviewers for their helpful suggestions.

References

[AB02] AYLWARD S. R., BULLITT E.: Initialization, noise, singularities, and scale in height ridge traversal for tubular object centerline extraction. *IEEE transactions on medical imaging 21*, 2 (2002), 61. 1

[ABS11] ARCELLI C., BAJA G. S. D., SERINO L.: Distance-driven skeletonization in voxel images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 4 (April 2011), 709–720. 3

[Ack98] ACKERMAN M.: The visible human project. *Proceedings of the IEEE 86*, 3 (1998), 504–511. 1

[AK01a] AMENTA N., KOLLURI R. K.: The medial axis of a union of balls. *Computational Geometry 20*, 1-2 (2001), 25–37. 2

[AK01b] AMENTA N., KOLLURI R. K.: The medial axis of a union of balls. *Computational Geometry Theory & Applications 20*, 1 (2001), 25–37. 2, 8

[ALL*20] ABERMAN K., LI P., LISCHINSKI D., SORKINE-HORNUNG O., COHEN-OR D., CHEN B.: Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics 39*, 4 (2020), 62–76. 1

[AM96] ATTALI D., MONTANVERT A.: Modeling noise for a better simplification of skeletons. In *Proceedings of International Conference on Image Processing* (1996), pp. 13–16. 2

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. *ACM Transactions on Graphics 27*, 3 (2008), 44:1–44:10. 1, 3

[AV99] ABELLO J., VITTER J.: *External Memory Algorithms and Visualization*. American Mathematical Society Press, 1999. 8

[Blu67] BLUM H.: A transformation for extracting new descriptors of shape. *Models for the Preception of Speech & Visual Form* (1967). 7

[BMA*98] BUTT, MUHAMMAD, AKMAL, MARAGOS, PETROS: Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing 7*, 10 (1998), 1477–1484. 3

[Bor96] BORGEFORS G.: On digital distance transforms in three dimensions. *Computer Vision and Image Understanding 64*, 3 (1996), 368–376. 3

[CB16] COUPRIE M., BERTRAND G.: Asymmetric parallel 3d thinning scheme and algorithms based on isthmuses. *Pattern Recognition Letters 76* (2016), 22 – 31. 1, 3

[ClO*10] CAO J., LIASACCHI A., OLSON M., ZHANG H., SU Z.: Point cloud skeletons via laplacian based contraction. In *Proceedings of the International Conference on Shape Modeling and Applications, SMI* (06 2010), pp. 187–197. 1, 3

[CTMT10] CAO T.-T., TANG K., MOHAMED A., TAN T.-S.: Parallel banding algorithm to compute exact distance transform with the gpu. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, Association for Computing Machinery, p. 83–90. 3

[CWL22] CHU Y., WANG W., LI L.: Robustly extracting concise 3d curve skeletons by enhancing the capture of prominent features. *IEEE Transactions on Visualization and Computer Graphics* (03 2022). 1, 2, 3, 5, 6, 7, 8, 11

[DS06] DEY T. K., SUN J.: Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (2006), SGP '06, Eurographics Association, pp. 143–152. 3

[FCTB08] FABBRI R., COSTA L. D. F., TORELLI J. C., BRUNO O. M.: 2d euclidean distance transform algorithms: A comparative survey. *Acm Computing Surveys 40*, 1 (2008), 1–44. 3

[GRS06] GIESEN J., RAMOS E. A., SADRI B.: Medial axis approximation and unstable flow complex. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry* (2006), pp. 327–336. 2

[JBS06] JONES M. W., BAERENTZEN J. A., SRAMEK M.: 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 581–599. 3

[JCZ19] JIANG H., CAI J., ZHENG J.: Skeleton-aware 3d human shape reconstruction from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 5430–5440. 1

[JKT13] JALBA A. C., KUSTRA J., TELEA A. C.: Surface and curve skeletonization of large 3d models on the gpu. *IEEE Transactions on Pattern Analysis & Machine Intelligence 35*, 6 (2013), 1495–1508. 2

[JST16] JALBA A. C., SOBIECKI A., TELEA A. C.: An unified multiscale framework for planar, surface, and curve skeletonization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 1 (2016), 30–45. 1, 3

[KMH11] KRISHNAMURTHY A., MCMAINS S., HALLER K.: Gpu-accelerated minimum distance and clearance queries. *IEEE Transactions on Visualization and Computer Graphics 17*, 6 (2011), 729–742. 3

[LB07] LOHOU C., BERTRAND G.: Two symmetrical thinning algorithms for 3d binary images, based on p-simple points. *Pattern Recognition 40*, 8 (2007), 2301 – 2314. 1, 3

[LGS12] LIVESU M., GUGGERI F., SCATENI R.: Reconstructing the curve-skeletons of 3d shapes using the visual hull. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 891–901. 3

[LH11] LI C., HAMZA A. B.: Skeleton path based approach for nonrigid 3d shape analysis and retrieval. In *Proceedings of the 14th International Conference on Combinatorial Image Analysis* (2011), pp. 84–95. 1

[LHW*13] LI G., HUI H., WU S., COHEN-OR D., CHEN B.: L1-medial skeleton of point cloud. *ACM Transactions on Graphics 32*, 4 (2013), 1–8. 1, 3

[Lin00] LINDSTROM P.: Out-of-core simplification of large polygonal models. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 259–262. 8

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 131–144. 1

[LS13] LIVESU M., SCATENI R.: Extracting curve-skeletons from digital shapes using occluding contours. *Visual Computer 29*, 9 (2013), 907–916. 3

[LW17] LI L., WANG W.: Contracting medial surfaces isotropically for fast extraction of centred curve skeletons. *Computer Graphics Forum 36*, 8 (2017), 529–539. 1, 3, 8

[LW18] LI L., WANG W.: Improved use of lop for curve skeleton extraction. *Computer Graphics Forum 37*, 7 (2018), 313–323. 1, 3

[LWC20] LI L., WANG W., CHU Y.: A simple and stable centeredness measure for 3d curve skeleton extraction. *IEEE Transactions on Visualization and Computer Graphics* (08 2020), 1–1. 1, 3

[LWS*15] LI P., WANG B., SUN F., GUO X., ZHANG C., WANG W.: Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG) 35*, 1 (2015), 8. 7

[Man04] MANOCHA K. D.: Exact computation of the medial axis of a polyhedron. *Computer Aided Geometric Design 21*, 1 (2004), 65–98. 2

[MBC12] MA J., BAE S. W., CHOI S.: 3d medial axis point approximation using nearest neighbors and the normal field. *Visual Computer 28*, 1 (2012), 7–19. 2

[MGP10] MIKLOS B., GIESEN J., PAULY M.: Discrete scale axis representations for 3d geometry. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 101. 8

[MNS09] MICHIKAWA T., NAKAZAKI S., SUZUKI H.: Efficient medial voxel extraction for large volumetric models. In *WSCG '2009: Full Papers Proceedings* (2009), pp. 189–196. 3

[MQR03] MAURER C. R., QI R., RAGHAVAN V.: A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 2 (2003), 265–270. 3, 4

[MTFS07] MICHIKAWA T., TSUJI K., FUJIMORI T., SUZUKI H.: Out-of-core distance transforms. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2007), SPM '07, Association for Computing Machinery, p. 151–158. 2, 3, 4, 6, 10

[PK98] PALÁGYI K., KUBA A.: A 3d 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recogn. Lett. 19*, 7 (May 1998), 613–627. 1, 3

[Pud98] PUDNEY C.: Distance-ordered homotopic thinning: a skeletonization algorithm for 3d digital images. *Computer Vision and Image Understanding 72*, 3 (1998), 404–413. 3

[QHL*20] QIN H., HAN J., LI N., HUANG H., CHEN B.: Mass-driven topology-aware curve skeleton extraction from incomplete point clouds. *IEEE Transactions on Visualization and Computer Graphics 26*, 9 (2020), 2805–2817. 1

[RAV*19] REBAIN D., ANGLES B., VALENTIN J., VINING N., PEETHAMBARAN J., IZADI S., TAGLIASACCHI A.: LSMAT: Least squares medial axis transform. *Computer Graphics Forum 38*, 6 (2019), 5–18. 2

[RP66] ROSENFELD A., PFALTZ J. L.: Sequential operations in digital picture processing. *J. ACM 13*, 4 (oct 1966), 471–494. 3

[RvWT08] RENIERS D., VAN WIJK J., TELEA A.: Computing multi-scale curve and surface skeletons of genus 0 shapes using a global importance measure. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (March 2008), 355–368. 3

[SBTZ02] SIDDIQI K., BOUIX S., TANNENBAUM A., ZUCKER S. W.: Hamilton-jacobi skeletons. *International Journal of Computer Vision 48*, 3 (2002), 215–231. 3, 5, 6

[SOW20] SCHÜTZ M., OHRHALLINGER S., WIMMER M.: Fast out-of-core octree generation for massive point clouds. *Computer Graphics Forum 39*, 7 (2020), 155–167. 7

[TAOH12] TAGLIASACCHI A., ALHASHIM I., OLSON M., HAO Z.: Mean curvature skeletons. *Computer Graphics Forum 31*, 5 (2012), 1735–1744. 3, 8

[THCO09] TAGLIASACCHI A., HAO Z., COHEN-OR D.: Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics 28*, 3 (2009), 1–9. 3

[THP*19] TANG J., HAN X., PAN J., JIA K., TONG X.: A skeleton-bridged deep learning approach for generating meshes of complex topologies from single rgb images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 4536–4545. 3

[TJ12] TELEA A., JALBA A. C.: Computing Curve Skeletons from Medial Surfaces of 3D Shapes. In *Proceedings of Theory and Practice of Computer Graphics* (Harwell Oxford, UK, 2012), The Eurographics Association, pp. 99–106. 3

[TTL*18] TANG Y., TIAN Y., LU J., LI P., ZHOU J.: Deep progressive reinforcement learning for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 5323–5332. 1

[WCLB09] WANG T., CHENG I., LOPEZ V., BRIBIESCA E.: Valence normalized spatial median for skeletonization and matching. In *IEEE International Conference on Computer Vision Workshops* (2009), pp. 55–62. 1

[XZK*20] XU Z., ZHOU Y., KALOGERAKIS E., LANDRETH C., SINGH K.: Rignet: Neural rigging for articulated characters. *ACM Transactions on Graphics 39*, 4 (2020), 58–72. 3

[XZKS19] XU Z., ZHOU Y., KALOGERAKIS E., SINGH K.: Predicting animation skeletons for 3d articulated models via volumetric nets. In *Proceedings of the International Conference on 3D Vision* (2019), pp. 298–307. 3

[YLJ18] YAN Y., LETSCHER D., JU T.: Voxel cores: efficient, robust, and provably good approximation of 3d medial axes. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 44. 2

[YSC*16] YAN Y., SYKES K., CHAMBERS E., LETSCHER D., JU T.: Erosion thickness on medial axes of 3d shapes. *ACM Trans. Graph. 35*, 4 (July 2016), 38:1–38:12. 1, 3, 8

[YXM*15] YU H., XIE J., MA K. L., KOLLA H., CHEN J. H.: Scalable parallel distance field construction for large-scale applications. *IEEE Transactions on Visualization and Computer Graphics 21*, 10 (2015), 1187–200. 3