Contraction-based curve skeletonization of 3D meshes

Yasan, H.C.

Award date:
2012

Eindhoven University of Technology
Faculty of Mathematics and Computer Science

CONTRACTION-BASED CURVE SKELETONIZATION OF 3D MESHES

A Thesis in
Computer Science and Engineering
by
Haluk Can Yasan

October 2012

The thesis of Haluk Can Yasan was reviewed and approved by the following:

Andrei Jalba
Thesis Advisor, Chair of Committee
Faculty of Mathematics and Computer Science
Eindhoven University of Technology

Huub van de Wetering
Faculty of Mathematics and Computer Science
Eindhoven University of Technology

Herman Haverkort
Faculty of Mathematics and Computer Science
Eindhoven University of Technology

Abstract

In this thesis, we present a framework for extracting high-quality curve skeletons from 3D meshes. For this, we propose one novel method and two others that are improved versions of two state-of-the-art methods. In our main method we utilize surface skeletons as an intermediate stage and investigate if this addition improves the quality of the final curve skeletons. For judging their quality, we build a set of criteria, which is then used to compare the results of relevant methods. Our framework extracts curve skeletons which are more detailed, better sampled, smoother and (in most of the cases) better centered. The extraction process is also insensitive to surface noise and small changes in parameters. We also propose an idea for a novel application area for curve skeletons and describe the current progress towards building this new framework and future plans.

Table of Contents

Chapter 1

Introduction

Skeletons constitute more abstract and more compact representations of shapes. They are used for several purposes, including, but not limited to: shape matching, surface reconstruction, animation, shape extraction and editing. In 3D, there exist two different types: surface and curve skeleton.

The surface skeleton of a 3D shape is formally defined, whereas the curve skeleton lacks such a definition. Therefore, in order to assess a method that extracts curve skeletons, a set of criteria needs to be selected. The criteria in this set vary based on the application area.

In this work, we present an approach for extracting curve skeletons that improves on existing state-of-the-art methods. We judge this improvement by comparing the results of our method to those of the existing ones, using a set of general-purpose criteria.

This thesis is organized as follows. In Chapter 1, we begin with a general introduction to skeletonization. In Chapter 2, we give an overview of four curve-skeleton extraction methods, which are closely related to our approach. Next, in Chapter 3, the details of our process are presented. We discuss our results and make detailed comparisons in Chapter 4. In Chapter 5, a road map for future work is drawn. Lastly, we conclude this thesis in Chapter 6.

In this chapter, we first give a general description of skeletonization, then an overview of the related previous work follows. We end the chapter by briefly explaining our contribution.

1.1 General Description of Skeletonization

This section briefly describes what skeletonization is. Deeper details follow in later chapters.

Skeletons (sometimes referred as medial axes) are structures which represent a shape in a more compact way. Among the two types of skeletons in 3D, the surface skeleton has a formal definition while the curve skeleton lacks one. Therefore, the definition of curve skeleton in 3D can be modified in favor of the application, by putting more emphasis on the desired qualities. We, too, have built a set of criteria to analyze our results and make comparisons.

In 2D, definitions of both types lead to the same representation: a 1D curve built by the curve segments that connect the centers of maximally inscribed discs within the object. For 3D objects, however, separate definitions are required. Points on the surface skeleton are the centers of maximally inscribed balls within the object [1]. Since a skeleton point is the center of such a ball, this means that there are at least two points on the surface which are equally far from the skeleton point. These points are also the closest points to the skeleton point and are called *feature points*. Feature points can be used for different purposes, e.g. determining the importance of a skeleton point, based on the angle between its feature points and the length of the shortest geodesic between the feature points on the original surface. The complete surface skeleton consists of a set of 2D manifold sheets containing the loci of surface skeleton points and a set of one dimensional curves where the manifolds intersect or join.

A curve skeleton, on the other hand, consists of only a set of 1D curves, which are "centered" within the object. This centeredness is mostly evaluated in a local sense. Even if we cannot formally define a curve skeleton, there are still some properties which are desired for most of the applications. Some of these properties are listed by Reniers et al. [1]. Among those, the ones which are essential to us are listed below.

- **Homotopic:** Isometric transformations should not affect the skeleton. Geometrical and topological properties of the shape (e.g. bends, branches, holes) should be visible on the curve skeleton.

- **Thin:** A curve skeleton must only consist of 1D curves, no faces or lines with different thickness are allowed.

- **Centered:** Skeleton nodes are centered with respect to the surface (mesh) boundary.
- **Junction detective:** Junctions of the original surface should be easily detectable just by looking at the curve skeleton.

We also propose a few additional criteria to judge the quality of a skeletonization method:

- **Detail preserving:** If an application needs to have a solid idea about how detailed the object is, it is desired to be able to see small (yet important) details in the curve skeleton, e.g. fingers of a hand. However, while preserving the details, skeletonization should not be prone to noise.
- **Smooth:** Sharp bends caused by poorly sampled curve skeletons (long line segments, inadequate number of skeleton nodes) look unnatural, since it gets harder to imagine the shape of the real object just by looking at the skeleton. Therefore, curves and bends of a surface should be preserved in the curve skeleton.
- **Independent from mesh sampling:** Mesh-based methods for extracting curve skeletons work better with dense meshes, as this provides more information and higher precision. We find it important that a method extracts geometrically and structurally similar curve skeletons using the same shape with different sampling rates.

Various approaches have been proposed for extracting skeletons. Here, however, we focus on a number of related methods, which we briefly describe in Section 1.2. In Chapter 2, we examine more thoroughly the ones which are very similar (in terms of extraction method) or relevant (in terms of method or purpose) to our work.

1.2 Related Work

There are many different methods in the literature for extracting curve skeletons of 3D shapes. These methods can be divided into two groups considering the shape representation used: volumetric methods employ a voxel-based representation and surface-based methods rely on polygon meshes or point clouds. Another division criteria can be the purpose of the application which makes use of the curve skeleton.

Reniers et al. compute surface and curve skeletons of 3D shapes based on a volumetric representation and utilize an importance measure. In order to compute the curve skeleton, a point is considered to be on the skeleton if it has at least two shortest geodesics (a path on the surface with minimum length) between its feature points [1]. During the computation, a map between skeleton points and object surface is also built using *collapse measure*. The importance of a point is therefore determined by how big the part of the surface mapped to it is, i.e., the size of its collapse measure. This importance measure is also monotonic and can be used to simplify the skeletons while keeping the connectivity of the object intact. The downside of this method is that it can only calculate skeletons of genus 0 objects, i.e. objects that do not have holes or tunnels.

Reniers and Telea use another method for voxel-based objects, based on the curve skeleton definition in [1]. This approach extracts better-centered curve skeletons because it selects the curve skeleton points from the surface skeleton, which is centered by definition. Also, the two shortest geodesics which belong to the skeleton point have the same length, which is another indication of centeredness. They extend the definition of curve skeletons given in [1] to cover objects which are of higher genera. In the new definition, a point is considered to be in the curve skeleton if its shortest geodesic set contains a ring. The shortest geodesic set also provides information if a skeleton point is a junction or not [2]. This is essential for the method since the main aim of Reniers and Telea in [2] is object segmentation, i.e. the identification of the different parts of the shape.

Li et al. propose a method for extracting skeletons from a mesh, to be used in interactive applications [3]. The aim of this work is to identify the meaningful parts of the object (e.g. distinguish the main body and its extremities). For this, first a series of edge collapses are performed until only a 1D set of curves remains (for the geometric definition of edge collapse, refer to Section 3.2.2). These curves are initially not connected, as this step does not preserve the connectivity of the mesh. The curves are then connected using virtual edges. In the last step, the skeleton is used as a path for the space sweeping which determines the different parts of the shape. Although the extracted skeleton looks like a curve skeleton, the results do not meet most of the criteria we consider essential, such as centeredness and smoothness (Figure 1.1).

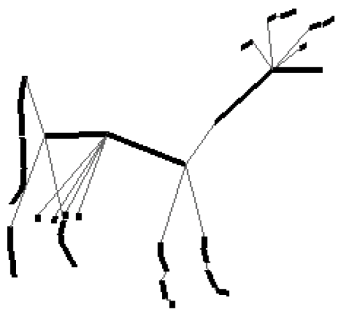


Figure 1.1: A skeleton extracted by the method of Li et.al. The thick lines are the results of edge collapsing and the thin lines are the virtual edges used to connect the unconnected skeleton parts [3].

Au et al. present an effective approach for extracting curve skeletons from meshes [4]. The mesh is first contracted until its volume gets close to zero by using a Laplacian-based surface smoothing. This is done by moving the mesh vertices in the direction of their inner normal, with speed proportional to the mean curvature at each vertex. Next, a series of edge-collapse operations are performed on the contracted mesh to get rid of the mesh structure (remove the faces) and extract the curve skeleton. This process stops when no faces remain. After edge collapses, the resulting curve skeleton is re-centered using a weighted average of the positions of the corresponding vertices of the original mesh. This correspondence information is built as a skeleton-node-to-mesh mapping during the edge-collapse stage (Figure 2.4, middle). Re-centering is necessary because the contraction step often moves the vertices off-center and even outside of the mesh boundary. This is even more likely to happen in thick regions of the shape or narrow parts with a sharp corner (such as a bent arm).

A method partly based on [4] and tuned specifically for surface reconstruction is developed by Cao et al. This method can extract curve skeletons from point clouds (possibly with missing parts) [5]. The initial point cloud is first contracted using the Laplacian-based contraction of [4]. After contraction, the initial shape is reduced to a point set with a very small volume. This point set is used to generate a connected skeleton graph, in order to be able to perform edge collapses. The connectivity information is generated using farthest point sampling. The edges of the graph are then collapsed using a simple cost scheme based on their lengths, until only a 1D curve remains. The resulting skeleton is re-centered as the contraction process disrupts centeredness.

Another approach for the purpose of surface reconstruction, similar to [5], is proposed by Tagliasacchi et al. in [6]. The idea is to use a generalized rotational symmetry axis (ROSA) as the re-definition of the curve skeleton. ROSA is based on rotational symmetry, as opposed to reflectional (positional) symmetry used by most of the other methods. Tagliasacchi et. al. also use point clouds as input, often retrieved from 3D scanners and might have significant missing data. In order to calculate the ROSA points, the surface is cut using planes and then a neighborhood within a small radius is determined around the cut area. A ROSA point is the one whose orientation minimizes the variance of angles between itself and normals of the point samples in the cut neighborhood (Figure 2.5a). Such a point also minimizes the sum of squared distances to the line extensions of the normals of the neighborhood samples (Figure 2.5b). After the skeleton (ROSA) is built, it is used to repair and reconstruct the surfaces of point clouds.

Telea and Jalba use a different kind of contraction, which uses the information provided by the surface skeleton (i.e. importance of feature points) [7]. This method also differs from other methods [4, 5, 6] by using an intermediate stage (given by the surface skeleton), rather than extracting the curve skeleton directly from the original surface. For each point in the surface-skeleton point cloud, Telea and Jalba define the importance as the length of the shortest surface geodesic between the two feature points of the surface skeleton point. The gradient field of this importance is used to iteratively contract the surface-skeleton mesh using an advection transport process. However, the whole process results a mesh with very small volume and small total area of faces, rather than a 1D curve. Therefore this fails to comply with one of the criteria given in Section 1.1: being thin. Another problem faced is that this contraction process does not have positional constraints for vertices which should preserve extremity details of the object. Hence, such details (e.g. fingers of a hand or horns or a cow) disappear or get unrecognizable as the procedure iterates.

Dey and Sun's method also extracts curve skeletons from the surface skeleton (which is also called medial surface) [8]. Here, feature points and the distance transform used in [7] are gathered under the name of *medial geodesic function* (MGF). MGF behaves like a distance function and is continuous everywhere except at a measure zero set, i.e. when the singularities of MGF comprise curves or isolated points within the set of feature points. Curve skeleton is then defined as the points on the surface skeleton which are the singularity points with respect to MGF. As a result, the divergence of the vector field generated by the gradient of MGF will be undefined at such points. The issues of such a method are clearly visible in Figure 1.2: the resulting skeletons make small zig-zags even in the regions where the surface is smooth. On the other hand, curve skeletons are fairly well-centered and preserve enough details since the curve skeleton is actually a subset of the surface skeleton, which has those properties by definition.

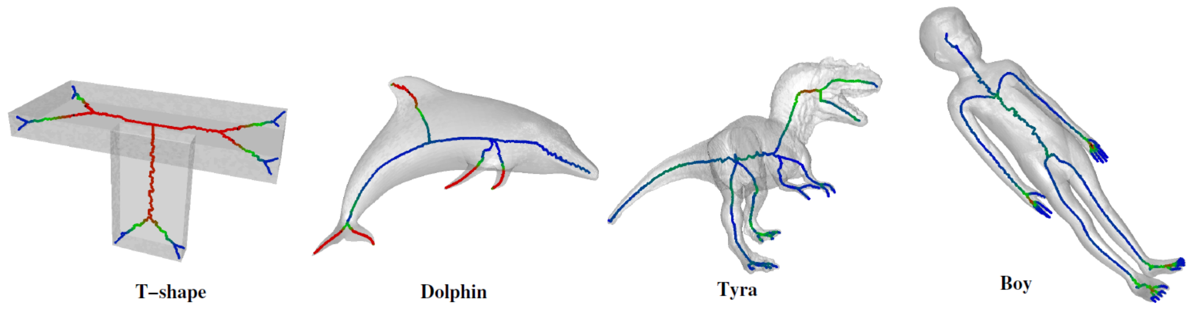


Figure 1.2: Some curve skeletons generated by the Dey and Sun's method [8].

Sharf et al. propose a quite different approach for on-the-fly curve-skeleton extraction using evolving fronts [9]. This method uses meshes and point clouds, and can handle surfaces with missing parts as well. The main idea behind the method is to get a good approximation of the surface by using competing fronts. When the surface is reconstructed, the centers of these fronts are used to build an initial skeleton. This initial version is then processed considering the geometry, branch structure and front performance to extract the final curve skeleton [9]. The processing step makes it possible for the method to handle shapes of higher genera, too. An interesting aspect of this method is that it computes the curve skeleton directly as a 1D curve, unlike many others where edge collapses or thinning procedures are employed. Yet, as a downside, the curve skeletons extracted with this method are not always centered well; since there is no explicit mechanism to enforce centeredness.

Both our method and [7] use the surface skeleton as an intermediate stage. For its fast computation, a ball-shrinking algorithm has been proposed by Ma et al. [10]. The algorithm starts with a ball big enough to contain the entire surface, which is also tangent to one surface vertex. The size of the ball is then iteratively reduced until it becomes maximally inscribed within the surface. Same operation is performed for each vertex on the surface. A detailed description of the algorithm is given in Section 3.1.1.

Contraction-based skeletonization methods are currently the state-of-the-art for extracting curve skeletons from meshes and point clouds. Here, edge collapsing reduces the mesh to a 1D structure and therefore plays a big role as the contraction. We base our edge-collapse step on the mechanism given in [4], which is actually a modified version of the scheme used for the mesh decimation method of Garland and Heckbert [11]. Garland and Heckbert calculate an error matrix for each vertex, based on the squared distance between the vertex and its incident faces. A cost of the edge is then calculated using the error matrices of its vertices. Then, at each iteration, the edge with minimum cost is removed. Such a cost mechanism preserves the similarity between the original mesh and the decimated one. In our method and in [4], the error matrices are calculated using the squared distances between a vertex and its incident edges, as opposed to faces in the original scheme.

1.3 Our Contribution

Our aim is to create a method that extracts curve skeletons from meshes which are better-centered, smoother and have more details. For this purpose, we base our ideas on two methods by Au et al. [4] and by Telea and Jalba [7]. These methods yield high-quality curve skeletons with respect to different sets of criteria. Our framework can extract high-quality curve skeletons which meet a larger set of criteria. As a result, we expect that the curve skeletons extracted by our method to be used in a wider area of application. For future work, we also provide a basis for an application that uses curve skeletons for surface parameterization, which is a novel application area for skeletonization.

Chapter 2

Contraction-Based Methods for Curve Skeletonization. Overview

In this chapter, we examine in detail four methods for extracting curve skeletons, without making extensive comparisons. We have chosen these methods because they either use similar (or the same) techniques to ours, or they focus on the criteria which were laid out in Section 1.1.

2.1 The Method of Au et al.

This method extracts curve skeletons from meshes using the three main steps listed below.

- **Mesh contraction:** In this step, the input mesh is contracted until it has a very small volume and the total area of its faces is close to zero. The contracted mesh preserves the topological properties of the original mesh.

- **Connectivity surgery:** A series of edge-collapse operations are then applied to the contracted mesh. At the end of this step, only a 1D curve remains. Topological properties of the original mesh are still intact at this point.

- **Embedding refinement:** Lastly, the curve skeleton generated by the previous step is re-centered and the skeleton nodes are pulled back inside the mesh boundary, in case they have gone outside, due to over-contraction.

Each of these steps is detailed in the following of this section.

2.1.1 Mesh Contraction

The mesh contraction procedure iteratively moves the mesh vertices in the directions of their inner normals with the moving speed based on the mean curvature at a vertex (first four steps in Figure 2.1). Naturally, this smoothing needs to be constrained so the process does not contract all vertices to a single one. There are thus two main forces competing against each other. One tries to contract the shape while the other tries to constrain the contraction in order to preserve the original shape.

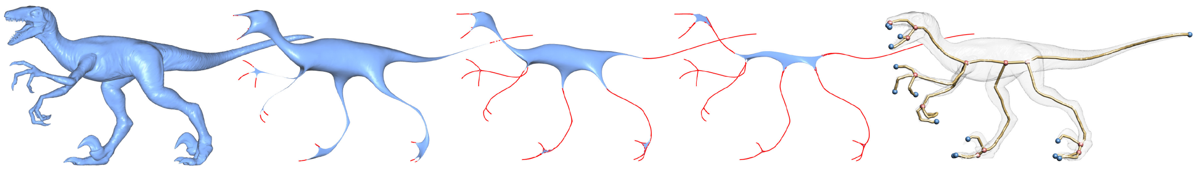


Figure 2.1: Overall view of the contraction process and the final product after all of the main steps [4].

In order to calculate new vertex positions V', the following equation is solved iteratively:

$$\begin{bmatrix} W_L L \\ W_H \end{bmatrix} V' = \begin{bmatrix} 0 \\ W_H V \end{bmatrix}, \tag{2.1}$$

where V is an array containing the current vertex positions. W_L and W_H are diagonal $n \times n$ weighting matrices ($|V| = n$), which contain the weights for contraction and attraction constraints, respectively. Since the weight matrices are diagonal, the ith diagonal elements of these matrices are denoted as $W_{L,i}$ and $W_{H,i}$. L is also an $n \times n$ matrix with cotangent weights (which are used to mimic a curvature flow process) calculated as follows:

$$L_{ij} = \begin{cases} cot\ \alpha_{ij} + cot\ \beta_{ij} & \text{if } (i,j) \in E \\ \sum_{(i,k)\in E}^{k} -L_{ik} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

where α_{ij} and β_{ij} are the opposite angles of the edge (i,j), and E is the set of edges.

Equation 2.1 is solved in least-squares sense, because it is over-determined. This means that solving this equation represents actually minimizing the quadratic energy given by:

$$\|W_L L V'\|^2 + \sum_i W_{H,i}^2 \|\mathbf{v_i'} - \mathbf{v_i}\|^2, \tag{2.3}$$

where $\mathbf{v_i'}$ is the new position of vertex i as opposed to its current position $\mathbf{v_i}$.

During the iteration, the weight matrices W_L and W_H need to be updated. The reason is that after a number of iterations, W_L will not be strong enough to contract the vertices and W_H will not be able to prevent over-contraction to protect the shape. Au et al. update each weight at iteration t as follows:

$$\begin{aligned} W_L^{t+1} &= s_L W_L^t \\ W_{H,i}^{t+1} &= W_{H,i}^0 \sqrt{A_i^0 / A_i^t}, \end{aligned} \tag{2.4}$$

with $s_L = 2.0$. A_i is the total area of the faces within the triangle fan around vertex i. Thus, the more the mesh is contracted, the smaller the face areas become, and the attraction force increases to preserve the shape. In this updating scheme, giving a larger value to s_L will cause the mesh to contract more in less number of iterations with significant detail loss. Giving larger values to $W_{H,i}^0$ will make the contraction harder but will also yield a thicker mesh, since vertices will be kept closer to their initial positions.

2.1.2 Connectivity Surgery

This process removes all faces from the mesh by performing several edge collapses. This is an iterative process where one edge is removed at each step based on a cost measure. The cost of an edge is the sum of two different types of costs. At each iteration, the edge with the smallest cost is removed and costs of related edges are updated. The process terminates when there are no faces left.

Briefly, the steps of this process are:

- **The shape cost** is calculated for each edge.
- **The sampling cost** is calculated for each edge, using the lengths of all edges belonging to the vertices of the subject edge.

- **The total cost** is the sum of these two costs and the removal candidate is the edge with the minimum cost.
- The candidate edge (i, j) is removed, i.e. $\mathbf{v_i}$ is collapsed onto $\mathbf{v_j}$. The faces of this edge are removed and $\mathbf{v_j}$ takes over all remaining edges of $\mathbf{v_i}$. This collapse is denoted as $i \to j$.

Shape Cost: The shape cost is actually based on the cost function developed by Garland and Heckbert for the purpose of mesh decimation [11]. The original scheme uses a quadratic error mechanism based on the squared distances between a vertex and its incident faces. Instead, Au et. al use the squared distance between the vertex and its edges, therefore getting rid of all the faces and obtaining a set of edges.

Initially, a matrix K_{ij} (Equation 2.5) is assigned to every edge with vertex pair (i, j). For an arbitrary point $\mathbf{p}$ (in homogeneous representation), the formula $\mathbf{p}^T(K_{ij}^T K_{ij})\mathbf{p}$ then gives the squared distance of $\mathbf{p}$ to the line formed by the edge (i, j).

In order to calculate the matrix K_{ij}, two vectors ($\mathbf{a}$ and $\mathbf{b}$) are needed. Consider the edge (i, j), where the vertices are $\mathbf{v_i}$ and $\mathbf{v_j}$. Then,

$$\begin{aligned} \mathbf{a} &= \frac{\mathbf{v_j} - \mathbf{v_i}}{\|\mathbf{v_j} - \mathbf{v_i}\|} \\ \mathbf{b} &= \mathbf{a} \times \mathbf{v_i} \end{aligned}$$

$$K_{ij} = \begin{pmatrix} 0 & -a_z & a_y & -b_x \\ a_z & 0 & -a_x & -b_y \\ -a_y & a_x & 0 & -b_z \end{pmatrix}. \tag{2.5}$$

For every vertex $\mathbf{v_k}$, first the quadratic error matrix Q_k is calculated. Then, the initial cost D_k for input point $\mathbf{p}$ is equal to the sum of the squared distances between $\mathbf{p}$ and all edges of $\mathbf{v_k}$:

$$\begin{aligned} Q_k &= \sum_{(k,j)\in E} (K_{kj}^T K_{kj}) \\ D_k(\mathbf{p}) &= \mathbf{p}^T Q_k \mathbf{p}, \end{aligned} \tag{2.6}$$

where E represents the set of the edges in the contracted mesh and Q_k represents the error matrix of $\mathbf{v_k}$. Finally, the shape cost S_1 of an edge $e(i,j)$ is:

$$S_1(i,j) = D_i(\mathbf{v_j}) + D_j(\mathbf{v_j}). \tag{2.7}$$

Note that $\mathbf{v_j}$ is used as the parameter for both D_i and D_j. By this, the cost $S_1(i,j)$ becomes the sum of squared distances between the collapse position, $\mathbf{v_j}$, and all edges of $\mathbf{v_i}$ and $\mathbf{v_j}$.

Sampling Cost: The main aim of the connectivity surgery is to preserve resemblance to the shape of the original surface. At the same time, it is undesired to get a very sparse skeleton with a small number of skeletal nodes together with a lot of long edges and sharp corners. For this purpose, an extra sampling cost is calculated in addition to the shape cost. The sampling cost S_2 for a candidate edge $e(i,j)$ calculates the total length of the edges incident to $\mathbf{v_i}$ and multiplies it with $\|e(i,j)\|$. This increases as the potential travel distance of edges, which $\mathbf{v_j}$ inherits when $\mathbf{v_i}$ is removed, is larger (note that the position of $\mathbf{v_j}$ is not changed after a collapse).

$$S_2(i,j) = \|\mathbf{v_j} - \mathbf{v_i}\| \sum_{(i,k)\in E} \|\mathbf{v_i} - \mathbf{v_k}\|. \tag{2.8}$$

Total Cost: The total cost of an edge $e(i,j)$ is then:

$$C(i,j) = \alpha S_1(i,j) + \beta S_2(i,j). \tag{2.9}$$

After the edge with the minimum cost is removed, the error matrix of the remaining vertex (Equation 2.6) is updated. For the collapse $i \to j$, this is simply the sum of two matrices:

$$Q_j \leftarrow Q_j + Q_i. \tag{2.10}$$

α and β in Equation 2.9 are the weights for the sum. Au et al. suggest $\alpha = 1.0$ and $\beta = 0.1$ [4], since the shape cost dominates the sampling cost due to the cumulative error matrix updates through the iterations.

2.1.3 Embedding Refinement

This post-processing step is required because it is not possible to find a perfect balance between the attraction and contraction weights for the contraction procedure, described in Section 2.1.1. As a result, depending on the geometric properties of the shape region (thickness, curvature, etc.), some skeleton nodes may be contracted too much or too less (Figure 2.2, right). Embedding refinement tries to re-center the nodes and pull them back inside the surface, if necessary. As a last operation, this step also merges the junction nodes which are adjacent to each other, if they fulfill a centeredness criterion.

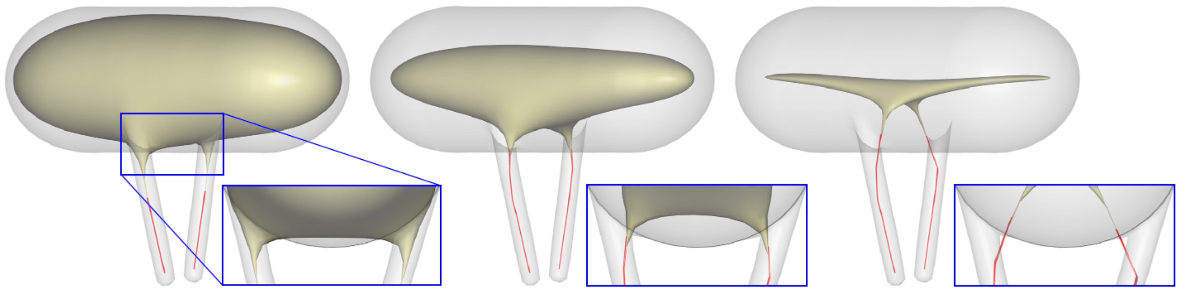


Figure 2.2: Snapshots after different number of contraction iterations. Transparent outer boundary represents the original mesh [4].

Re-Centering

For this purpose, first a skeleton-node-to-mesh mapping is built during the edge collapses. This mapping generally corresponds to a cylindrical mesh region for each skeleton node and provides information about which mesh vertices are contracted or collapsed onto a certain skeleton node (Figure 2.4, middle). Then, a new position is calculated for the skeleton node using a weighted averaging scheme. Each cylindrical mesh region $\Pi_{\mathbf{k}}$, corresponding to the skeleton node $\mathbf{k}$, has two cyclic boundary loops, if $\mathbf{k}$ is not a junction or tip node. Mappings of junction nodes have more than two boundary loops and mappings of tip nodes have only one. For a boundary loop j and vertex index set of such a loop I_j, the following displacement average is calculated:

$$\mathbf{d_j} = \frac{\sum_{i\in I_j} L_{j,i}(\mathbf{v_i'} - \mathbf{v_i})}{\sum_{i\in I_j} L_{j,i}}. \tag{2.11}$$

Here, $L_{j,i}$ is the total length of the two edges that belong to the loop j and incident to vertex i (Figure 2.3). $\mathbf{v_i'}$ and $\mathbf{v_i}$ are the positions of vertex i on the contracted mesh and on the original mesh, respectively. The final position of the node is then calculated as:

$$\mathbf{u} = \mathbf{u} - \frac{\sum_{j=1}^{N} \mathbf{d_j}}{N}, \tag{2.12}$$

where $\mathbf{u}$ is a curve-skeleton node and N is the number of the boundary loops in $\Pi_{\mathbf{u}}$.

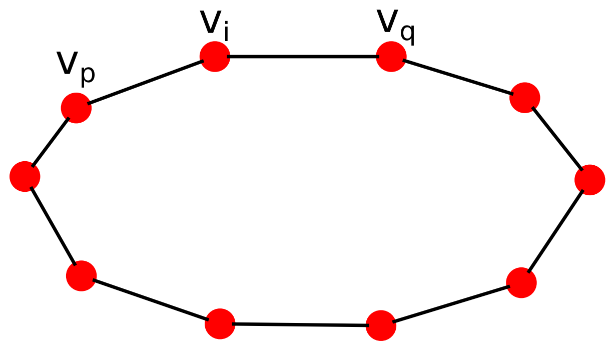


Figure 2.3: A boundary loop j belonging to a skeleton-node-to-mesh-mapping $\Pi_{\mathbf{k}}$. Here, $L_{j,i} = \|\mathbf{v_i} - \mathbf{v_p}\| + \|\mathbf{v_i} - \mathbf{v_q}\|$ (see Equation 2.11).

Merging Junction Nodes

One last issue remaining after re-centering the skeleton is the situation where the curve skeleton has more than one junctions adjacent to each other (Figure 2.4, left). Such junction nodes are merged pair-wise, if one is better centered than the other. The centeredness $\sigma_{\mathbf{k}}$ of a skeleton node $\mathbf{k}$ is defined as the standard deviation of the distances between $\mathbf{k}$ and the vertices in the mapping $\Pi_{\mathbf{k}}$. Two nodes $\mathbf{k}$ and $\mathbf{k'}$ are merged if the inequality $\sigma_{\mathbf{k'}} < 0.9\sigma_{\mathbf{k}}$ holds. In Figure 2.4, right, we can see both cases. At the junction where the arms meet (red frame), the criterion apparently held and the two junction nodes are merged. However, in the foot (green frame), there are still two adjacent junctions, which means that the criterion did not hold.

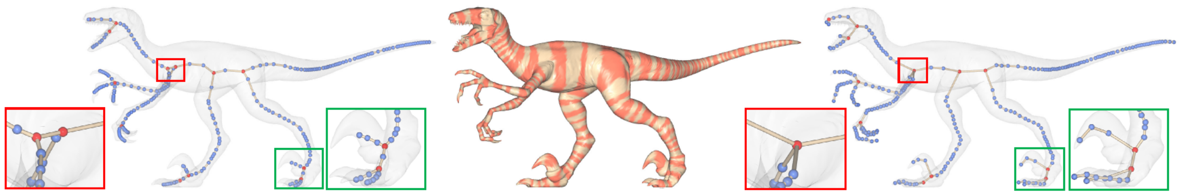


Figure 2.4: Curve skeleton before embedding refinement (left). Skeleton-node-to-mesh mapping visualized (middle). Curve skeleton after embedding refinement (right) [4].

2.2 The Method of Tagliasacchi et al. (ROSA)

The aim of this method is to extract curve skeletons from point clouds which are possibly incomplete. Such point clouds are often obtained by 3D scanning. The extracted curve skeletons are used to repair the missing parts by reconstructing the surface areas corresponding to them [6].

Here, the curve skeleton is renamed to *rotational symmetric axis (ROSA)*, because rotational symmetry is given more emphasis, whereas positional symmetry has more importance while extracting surface skeletons. Therefore, a ROSA point has a minimal variance of angles with respect to the surface points in its local neighborhood (Figure 2.5a). This also means that the ROSA points consider symmetry in a local sense.

The main steps of this extraction process are:

- Finding an optimal cutting plane for each point sample in the point cloud.
- Calculating relevant neighborhood of surface points around the plane.
- Calculating the ROSA points for each neighborhood.
- Handling the joints (identifying the joint regions).
- Re-centering the ROSA points and constructing the final curve skeleton.

2.2.1 Finding ROSA Points

The process starts with choosing a point sample $\mathbf{p_i}$ from the point cloud. Then it finds the plane π_i where $\mathbf{p_i}$ is on. $\mathbf{p_i}$ has normal $\mathbf{v_i}$, which is later used to determine the optimality of the plane. The local neighborhood N_i around π_i consists of the sample points which are at most at a distance δ away from the plane.

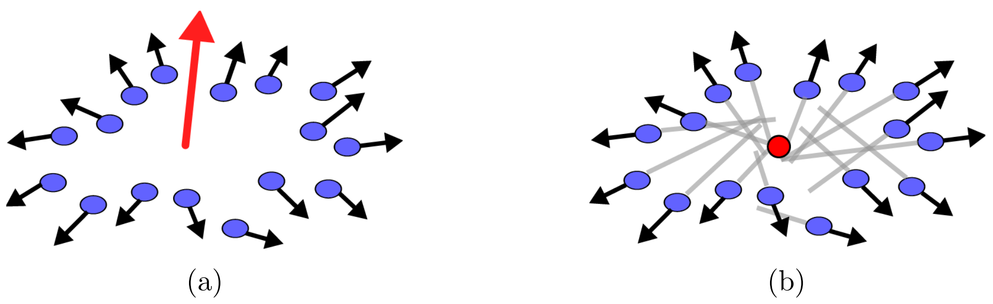


Figure 2.5: A ROSA point minimizes the variance in the angles with respect to the surface points in its local neighborhood (a) and the sum of squared distances to the line extensions of the normals of the neighborhood samples (b) [6].

However, since a plane might cut the shape in more than one parts (as in Figure 2.6), the distance by itself is not enough to determine the neighborhood. To overcome this, the distance is combined with orientation by using the Mahalanobis distance. The Mahalanobis distance is calculated between point pairs, and two points are considered as a pair if the distance between them is below a threshold.

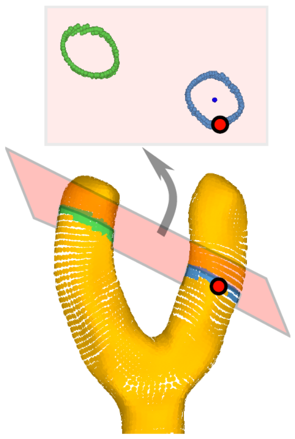


Figure 2.6: A plane cutting the object in more than one regions, where the red dot represents a point of interest $\mathbf{p_i}$ [6].

The optimality of the cutting plane states that the plane normal $\mathbf{v_i}$ minimizes the angle variance among the points within the local neighborhood N_i:

$$\mathbf{v_i}^{(t+1)} = \underset{\mathbf{v}\in\mathbb{R}^3, \|\mathbf{v}\|=1}{\operatorname{argmin}} \; var\{\langle \mathbf{v}, \mathbf{n}(\mathbf{p_j})\rangle : \mathbf{p_j} \in N_i^{(t)}\}, t \geq 0. \tag{2.13}$$

However, this is a complex non-linear problem, where t is the iteration number and $\mathbf{n}(\mathbf{p_j})$ is the unit normal of the point $\mathbf{p_j}$. The initial value $\mathbf{v_i}^0$ is selected randomly among the directions which are perpendicular to $\mathbf{p_i}$. This problem is solved as a quadratic minimization problem using singular value decomposition. However, for junction points, this is not a suitable solution since direction information around the joints is not reliable.

After the optimal plane and the relevant neighborhood are calculated, the ROSA points are created. This is done by finding points $\mathbf{p_i}$ that minimize the sum of squared distances between $\mathbf{p_i}$ and the normals of the points in N_i (Figure 2.5b):

$$\mathbf{r_i} = \underset{\mathbf{r}\in\mathbb{R}^3}{\operatorname{argmin}} \sum_{\mathbf{p_j}\in N_j} \|(\mathbf{r} - \mathbf{p_j}) \times \mathbf{n}(\mathbf{p_j})\|^2. \tag{2.14}$$

Similar to Equation 2.13, this is also a quadratic minimization problem. Here, it is solved by straightforward differentiation.

2.2.2 Junction Handling and Skeleton Building

The steps so far only build a skeleton point cloud, rather than a 1D connected curve. However, before building the skeleton, junction areas need to be processed further. As stated before, this comes from the fact that such regions do not have reliable orientation information required to find optimal cutting planes. Without extra processing, these areas yield chaotic-looking results (Figure 2.7a). To fix this, spatial coherence is enforced in these areas. This is done by building a correspondence between the skeletal samples and the original point cloud samples. Scattered skeletal points are smoothed using Laplacian smoothing which uses connectivity information provided by the Mahalanobis distances. This process also handles the noise caused by the unreliable orientation information and yields a region where the points are less scattered (Figure 2.7b).

Thinning and Junction Identification

Another sub-step is performed in order to further thin the skeletal cloud, since the steps so far do not yield a cloud thin enough for a 1D curve, especially around the joints. For this purpose, a 1D maximum length sequence (MLS) is applied by projecting the skeletal samples onto local best-fitting lines using principal component analysis (PCA).

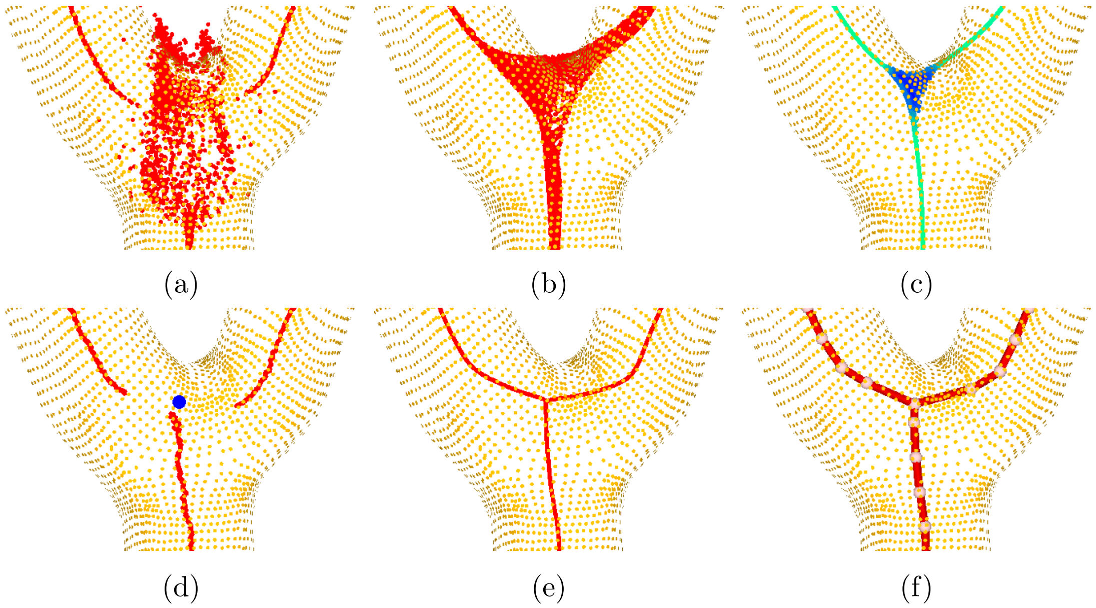


Figure 2.7: The steps of curve-skeleton extraction: ROSA point cloud (a), joint recovery (b), thinning (c), re-centering and joint collapsing (d), re-distribution of skeletal samples (e) and the final curve (f) [6].

For a skeletal sample $\mathbf{r_i}$, the following standard linearity measure is checked:

$$\psi(\mathbf{r_i}) = \lambda_i^{(1)}/(\lambda_i^{(1)} + \lambda_i^{(2)} + \lambda_i^{(3)}),$$

where $\lambda_i^{(1)}$, $\lambda_i^{(2)}$ and $\lambda_i^{(3)}$ are the three largest eigen values belonging to local PCA of $\mathbf{r_i}$. However, MLS is only applied to the samples which satisfy $\psi(\mathbf{r_i}) < \epsilon$, which means $\mathbf{r_i}$ is not a junction sample with a pre-determined constant threshold ϵ. The skeleton cloud before and after this part can be seen in Figure 2.7b and 2.7c, respectively.

Centering and Building the Curve Skeleton

After all the previous steps, some skeleton points are likely to go off-center (Figure 2.7c). In order to re-center the cloud, Equation 2.14 is solved again using the samples of the thinned skeleton cloud. While this process successfully re-centers the non-junction samples, the junction samples are gathered towards a single center point (Figure 2.7d). These junction samples are then re-distributed using a smoothing process (Figure 2.7e). At the end, all the skeletal samples are sub-sampled and then connected using line segments. This finalizes the curve-skeleton extraction (Figure 2.7f).

2.3 The Method of Cao et al.

Cao et al. propose a method which takes a point cloud (may be incomplete) as input and extracts a curve skeleton [5]. The purpose of this method is similar to that in [6]: reconstructing incomplete surfaces using curve skeletons.

The curve-skeleton extraction procedure proposed by this method uses the following steps (also shown in Figure 2.8):

- Laplacian-based contraction of the input point cloud.
- Building a connected skeleton graph from the contracted point cloud.
- Performing a series of edge collapses to extract a 1D curve.
- Re-centering the curve skeleton using the information of the original input.

This method is heavily based on the method of Au et al. [4], with improvements which make the method able to handle inputs that do not have connectivity information, i.e. point clouds.

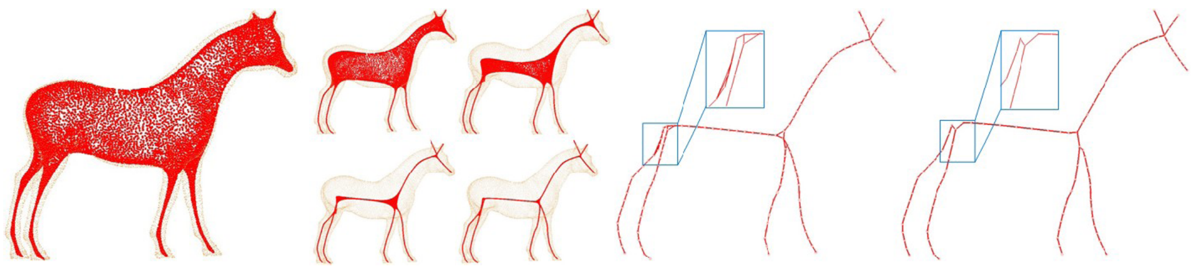


Figure 2.8: Overall view of the main steps and the end result [5].

2.3.1 Point-Cloud Contraction

As mentioned above, this method does not use connectivity information. Therefore, the steps and some parameters of the contraction process vary from [4].

The first important difference is building local neighborhoods to compensate the lack of connectivity information. For a point sample $\mathbf{p_i}$ in the input, first k nearest neighbors are found and put in the set $N_i(\mathbf{p_i})$. The value is selected as a small fraction of the total number of point samples, default value being $k = 0.012\#samples$. The value is selected small because the need is having enough neighbors to only construct a triangle fan. After finding the neighbors, a Delaunay triangulation is constructed. This triangulation is used to build the triangle fans and calculate the weights for the contraction, as given in Equation 2.4.

One problem with this approach is that while the point cloud gets more contracted after each iteration, it gets harder to calculate the neighborhood information. This issue is solved by calculating the neighborhood information only during the first iteration. In later iterations, only the weights are updated.

For the parameters of the contraction, further changes follow. In the original equation (Equation 2.4), parameter $W_{H,i}$ is updated using the total area of the triangle fan of the vertices. Here, $W_{H,i}$ is updated as: $W_{H,i}^{t+1} = W_{H,i}^{0} S_i^0 / S_i^t$, where S_i represents the neighborhood extent of point i and the superscript represents the iteration number.

The second parameter, W_L is updated the same way as the original method: $W_L^{t+1} = s_L W_L^t$, but they chose to use $s_L = 3.0$ as the initial value. Initial value W_L^0 is also changed to $W_L^0 = 0.2S^0$, where S^0 is the average neighborhood extent of all points in the input.

2.3.2 Topological Thinning

This step takes the contracted point cloud and extracts the final 1D curve. There are some intermediate processes to handle the lack of connectivity and the imperfections caused by the contraction step.

Building Connectivity

Since the points in the contracted point cloud do not have explicit connectivity, a connected graph needs to be built. This is done by sampling the contracted point cloud C using farthest point sampling with a radius of r. The sampling creates the graph G. Each $g_i \in G$ is a set of points C_i such that no subset shares a point with another subset C_j ($C_i \cap C_j = \emptyset, \forall i \neq j$). The union of all such subsets is the contracted point cloud itself, i.e. $C = \bigcup_i C_i$.

After the sampling, two graph samples g_i and g_j are connected if the points in their point cloud subsets share a local 1-ring neighbor. This process creates a connected structure suitable for the edge collapse.

Edge Collapsing

Unlike [4], here a simple edge cost mechanism is used: the Euclidean length of an edge. So, at each iteration, the edge with the shortest length, and the associated faces of this edge are removed. This continues until no faces remain in the structure.

Re-Centering

For centering purposes, a positional averaging similar to the embedding refinement of [4] is applied, since here also a skeleton-node-to-original-point-cloud mapping can be constructed during the connectivity-building step. However, the averaging applied here is simpler: a skeletal node is moved to the average of the positions of the points in the mapping, and no weights are used. The final result can be seen in the right-most part of Figure 2.8.

2.4 The Method of Telea and Jalba

Telea and Jalba propose a different curve-skeleton extraction compared to the other methods we have examined so far. Instead of starting the extraction process directly from the original surface, this method first extracts the surface skeleton and applies a different kind of contraction to shrink the mesh in volume [7]. What it fails to accomplish is, however, that the extracted curve skeletons are not exactly 1D. The end product is still a mesh, though it is very thin. Another shortcoming of the method is the loss of detail in the extremities (e.g. fingers of a hand) of the shapes. These parts of the objects are smoothed out only within a few iterations. Figure 2.9 shows the original mesh, the surface-skeleton mesh and the final curve skeleton. If we zoom in on the final outcome, we see the two main issues of this method: thickness of the structure (Figure 2.9d) and loss of detail (Figure 2.9e).

2.4.1 Contracting Surface-Skeleton Mesh

In this part, first the surface skeleton is calculated as a point cloud, together with the importance measure for the skeleton points. Later, the surface-skeleton mesh is reconstructed using the point cloud and the importance measure. Contraction is then

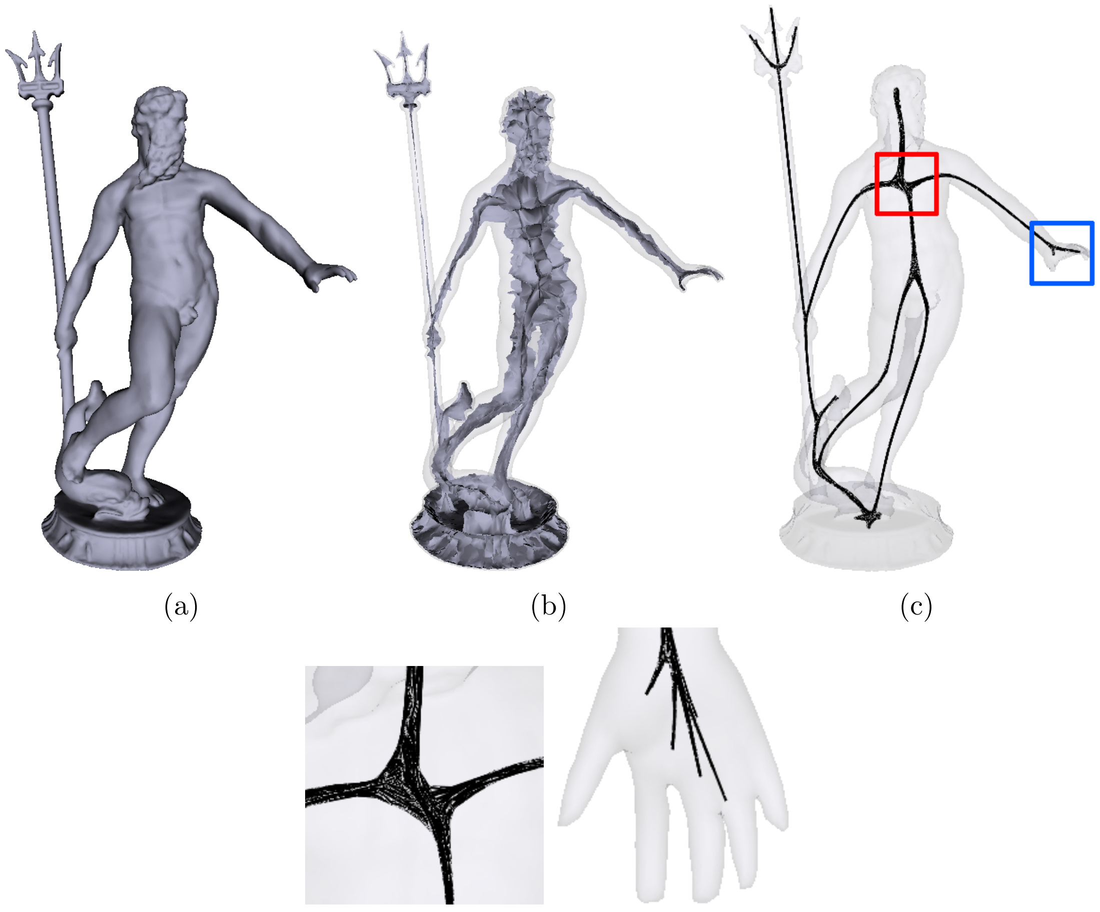


Figure 2.9: The original mesh (a), the surface-skeleton mesh (b), the extracted curve skeleton rendered in wireframe to show it is still a mesh (c) [7]. Zooming in on two main issues: Thickness, red frame (d) and loss of detail, blue frame (e). These are also rendered in wireframe, hence the white spaces.

applied to the surface-skeleton mesh, utilizing an advection scheme based on the gradient field of the importance measure.

The surface-skeleton point cloud is defined as the centers of the maximally inscribed balls within the surface. This method assigns to each surface-skeleton point an importance value based on the length of the shortest geodesic between two feature points of the skeleton point. In this scenario, long geodesics mean bigger importance.

Next, using the relationship between skeleton points and feature points, the surface-skeleton mesh is constructed. This reconstructed surface-skeleton mesh is the input given to the curve-skeleton extraction.

Below, the most important definitions are given.

Distance Transform

For any shape $\mathbf{\Omega} \subset \mathbb{R}^3$ which has boundary $\partial\mathbf{\Omega}$, distance transform is defined as:

$$DT_{\partial\mathbf{\Omega}}(\mathbf{x} \in \mathbf{\Omega}) = \min_{\mathbf{y} \in \partial\mathbf{\Omega}} \|\mathbf{x} - \mathbf{y}\|. \tag{2.15}$$

Surface Skeleton Definition

The points of the surface-skeleton point cloud satisfy:

$$\begin{aligned} \forall \mathbf{x} \in \mathbf{\Omega} \mid \exists \mathbf{y}, \mathbf{z} \in \partial\mathbf{\Omega}, \mathbf{y} \neq \mathbf{z}, \\ \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{x} - \mathbf{z}\| = DT_{\partial\mathbf{\Omega}}(\mathbf{x}). \end{aligned} \tag{2.16}$$

Thus, feature points $\mathbf{y}$ and $\mathbf{z}$ of a surface-skeleton point $\mathbf{x} \in \Omega$ are two points on the surface which are closest to $\mathbf{x}$.

Definition of Curve Skeleton

Using Equation 2.16 and the surface skeleton S, the curve skeleton $C(\mathbf{\Omega})$ is defined as:

$$\begin{aligned} C(\mathbf{\Omega}) =& \{\mathbf{x} \in S \mid \exists \mathbf{y}, \mathbf{z} \in \partial S, \mathbf{y} \neq \mathbf{z}, \\ & \|\mathbf{x} - \mathbf{y}\|_S = \|\mathbf{x} - \mathbf{z}\|_S = DT_{\partial S}(\mathbf{x})\}. \end{aligned} \tag{2.17}$$

The operator $\|\cdot\|_S$ is the geodesic distance metric on S, rather than the Euclidean distance. However, considering the unusual spiky shape of the surface-skeleton mesh, it is not feasible to calculate geodesic lengths, which are required to solve Equation 2.17. As a result, the curve skeleton is calculated by advecting the points on the outer boundary of the surface-skeleton mesh S within $\nabla DT_{\partial S}$. Termination occurs when the points reach the locus of singularities of $DT_{\partial S}$.

Calculating $\nabla DT_{\partial S}$

The tangent vectors of the geodesic between the two feature points of a surface skeleton point is used for this purpose. Basically, the average of these two tangent vectors points in the direction of $\nabla DT_{\partial S}$. Therefore, technically, there is no extra effort needed to calculate $\nabla DT_{\partial S}$.

Advection

Finally, the following equation is used to advect each point $\mathbf{s}$ on the surface-skeleton mesh:

$$\mathbf{s}^{i+1} = P_{T(\mathbf{s}^i)} \left(\mathbf{s}^i + \frac{\nabla DT_{\partial S}(\tilde{\mathbf{s}}^i)}{\|\nabla DT_{\partial S}(\tilde{\mathbf{s}}^i)\|} \delta \right), \tag{2.18}$$

where $\tilde{\mathbf{s}}^i$ is the closest point in the skeleton cloud to $\mathbf{s}$ and the superscript i denotes the iteration number. δ is calculated for each point and is equal to half of the average edge length belonging to the surface-skeleton mesh triangle fan around $\mathbf{s}$. The reason behind this weight is to provide a correspondence between the amount of advection and the local mesh density. In order to keep the advection on the surface skeleton, a projection operator $P_{T(\mathbf{s}^i)}$ is used. $T(\mathbf{s}^i)$ here is the triangle fan which the advected point is projected onto.

Chapter 3

Our Methods for Curve-Skeleton Extraction

Our process for the extraction overlaps in some parts with some of the methods detailed in Chapter 2. Yet, it differs in ideas and in application of the overall process. The method consists of two main parts: surface-skeleton extraction and curve-skeleton extraction. These parts have a total of five steps, as listed below and also shown in Figure 3.1.

1. Extracting the surface-skeleton point cloud from the original mesh.
2. Calculating a geodesic-length-based importance measure for all skeleton points.
3. Reconstructing a surface-skeleton mesh using the skeleton point cloud and the importance measure.
4. Contracting the reconstructed skeleton mesh using Laplacian-based contraction.
5. Extracting the curve skeleton by collapsing the edges of the contracted mesh until no faces remain.

Aside from the main method, we also present our modifications to the method of Telea and Jalba [7]. As described in Section 2.4, that method also uses the surface skeleton as an intermediate medium. Our improvements solve its shortcomings shown in Figures 2.9d and 2.9e.

In the following of this chapter, we first explain the details of our main method. At the end of the chapter, after we explain our main method, we discuss our improvements to Telea and Jalba's method.

3.1 Extracting Surface Skeletons

The purpose of this step is to construct a mesh for the surface skeleton. For this, first the surface-skeleton point cloud is calculated, and then an importance value is assigned to each point. Later, the vertices of the original mesh, surface-skeleton points and their importance values are used to construct the skeleton mesh.

Note that technically it is not required for our method to construct the surface-skeleton mesh in order to extract a curve skeleton, while other methods, for instance, the method of Telea and Jalba [7], have to. Since we are trying to improve curve-skeleton extraction by mesh contraction, we use the surface skeleton as an intermediate stage in the overall process. In Chapter 4, we discuss in detail if using the surface skeleton improves the quality of the curve skeletons. When the surface skeleton is not used, the process is, in terms of main ideas, similar to the method in [4].

3.1.1 Surface Skeleton as a Point Cloud

To extract the surface-skeleton point cloud, we use the so-called ball shrinking algorithm [10, 12]. This algorithm finds the maximally inscribed balls within the surface mesh and builds a collection of their centers. Together with these centers, the algorithm also records a pair of feature points for each skeleton point, where the ball touches the surface. A parameter of the algorithm can also allow these points to be very close to the ball (though not actually touching), in order to qualify as feature points. The feature points are two of the closest points on the surface to the ball center (there might be more than two). Formal definitions of the distance transform and the surface skeleton points for a mesh $\mathbf{\Omega}$ with the boundary $\partial\mathbf{\Omega}$ are given in Equation 3.1 and 3.2 [7].

$$DT_{\partial\mathbf{\Omega}}(\mathbf{x} \in \mathbf{\Omega}) = \min_{\mathbf{y}\in\partial\mathbf{\Omega}} \|\mathbf{x} - \mathbf{y}\|, \tag{3.1}$$

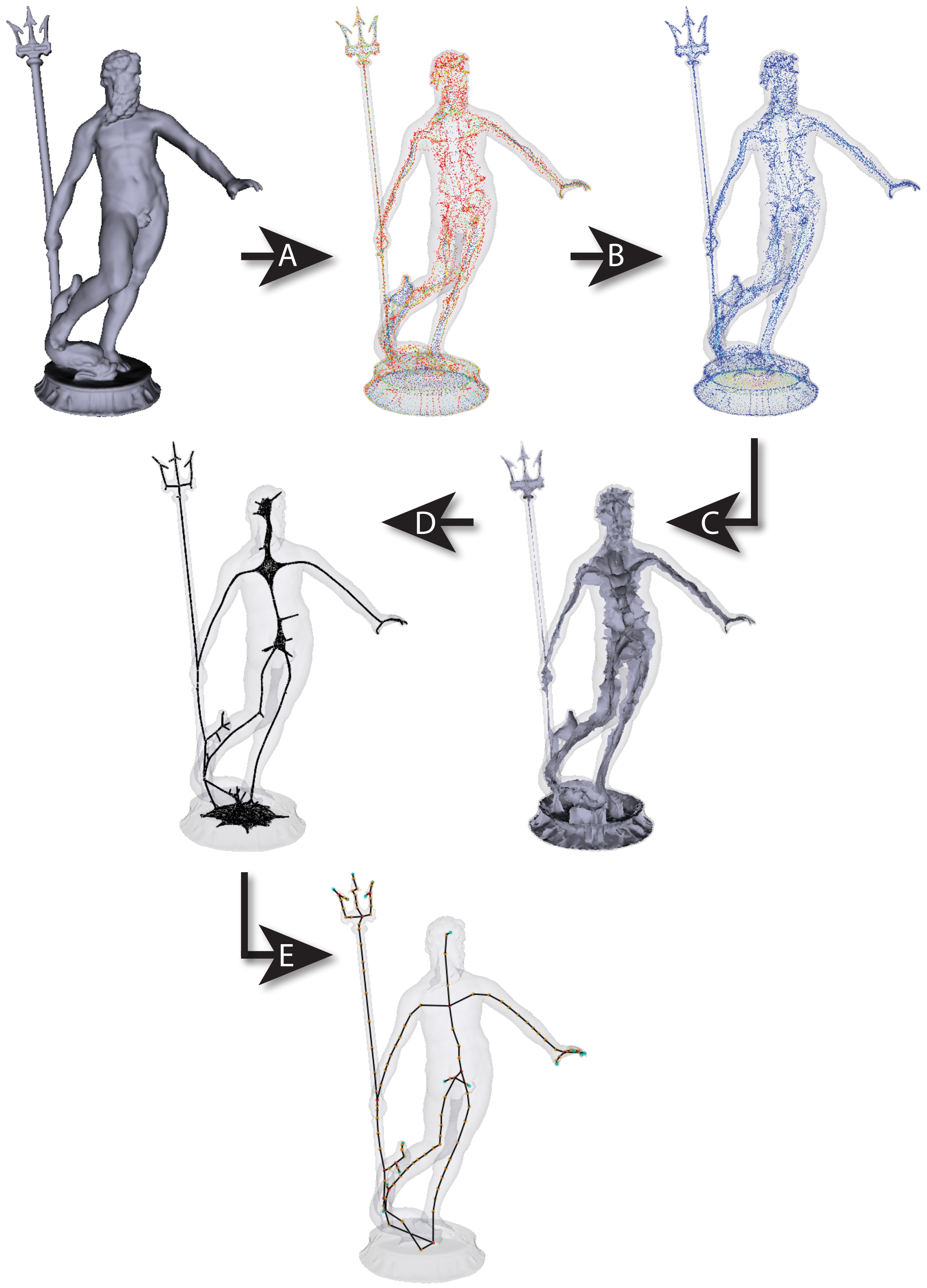


Figure 3.1: Overall view of the skeletonization steps: Surface-skeleton point cloud extraction (A), importance calculation (B), surface-skeleton mesh reconstruction (C), mesh contraction (D) and curve-skeleton extraction (E).

$$\begin{aligned}\forall \mathbf{x} \in \mathbf{\Omega} \mid \exists \mathbf{y}, \mathbf{z} \in \partial\mathbf{\Omega}, \mathbf{y} \neq \mathbf{z}, \\ \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{x} - \mathbf{z}\| = DT_{\partial\mathbf{\Omega}}(\mathbf{x}).\end{aligned} \tag{3.2}$$

Ball Shrinking Algorithm

Algorithm 1 shows the pseudo-code for finding one skeleton point as described by [12], with a visual description in Figure 3.2. Before running the algorithm, the original mesh is rescaled so that it fits a box with size of $\{2, 2, 2\}$, centered at the position $(0, 0, 0)$.

The summary of the algorithm flow is as follows (This procedure is applied to every point $\mathbf{p}$ on the mesh):

- First we start with a ball of radius 2 and also tangent to $\mathbf{p}$, which is for sure larger than or equal to the maximally-inscribed ball, after rescaling the point cloud. $\mathbf{p}$ is naturally the first feature point (lines 2-3).
- In the first loop starting at line 4, the radius is first shrunk by moving the ball center in the opposite direction of the skeleton-point normal. Additionally, a search for the second feature point, $\mathbf{f_2}$ is performed. The search parameter ϵ allows a small distance tolerance for the search.
- If $\mathbf{f_2}$ is not found, the search radius is updated (lines 13-14) and a new search is performed. The loop terminates either if $\mathbf{f_2}$ is found or the radius is shrunk too much.
- The second loop starting at line 16 is executed only if the radius is shrunk too much. At this point the radius is increased with smaller steps until $\mathbf{f_2}$ is found.
- The parameter τ at lines 7 and 16 stands for centeredness. Smaller τ values yield more accurate centeredness while requiring more shrinking steps. Therefore this parameter can be used to tune the trade-off between speed and precision.

This procedure tries to find one surface-skeleton point for each vertex in the mesh. Assume a skeleton point $\mathbf{s}$ and feature point $\mathbf{f_2}$ are found for the surface (and feature) point $\mathbf{f_1}$. It is possible that when the ball shrinking algorithm starts a skeleton-point search for $\mathbf{f_2}$, it can once more calculate $\mathbf{s}$ as a new skeleton point. Alternatively, a different skeleton point very close to $\mathbf{s}$ can be found. To overcome these, a skeleton point is kept only if its feature points were not used before for another skeleton point.

Algorithm 1 Ball shrinking algorithm [12]

```
1: procedure FindSkeletonPoint(Point $\mathbf{p}$)
2:     $r_L \leftarrow 2; r_R \leftarrow r_L$
3:     $\mathbf{f_1} \leftarrow \mathbf{p}$
4:     while (true) do
5:         $\mathbf{s} \leftarrow \mathbf{p} - r_L\mathbf{n}$                ▷ $\mathbf{n}$ is the surface normal of $\mathbf{p}$
6:         $\mathbf{f_2} \leftarrow search(\mathbf{s}, \epsilon r_L)$
7:         if $(|r_L - \|\mathbf{f_2} - \mathbf{s}\|| < \tau)$ then
8:             return $\{\mathbf{s}, \mathbf{f_1}, \mathbf{f_2}\}$                ▷ $\mathbf{f_2}$ is found
9:         end if
10:        if $(\mathbf{f_2} = \mathbf{p})$ then
11:            break                ▷ ball is shrunk too much
12:        end if
13:        $r_R \leftarrow r_L$
14:        $r_L \leftarrow -\frac{\|\mathbf{f_2}-\mathbf{p}\|^2}{2\mathbf{n}\cdot(\mathbf{f_2}-\mathbf{p})}$
15:    end while
16:    while $(r_R - r_L < \tau)$ do
17:        $r \leftarrow (r_R + r_L)/2$
18:        $dr \leftarrow r_R - r_L$
19:        $\mathbf{s} \leftarrow \mathbf{p} - r\mathbf{n}$
20:        $\mathbf{f_2} \leftarrow search(\mathbf{s}, \epsilon dr)$
21:        if $(\mathbf{f_2} = \mathbf{p})$ then
22:            $r_L \leftarrow r$
23:        else
24:            $r_R \leftarrow r$
25:        end if
26:    end while
27:    return $\{\mathbf{s}, \mathbf{f_1}, \mathbf{f_2}\}$
28: end procedure
```

Some skeleton points still can be very close to each other even though they have different feature points. Those points, which are in a very small neighborhood, are compacted with a post-processing step. The compaction operation performs a nearest-neighbor search with a small radius for each skeleton point and removes the ones which are indeed neighbors. The compaction step gets rid of the redundant skeleton points and provides a uniform-like distribution of the point cloud. But the trade-off is that the quality of the sampling may be reduced, i.e. a significant amount of points might be removed from the cloud.

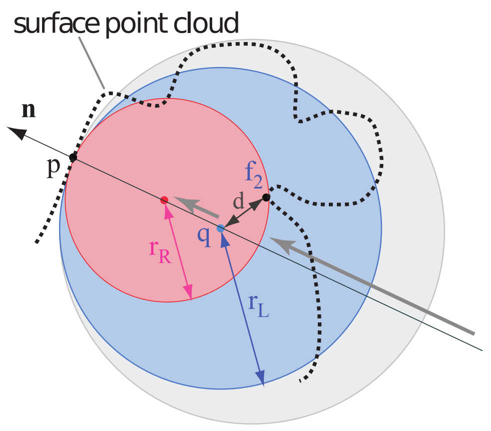


Figure 3.2: Depiction of ball shrinking given in Algorithm 1 [12]. **q** is the candidate for **s**. The dashed path represents the part of the surface and the grey arrows show the shrinking direction.

3.1.2 Geodesic-Length Based Importance Measure

In order to distinguish the skeleton points and to be able to compare them, each point in the skeleton is given an importance value. In our method, we use the importance for reconstructing the skeleton mesh. It can also be used for several other purposes, such as importance-based skeleton filtering.

The importance value we have chosen, similar to [7], is the length of the shortest geodesic between the two feature points of a surface-skeleton point (A geodesic is a minimum-length path on the surface that connects two surface points). This measure gives bigger importance to points which are closer to the "center" of the object. Thus, importance decreases from the center towards the surface. This is shown as a rainbow color map in Figure 3.3, where low and high importance values are mapped to blue and red, respectively.

For importance calculation, Jalba et. al claim that previous methods used to find geodesics (Dijkstra's algorithm, Fast Marching Method, distance field, etc.) are too costly or inaccurate. Instead, they present a more efficient method called *shortest, straightest geodesic (SSG)* [12].

In order to find SSGs, the method starts traces from feature point $\mathbf{f_1}$ in m different directions and tries to reach the other feature point $\mathbf{f_2}$. Tracing is done around the vector $\mathbf{f} = \mathbf{f_1} - \mathbf{f_2}$. Angles of the directions are $\alpha_i = (2\pi i)/m$ where $i \in \{1, 2, \ldots, m\}$.

During the trace, the mesh edges encountered are intersected with the plane defined by $\mathbf{f}$ and the path direction $\mathbf{d_i}$, i.e. the plane with normal $\mathbf{f} \times \mathbf{d_i}$. Then, the importance of the skeleton point is set to the length of the shortest path with the end points $\mathbf{f_1}$ and $\mathbf{f_2}$.

At the end, the importance (length of the SSG) is rainbow-color mapped: low importance is mapped to blue while high importance is mapped to red (Figure 3.3); note the importance distribution where the points closer to the "center" of the object have higher importance measure.

Although geodesics are only used for importance calculation during the extraction process, in Chapter 5 we propose to use them for surface parameterization as well.

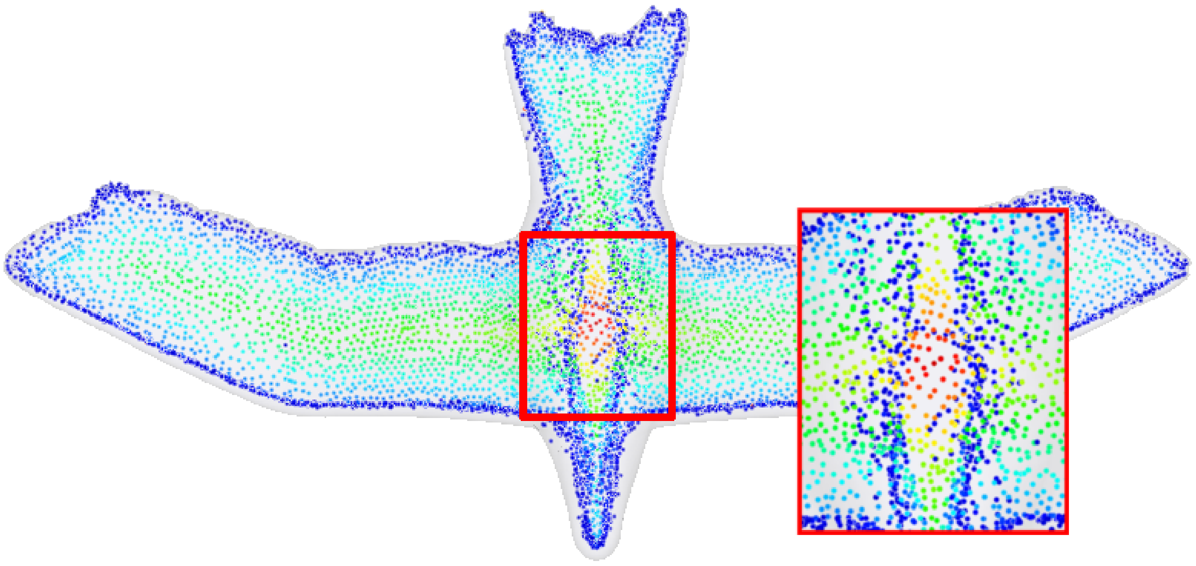


Figure 3.3: Skeleton points with color-mapped importance values. As shown in the zoom-in, the points in the "global center" of the object have the highest importance.

3.1.3 Reconstruction of Surface-Skeleton Mesh

After calculating the importance values for the skeleton points, it is trivial to reconstruct a skeleton mesh. Algorithm 2 describes the reconstruction procedure as first used by Telea and Jalba in [7].

The algorithm iterates over the faces of the original mesh M (first loop starting at line 3) and does the following:

- For face f of the original mesh, we look at the three vertices that define it (the loop starting at line 5).

Algorithm 2 Reconstruction algorithm for the surface-skeleton mesh

```
1: procedure ReconstructSkelSurface(Mesh $M$, Skeleton $S$)
2:     SkelMesh $M_S$;
3:     for all Face $f \in M$ do
4:         SkelFace $f_S$;
5:         for $i = 1 \to 3$ do
6:             $f_S.\mathbf{v_i} \leftarrow$ MatchingSkelPoint($f.\mathbf{v_i}$)    ▷ Skeleton point with maximum
   importance and has $f.\mathbf{v_i}$ as feature point
7:         end for
8:         $M_S \leftarrow M_S \cup f_S$
9:     end for
10:    return $M_S$
11: end procedure
```

- For each vertex $\mathbf{v_i}$ of f, we find the skeleton point with maximum importance and $\mathbf{v_i}$ as feature point. This information is kept during the extraction of the skeleton point cloud as *inverse feature transform* (a one-to-many mapping that maps the surface-skeleton points to the original mesh vertices). The vertex that *MatchingSkelPoint* returns at line 6 is found using the inverse feature transform and the following equation [7]:

$$\mathbf{v_i}^{f_S} = \operatorname*{argmax}_{\mathbf{s} \in S | \mathbf{f_1^s} = \mathbf{s}(\mathbf{v_i}^f) \vee \mathbf{f_2^s} = \mathbf{s}(\mathbf{v_i}^f)} \rho(\mathbf{s}), \tag{3.3}$$

  where $\mathbf{s}$ is a skeleton point with importance value $\rho(\mathbf{s})$ and feature points $\mathbf{f_1^s}$ and $\mathbf{f_2^s}$. $\mathbf{v_i}^f$ represents the ith vertex of f where $i \in \{1, 2, 3\}$.

- A new skeleton-mesh face f_S is created using the three such skeleton points. In the unlikely case that a surface vertex has no corresponding skeleton point, no new face is created.

- The resulting skeleton mesh M_S is the union of all faces created using the previous steps and the skeleton points which define those faces.

3.2 Extracting Curve Skeletons

This step is partly based on the method presented by Au et al. [4], with differences in application. The biggest difference is the starting point, since we begin with the surface-skeleton mesh, rather than with the input one. With this, we aim to get a curve skeleton which is smoother, has more details and is better centered.

Other differences include a small modification introduced to the cost scheme for edge-collapse procedure and a vastly different post-processing part. The three steps (two main and one post-processing) for extracting the curve skeletons are as follows:

1. Mesh contraction by Laplacian-based smoothing (see Section 2.1.1).
2. Series of edge collapses to reduce the contracted mesh to a set of 1D curves.
3. Post-processing of the curve skeleton: removing unnecessary edges and re-centering the nodes.

3.2.1 Mesh Contraction

For this step, we use the method developed by Au et.al. [4]. As stated before, the difference at this point is the type of surface which the contraction is applied to.

A difference related to this change, we have observed, is the amount of detail preserved after the same number of iterations. Another visible difference is the amount of displacement for the vertices. When the original surface is contracted directly, vertices drift farther away from the local center of the shape region. There are two reasons for these differences. The first one is the spiky nature of the surface-skeleton mesh (e.g. Figure 3.1, C), which causes stronger attraction constraints and can hold the details in-place easier. The second reason is the natural centeredness of the surface-skeleton points. These points are already in their local centers and together with a stronger attraction constraint, they are pulled less off-center.

For comparison purposes, the results after 5, 10 and 15 iterations are shown for the surface-skeleton mesh (Figure 3.4 a, b, c) and for the original mesh (Figure 3.4 d, e, f). Images of 10$^{\text{th}}$ and 15$^{\text{th}}$ iterations are rendered in wireframe mode to make thin parts more visible. If we compare the meshes after the 15$^{\text{th}}$ iteration, we see that the contracted surface skeleton is thinner, yet it has more details (e.g. the head of the frog), even though the number of iterations is the same for both meshes. A further discussion about the difference between two approaches follows in Chapter 4.

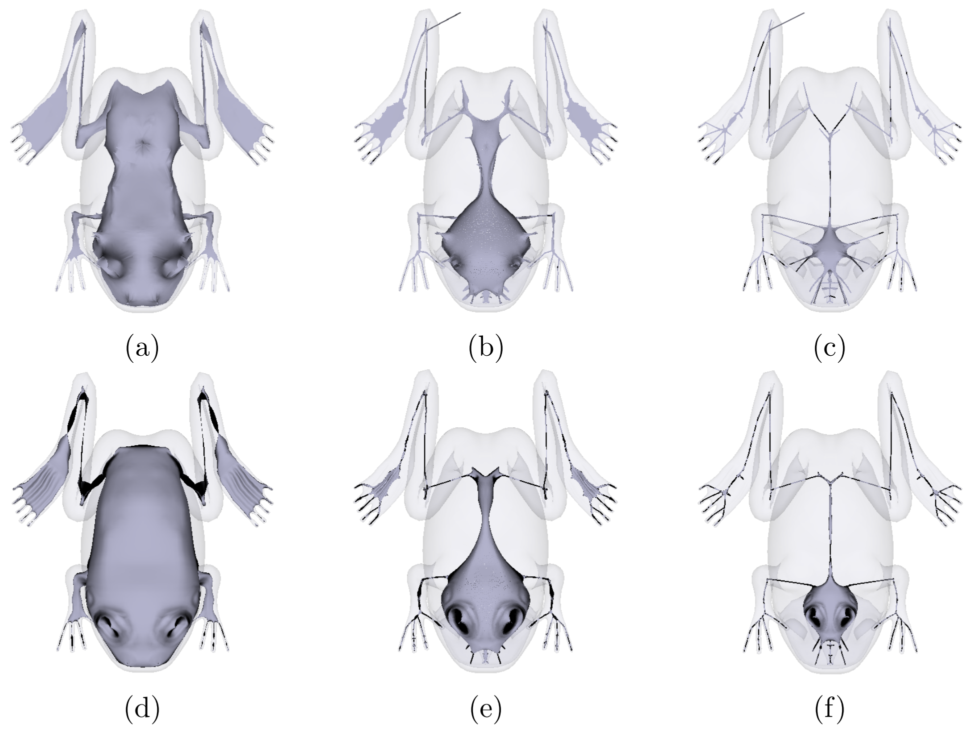


Figure 3.4: Snapshots from different iterations of the contraction step. Surface-skeleton mesh after 5th(a), 10th(b) and 15th(c) iterations. Original mesh after 5th(d), 10th(e) and 15th(f) iterations

3.2.2 Edge Collapsing

The contraction process yields a mesh which has a very small volume and a total area of faces close to zero. Yet, it is still a mesh and in some regions (e.g. bulky parts and branch junctions, Figure 3.4f) it is thicker than desired. The aim of this step is to remove all the faces by a series of edge collapses and leave only a set of 1D curves. While working towards this aim, it is important to:

- Disrupt the shape as little as possible
- Preserve the topology
- Keep a decent sampling of skeleton nodes.

For these purposes, we use the cost mechanism of Au et al. [4] and remove the edge with the smallest cost at each iteration. An edge removal is a simple half-edge collapse. This process terminates when the mesh becomes a set of connected 1D curves, i.e. after no faces remain.

Overall Description of the Process

Here we first give the algorithms in pseudo-code for the overall process (Algorithm 3), and the two important subroutines for calculating the edge costs and updating them after an edge collapse (Algorithms 4 and 5).

Algorithm 3 Overall edge collapsing process

```
1:  procedure CollapseEdges($V, E, F$)
2:      MinHeap $heap$;                                   ▷ Minimum-heap
3:      for all Edge $e \in E$ do                         ▷ Initial costs
4:          $cost \leftarrow CalcCost(e)$
5:          $heap.insert(e, cost)$
6:      end for
7:      while $F.size > 0$ do                             ▷ Until all the faces are gone
8:          $candidate \leftarrow heap.pop()$              ▷ Get the edge with minimum cost
9:          $EdgeCollapse(candidate)$
10:         $UpdateCosts(candidate, heap)$
11:     end while
12: end procedure
```

Algorithm 4 Calculating the cost of an edge

```
1:  procedure CalcCost($e(\mathbf{v_i}, \mathbf{v_j})$)
2:      $\mathbf{p_i} \leftarrow \{v_{i,x}, v_{i,y}, v_{i,z}, 1\}$             ▷ Convert to homogeneous form
3:      $\mathbf{p_j} \leftarrow \{v_{j,x}, v_{j,y}, v_{j,z}, 1\}$
4:      $Q_{\mathbf{v_i}} \leftarrow calcQ(\mathbf{p_i}), \quad Q_{\mathbf{v_j}} \leftarrow calcQ(\mathbf{p_j})$   ▷ See Equation 2.6 for Q calculation
5:      $shapeCost \leftarrow \mathbf{p_j}^T Q_{\mathbf{v_i}} \mathbf{p_j} + \mathbf{p_j}^T Q_{\mathbf{v_j}} \mathbf{p_j}$
6:      $sampCost \leftarrow 0$                                              ▷ Start calculating sampling cost
7:      for all Edge $e' \in \mathbf{v_i}.edges$ do
8:          $sampCost \leftarrow sampCost + \|e'\|$
9:      end for
10:     $sampCost \leftarrow sampCost * \|e\|$
11:     return ($\alpha\ shapeCost + \beta\ sampCost$ + BOUNDARY_COST )
12: end procedure
```

Algorithm 5 Updating costs after an edge collapse

```
1: procedure UpdateCosts(Edge $c(\mathbf{v_i}, \mathbf{v_j})$, MinHeap $heap$)
2:     for all Edge $c'(\mathbf{v_i'}, \mathbf{v_j'}) \in (\mathbf{v_j}.edges - c)$ do        ▷ Update $\mathbf{v_j}$ edges
3:         $heap.update(c', CalcCost(c'))$
4:         Vertex $\mathbf{t} \leftarrow \mathbf{v_i'} = \mathbf{v_j}$ ? $\mathbf{v_j'}$ : $\mathbf{v_i'}$
5:         for all Edge $c^* \in (\mathbf{t}.edges - c')$ do        ▷ And the neighbors
6:             $heap.update(c^*, CalcCost(c^*))$
7:         end for
8:     end for
9: end procedure
```

Here are some notes about the overall process:

- V, E and F in Algorithm 3 are the sets of vertices, edges and faces in the contracted skeleton mesh, respectively.
- We keep the edges in a min-heap ordered by their costs. Choosing the candidate edge is therefore simply popping the top element of the heap (Algorithm 3, line 8).
- After a removal, the edges which need cost update are: the edges of $\mathbf{v_i}$ and $\mathbf{v_j}$ plus the neighbors of such edges. Therefore only a small part of the heap is modified at each iteration (Algorithm 5). Note that the algorithm iterates only over the edges of $\mathbf{v_j}$, since during the collapse $\mathbf{v_j}$ inherits all edges of $\mathbf{v_i}$.

Half-Edge Collapse

For an edge e with the vertex pair $(\mathbf{v_i}, \mathbf{v_j})$, the edge collapse $i \rightarrow j$ happens as follows (visual depiction in Figure 3.5):

- All faces containing e (shown in red), are removed (in this example f_0 and f_1).
- The edges incident to $\mathbf{v_i}$, which were part of the removed faces are also removed (green edges). If these edges have other remaining faces, the edges of $\mathbf{v_j}$ which are part of the removed faces (blue edges) take the places of the removed (green) edges.
- $\mathbf{v_i}$ is removed and $\mathbf{v_j}$ inherits all its remaining edges.
- The position of $\mathbf{v_j}$ is not changed.

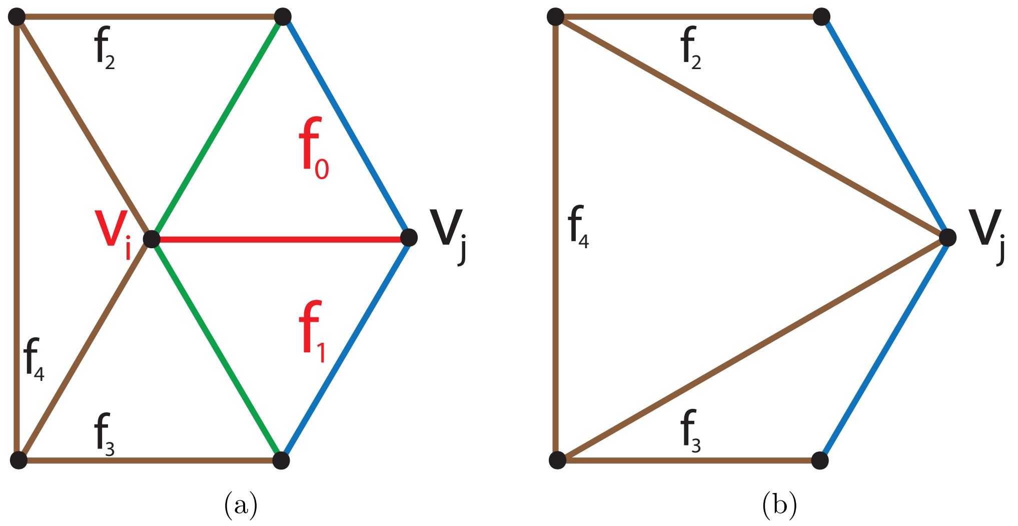


Figure 3.5: Example case before (a) and after (b) one half-edge collapse. See text for the details.

Calculating Edge Costs

For this purpose, we use the same two costs used in [4], as well as another cost that we have added to punish possible occurrences of long edges more harshly. This new cost is called *boundary cost* (S_3) which is a large constant value added to the total cost of the edge in case the edge has lost all of its faces. We consider such an edge to already be a part of a curve skeleton, thus we try to prevent its removal. But we do not want to disable the removal of faceless edges completely, as they may still be removed in order to refine the curve skeleton. If the edge still has faces, boundary cost is set to zero. Thus,

$$S_3 = \begin{cases} 3 \times |V|, & \text{if edge has no faces} \\ 0, & \text{otherwise,} \end{cases} \tag{3.4}$$

where $|V|$ is the number of vertices in the contracted mesh. With this addition, Equation 2.9 becomes:

$$C(i,j) = \alpha S_1(i,j) + \beta S_2(i,j) + S_3(i,j). \tag{3.5}$$

As explained in Section 2.1.2, the cumulative update of the error matrices of the vertices (Equation 2.10) causes the shape cost to dominate over time the sampling cost. This is another reason why we introduced the boundary cost. Sometimes an edge may end up with a small error matrix even though it has no faces left. We protect such edges to keep the curve skeleton detailed and fairly sampled.

As for the weights α and β, we observed that they have less significant effect in our application, compared to the claims of Au et. al in [4]. We sometimes use larger values for β to get slightly better results compared to smaller values, depending on the mesh. For the boundary cost, we simply use the value $3|V|$, which makes it proportional to the mesh size. In Section 4.3.1, we examine the effects of the cost types and different weights in detail.

Alternative Cost Experiment: In the earlier stages, we have considered adding another cost, based on the geodesic importance measure. The idea was to exploit the fact that surface-skeleton points close to the "center" have bigger importance. We have found k-nearest surface-skeleton points for each curve-skeleton node and assigned the average importance as the fourth element to Equation 3.5. This was hoped to improve the centeredness during the edge-collapse step, therefore reducing the need for an extra re-centering step. Even though this *importance cost* did preserve the edges close to the center, almost all edges representing the branch details were removed. We have realized that the importance is highest in the global center of the shape and local centers have high importance only in a local sense (recall Figure 3.3). Therefore, the surface-skeleton importance is not employed in the costing scheme.

3.2.3 Post Processing

After all the faces were removed from the contracted mesh, what we obtain is a set of connected edges. This curve skeleton is also still homotopic to the original surface. However, this structure is not perfect after only performing the edge-collapse step. Therefore we need some extra post-processing steps after collapsing is done. The issues we try to fix and their solutions follow below.

Removing Duplicate Edges

After several iterations of edge collapse, some edges with same vertex pairs may occur. This is a coincidental situation because of the re-assigning of vertex-edge relationships during the edge collapses. After all the collapsing is done, we simply remove all edge duplicates.

Unless otherwise noted, the color scheme used in our images for the curve-skeleton nodes is as follows: junction nodes are rendered in red while tip nodes and regular nodes are rendered in cyan and yellow, respectively. There are lots of red nodes in Figure 3.6, which is also a sign for duplicate edges. A vertex incident to multiple duplicate edges is rendered in red, even though there is only one edge visible.

Reviving Dead Vertices

During the edge collapse, if it happens that one of the duplicate edges are picked for removal, one vertex will "die" when the collapse step removes it. But since the duplicate of the removed edge is still present, this leaves it with only one alive vertex. This does not cause any procedural problems, but when the final outcome is rendered, it looks odd (Figure 3.6b). If an alive edge has one or two dead vertices, we revive such vertices.

Cleaning Unwanted Extremities

Our aim is to keep as much detail as possible in the curve skeleton. But when the parameters are tuned for this purpose (especially the attraction constraint for the contraction step), the process may leave noisy details behind. These are small branches which are sticking out in places where they are not needed. We apply a parameterized cleaning method to get rid of such edges; compare Figure 3.6 and 3.7. The pseudo-code for the cleaning process is given in Algorithm 6.

In the curve skeleton, a tip node has only one edge while a regular node has two edges and a junction node has three or more edges. Starting from a tip node, we move towards a junction node by traversing edges. Our threshold here is a pre-defined length and the default value is the average length of all skeletal edges. When we traverse the edges, we keep track of the distance we have traversed. If we reach a junction before we reach the length threshold, we consider the path we have traversed as noise and remove its edges and vertices. If we reach the threshold before reaching a junction, that means the path we are traversing is an actual branch of the skeleton and no change is made.

The threshold can also be tuned accordingly. Normally, the average length is a good choice. If the curve skeleton has mostly short or mostly long edges, the parameter *factor* of Algorithm 6 can be set to values smaller or larger than 1 to prevent the removal of

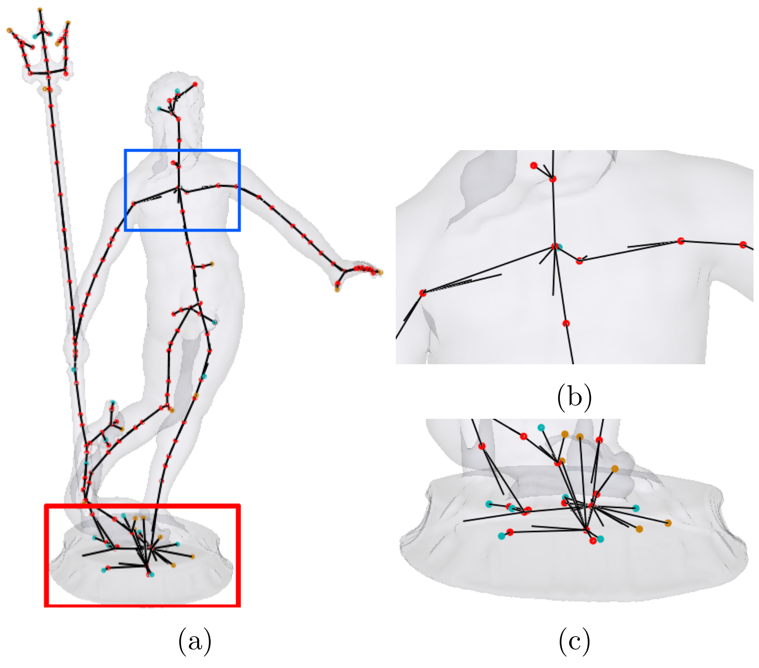


Figure 3.6: Curve skeleton without any post-processing (a). Dead vertices and unwanted edges are more visible in the zoomed part (b). In bulky areas, curve skeleton is not well-defined and this yields ugly spikes (c).

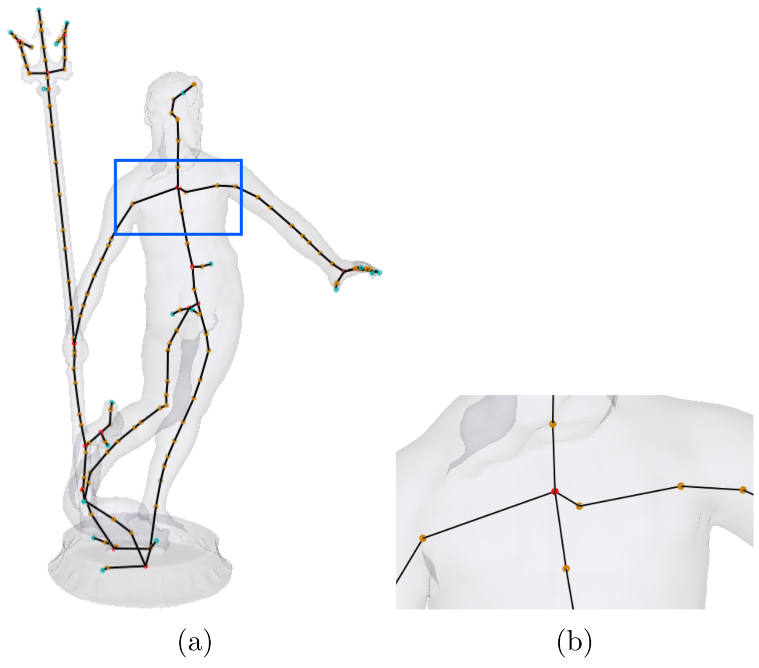


Figure 3.7: Curve skeleton after cleaning (a). Zoomed in section (b)

Algorithm 6 Cleaning unwanted extremities

```
 1: procedure CleanExtremities($E, V, factor$)
 2:     $threshold \leftarrow factor * AvgEdgeLength(E)$
 3:     for all Vertex $\mathbf{v} \in V$ do
 4:         if $\mathbf{v}.edges.size = 1$ then                              ▷ Tip node
 5:             $pathE \leftarrow \{\ \}, \quad pathV \leftarrow \{\ \}$
 6:             $dist \leftarrow 0$
 7:             Vertex $\mathbf{t} \leftarrow \mathbf{v}$
 8:             Edge $e(\mathbf{v_i}, \mathbf{v_j}) \leftarrow \mathbf{t}.edges[0]$
 9:             $pathV.add(\mathbf{t})$
10:             while true do
11:                 $pathE.add(e)$
12:                 $\mathbf{t} \leftarrow \mathbf{v_i} = \mathbf{t}\ ?\ \mathbf{v_j} : \mathbf{v_i}$
13:                 $dist += \|e\|$
14:                 if $\mathbf{t}.edges.size > 2 \vee dist >= threshold$ then
15:                     break                            ▷ Reached joint or threshold
16:                 end if
17:                 $pathV.add(\mathbf{t})$
18:                 $e \leftarrow e = \mathbf{t}.edges[0]\ ?\ \mathbf{t}.edges[1] : \mathbf{t}.edges[0]$      ▷ Next edge
19:             end while
20:
21:             if $\mathbf{t}.edges.size > 2$ then          ▷ Reached joint before threshold
22:                 for all Vertex $\mathbf{v}' \in pathV$ do
23:                     $\mathbf{t}.inherit(\mathbf{v}'.edges)$            ▷ $\mathbf{t}$ inherits the edges of $\mathbf{v}'$
24:                     $remove(\mathbf{v}')$
25:                 end for
26:                 for all Edge $e' \in pathE$ do
27:                     $remove(e')$
28:                 end for
29:             end if
30:         end if
31:     end for
32: end procedure
```

actual details, or make a harsher cleaning.

Re-centering the Skeleton

The mesh-contraction process may move the vertices off-center no matter how good the parameters are tuned. Also, the edge-collapse procedure does not try to find an optimal position for the remaining vertex after the edge is collapsed, unlike [11]. Thus, after all post-processing is done, we re-center the skeleton using the coordinate information of the original mesh.

Similar to the re-centering method in [4], we keep a skeleton-node-to-mesh mapping $\Pi_{\mathbf{v}}$ for each curve-skeleton node $\mathbf{v}$ to keep a reference to the vertices which are collapsed onto it. We also have a mapping between the skeleton mesh and the original mesh to determine the relationship between their vertices. Using the composed $\Pi_{\mathbf{v}}$ mapping, we calculate a new position for each skeleton node $\mathbf{v}$ by a weighted average using the coordinates of the related vertices, i.e. the vertices in $\Pi_{\mathbf{v}}$.

The reason for using a weighted average can be understood when the node to mesh mapping is visualized, see Figure 3.8: Often one side of the mapping (which resembles a cylinder with no top and bottom) has more vertices than the other side. We chose the weighting scheme in such a way that the side with less vertices has comparable influence to the other side. Thus, our weight for each vertex in $\Pi_{\mathbf{v}}$ is the average area of the faces in the triangle fan of that vertex (Larger face areas mean sparser distribution of vertices). Based on these, the steps of the process are given in Algorithm 7. The skeleton after re-centering can be seen in Figure 3.9.

Lastly, a side-by-side comparison after each post-processing step is shown in Figure 3.10.

Algorithm 7 Re-centering of a skeleton node

```
1: procedure RecenterCurveSkel($V$)
2:     for all Vertex $\mathbf{v} \in V$ do
3:         $\Pi'_{\mathbf{v}} \leftarrow \mathbf{v} \cup \Pi_{\mathbf{v}}$
4:         $totalWeight \leftarrow 0$
5:         $\mathbf{v}' \leftarrow \{0, 0, 0\}$
6:         for all Vertex $\mathbf{v}^* \in \Pi'_{\mathbf{v}}$ do
7:             $w \leftarrow AvgAreaFaces(\mathbf{v}^*)$
8:             $\mathbf{v}' \leftarrow \mathbf{v}' + w * \mathbf{v}^*$
9:             $totalWeight = totalWeight + w$
10:        end for
11:        $\mathbf{v} \leftarrow \mathbf{v}' / \ totalWeight$
12:    end for
13: end procedure
```

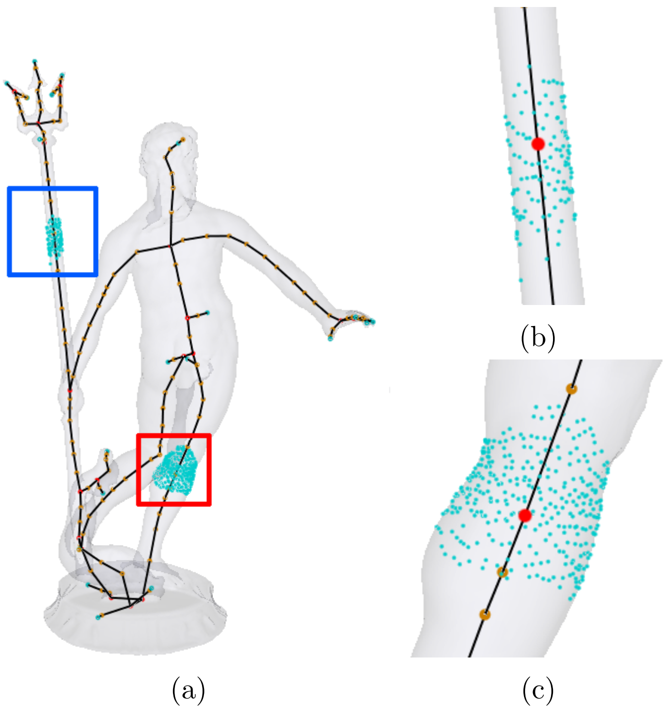


Figure 3.8: Skeleton-node-to-mesh mapping examples (a). Detail showing the skeleton node in red and the surface vertices in cyan (b, c).

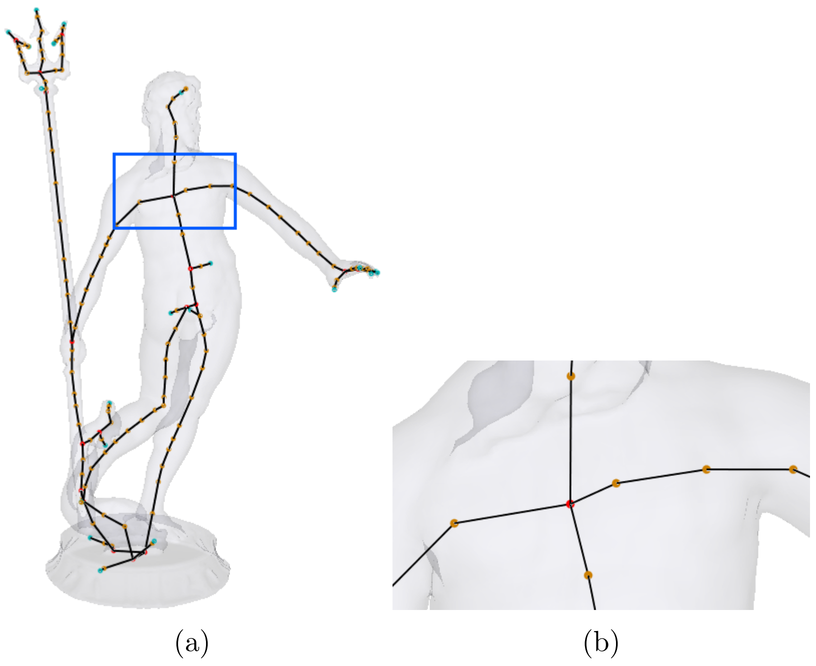


Figure 3.9: Curve skeleton after re-centering (a). Zoomed in section (b)

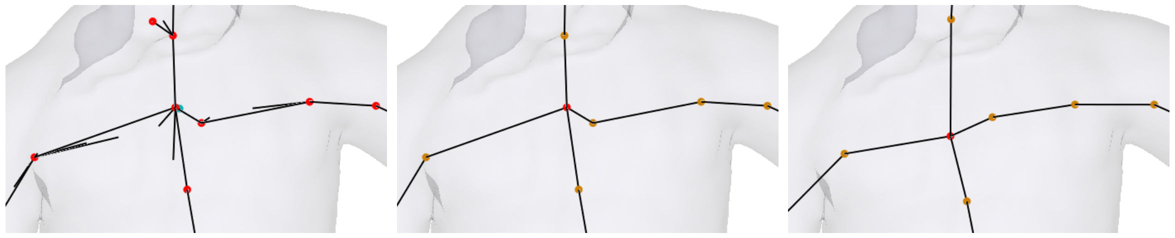


Figure 3.10: Side-by-side comparison of the steps. Initial state (a), after dead vertex revival and cleaning (b), after re-centering (c).

Alternative Re-centering Experiments: Before the method which is based on face-area-weighted averaging, we have experimented with different re-centering approaches.

Initially, re-centering was done only using a regular averaging using a mapping between the curve-skeleton node $\mathbf{v}$ and the contracted surface-skeleton mesh M_S. It was quickly realized that this is not effective since the contraction step already moves the vertices off-center and using the contracted mesh is not a good choice.

Later, we have tried an importance-based centering, using the gradient vector field of the importance measure. A curve skeleton node was moved iteratively along this vector field, since this was supposed to center the node, as the vectors were pointing towards the "center" of the object. But his method brought the nodes to the global center of the shape, rather than the local centers. This was also very sensitive to the sampling of surface-skeleton nodes, since we determined the closest vectors using a nearest-neighbor search. But the main problem was that this approach was unable to recognize local centers, only moving nodes towards the global one, hence causing loss of detail (similar to the case in [7]).

All in all, we have decided the re-centering can be done using a geometrical, weighted averaging, and at the end, decided on the face-area-weighted scheme.

3.3 Improving the Method of Telea and Jalba [7]

Telea and Jalba propose a very effective method for extracting curve skeletons in terms of centeredness and smoothness. However, the method also has two issues that reduce the quality of the skeletons:

1. The final outcome of the method is not a 1D skeleton, but a thin mesh. This mesh is thicker in the junctions of the shape, where two or more branches join. This also limits the usage of the curve skeleton for many purposes, for instance skeletal animation.

2. Small details of the shape are smoothed out. This makes the method completely insensitive to noise, but at the same time, important small details such as fingers and horns are not present in the curve skeleton.

In order to improve the method and overcome its issues, we have made the following additions:

- First, we have added a Mean-shift clustering step to the contraction part. Although this does not solve either of the problems, it yields a thinner and more compact mesh in less number of iterations. This feature however, reduces the effectiveness of the constraints for detail preserving.

- Then, to preserve the small details, we have introduced positional constraints for the contraction step.

- Finally, to convert the thin mesh into a 1D curve skeleton, we have simply added the edge-collapse and post-processing steps of our method. Details of these steps are described in Section 3.2.2 and 3.2.3.

In the following of this section, we explain the new additions (Mean-shift and positional constraint) in detail.

3.3.1 Mean-Shift Clustering

During the contraction, we employ a simple version of mean-shift clustering to cluster vertices together and accelerate the process. By this, we get a thinner mesh in fewer iterations. Instead of a large number of consecutive contraction iterations, we divide the operation into smaller set of iterations and invoke clustering in between these sets.

In order to cluster the vertices, we first find at most k nearest neighbors of the vertex within a fixed radius ϵ. Then the vertex is moved to the average of the positions of the neighbors and the vertex itself. The pseudo-code for this iterative process is given in Algorithm 8. The parameters V, k, ϵ and N represent the set of vertices, number of neighbors, search radius and number of iterations, respectively. Unless otherwise noted, we use $k = 300, \epsilon = 0.03$ and $N = 3$.

Algorithm 8 Simple mean-shift clustering

```
1: procedure MeanShift($V, k, \epsilon, N$)
2:     for $i = 1 \to N$ do                                  ▷ Iterate $N$ times
3:         for all Vertex $\mathbf{v} \in V$ do
4:             $neighbors \leftarrow search(\mathbf{v}, k, \epsilon)$
5:             $\mathbf{p} \leftarrow \mathbf{v}$
6:             for all Vertex $\tilde{\mathbf{v}} \in neighbors$ do
7:                 $\mathbf{p} \leftarrow \mathbf{p} + \tilde{\mathbf{v}}$
8:             end for
9:             $\mathbf{v} \leftarrow \frac{\mathbf{p}}{neighbors.size + 1}$
10:        end for
11:    end for
12: end procedure
```

The effect of this algorithm can be seen in Figure 3.11. The skeleton is rendered in wireframe mode to make the effect of clustering more visible. The result without applying mean-shift is also shown for comparison.

3.3.2 Positional Constraints for the Contraction Step

The original contraction moves the vertices in the gradient field of the importance of the surface-skeleton points, which is similar to the measure described in Section 3.1.2. Since this is a global scheme, importance is bigger in the centers of larger parts of a shape. This causes the vertices in smaller parts to be moved towards the global centers, hence the curve skeleton loses these details (Figure 3.12).

Without enabling the constraint, the simplified flow of the contraction step is as given in Algorithm 9. The method *evalAdvectionStep* at line 5 performs the advection based on Equation 2.18. To keep vertices corresponding to small details in their places, we have introduced a positional constraint weight W_P, related to the attraction constraint W_H of Au et.al. [4]. The similarity comes from the fact that our constraint, too, is based on the one-ring face areas of the vertices.

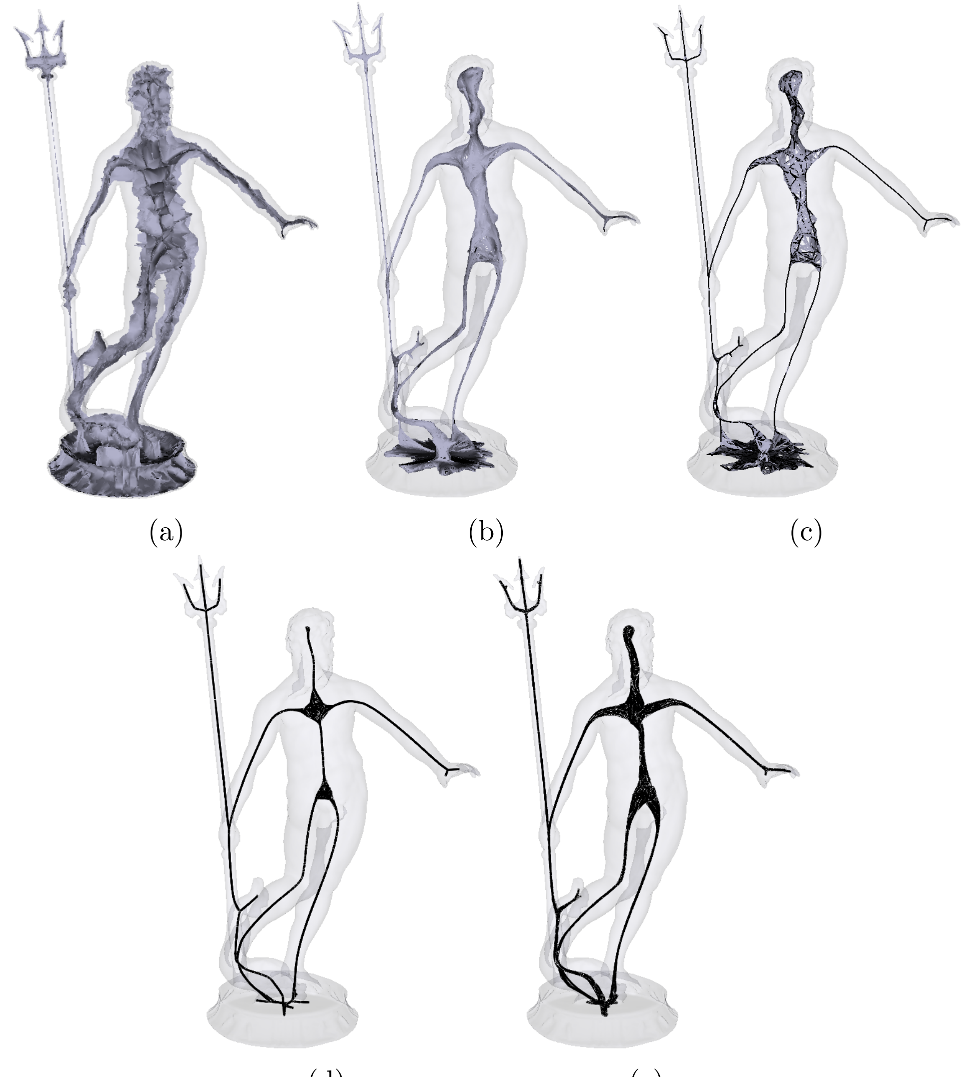


Figure 3.11: Skeleton mesh (a). The mesh after 30 contraction iterations (b). The mesh after 30 contraction + 3 mean-shift iterations (c). The result after 4 sets of 30+3 iterations (d). The result of 120 contraction iterations, without mean-shift (e).

We calculate the initial value of $W_{P,\mathbf{v}}$ for vertex $\mathbf{v}$ of the skeleton mesh as follows.

$$W_{P,v} = \begin{cases} \dfrac{A_{(0),\mathbf{v}}}{A_{(s),\mathbf{v}}}, & \text{if } \frac{A_{(0),\mathbf{v}}}{A_{(s),\mathbf{v}}} \geq 1 \\ 1, & \text{otherwise} \end{cases}, \tag{3.6}$$

where $A_{(0),\mathbf{v}}$ is the total one-ring face area of vertex $\mathbf{v}$ on the original surface and $A_{(s),\mathbf{v}}$ is the total one-ring face area on the current skeleton mesh. If we had started the

Algorithm 9 Advection-based mesh contraction

```
 1: procedure AdvectionContraction($V, N$)
 2:     $f \leftarrow 0.5$                                  ▷ Linear interpolation weight
 3:     for $i = 1 \rightarrow N$ do                        ▷ Iterate $N$ times
 4:         for all Vertex $\mathbf{v} \in V$ do
 5:             $\mathbf{v}' \leftarrow \mathit{evalAdvectionStep}(\mathbf{v})$
 6:             $\mathbf{v} \leftarrow f\mathbf{v} + (1 - f)\mathbf{v}'$
 7:         end for
 8:     end for
 9: end procedure
```

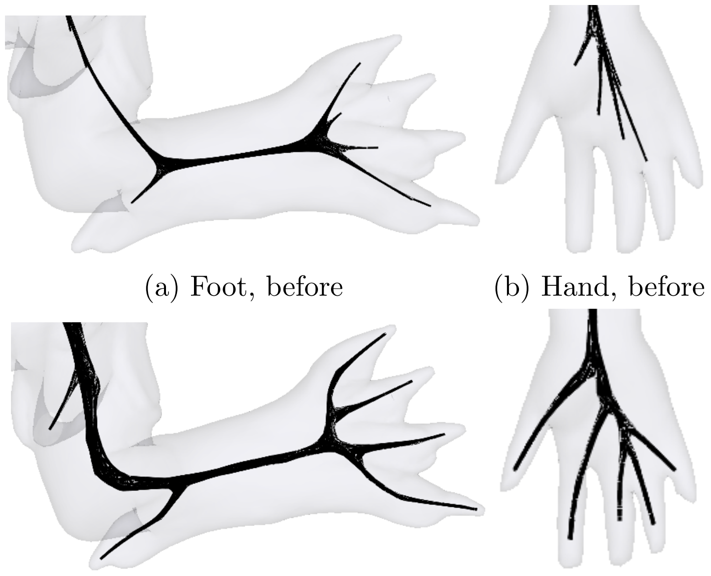
Algorithm 10 Advection-based mesh contraction with positional constraint

```
 1: procedure AdvectionContraction($V, N$)
 2:     for all Vertex $\mathbf{v} \in V$ do
 3:         $W_{P,v} = \frac{A_{(0),v}}{A_{(s),v}}$              ▷ Using Equation 3.6
 4:     end for
 5:     for $i = 1 \rightarrow N$ do                        ▷ Iterate $N$ times
 6:         for all Vertex $\mathbf{v} \in V$ do
 7:             $\mathbf{v}^* \leftarrow \mathit{evalAdvectionStep}(\mathbf{v})$
 8:             $\mathbf{n} \leftarrow \dfrac{\mathbf{v}^* - \mathbf{v}}{\|\mathbf{v}^* - \mathbf{v}\|}$
 9:             $\mathbf{v} \leftarrow \mathbf{v} + \dfrac{\|\mathbf{v}^* - \mathbf{v}\|}{W_{P,v}}\mathbf{n}$
10:         end for
11:         for all Vertex $\mathbf{v} \in V$ do
12:             $W_{P,\mathbf{v}} = \frac{A_{(0),\mathbf{v}}}{A_{(s),\mathbf{v}}}$      ▷ Using Equation 3.6
13:         end for
14:     end for
15: end procedure
```

contraction on the original mesh, the initial value of $W_{P,\mathbf{v}}$ would have been 1. Since the contraction is applied on the surface-skeleton mesh, the constraint is active starting from the first iteration. Weights are updated after each iteration of the process. The new procedure is given in Algorithm 10.

The modified process first uses the position calculated by the advection step to determine the direction of the contraction (line 8). Then, using the weight, it determines how far the vertex should be moved on that direction (line 9). This is the reason we are setting a lower bound of 1 for $W_{P,\mathbf{v}}$ in Equation 3.6, so that the vertex never moves farther from the position calculated by the advection step.

As the mesh gets smaller in area and volume, the value of $W_{P,\mathbf{v}}$ increases. This increase is faster in the smaller parts of the object, hence slowing down their contraction in earlier iterations. The difference between the results before and after the implementation can be seen in Figure 3.12.



(a) Foot, before (b) Hand, before

(c) Foot, after (d) Hand, after

Figure 3.12: Most details are lost without a constraint (top row). After the addition of the constraint, details can be preserved (bottom row). These results are obtained after the same number of iterations.

3.3.3 Discussion

Introducing the Mean-shift indeed yields thinner meshes within the same number of iterations, but there is one downside: this operation also smooths out small details faster. In the bottom row of Figure 3.13, we see the hand of the Neptune model in three different cases: the original method, only positional constraint enabled and mean-shift and positional constraint enabled together.

The positional constraint also introduces an issue. Although this addition preserves the small details quite well by preventing their over-contraction, at the same time, it limits the contraction of all other vertices as well. Therefore, after the same number of iterations, constrained contraction yields slightly thicker meshes. But, this is mostly not a big problem since we have also added the edge-collapse step to this method. Therefore, the small increase in thickness does not cause significant problems. Still, one visible artefact for Neptune model is the sharp bends around the shoulders (Figure 3.13b, top). Other methods have smoother bends in the same area. This shows that the junction area in a bulky body part can be problematic, but this is a tolerable trade-off considering the difference in the degree of preserved detail (Figure 3.13b, bottom).

In conclusion, we have managed to combine the near-perfect centeredness of the method with an adequate detail-preserving mechanism. Also, if small details are not essential, the mean-shift clustering can be used accelerate the contraction process. If detail preservation is essential, then it is more suitable to enable only the positional constraint. In conclusion, by all these additions, method of Telea and Jalba has been improved and it can be compared to state-of-the-art approaches, since the quality of the extracted curve skeletons has increased significantly concerning two important criteria.

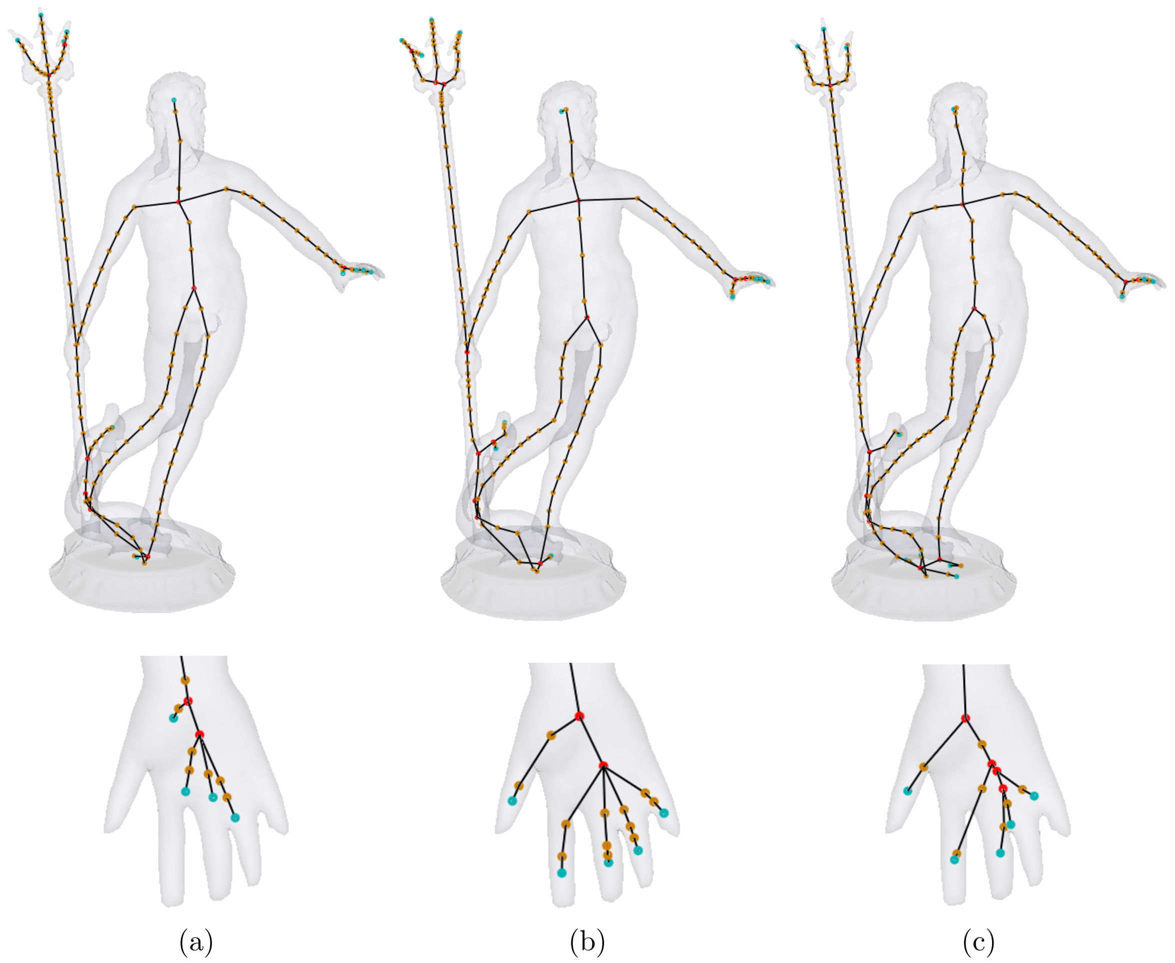


Figure 3.13: The curve skeleton of the Neptune model (top) and its hand (bottom) in four cases: The original method (a), only positional constraint enabled (b), mean-shift and positional constraint enabled together (c).

Chapter 4

Results and Comparisons

So far, a number of state-of-the-art methods for extracting curve skeletons were described in detail, but no result analysis or comparison has been done. In this chapter, we analyze the results of our methods by a set of criteria, and compare them to the results of the other methods. The methods which are considered for comparison are examined in detail in Chapter 2.

The criteria used in this chapter are:

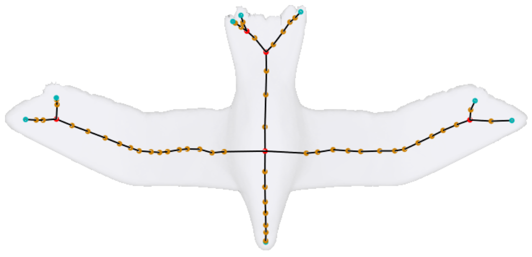
- Centerdedness
- Detail preservation
- Smoothness of the skeleton
- Independence from mesh sampling.

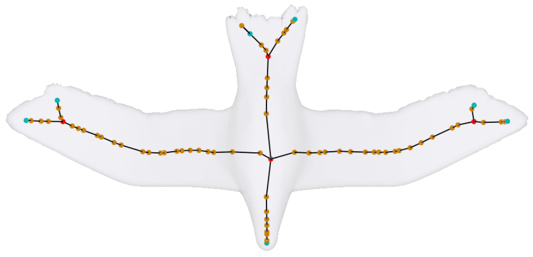
Note that speed is not considered as a criterion. The reason for this is that we are interested in the quality of the skeletons, rather than the efficiency of the extraction.

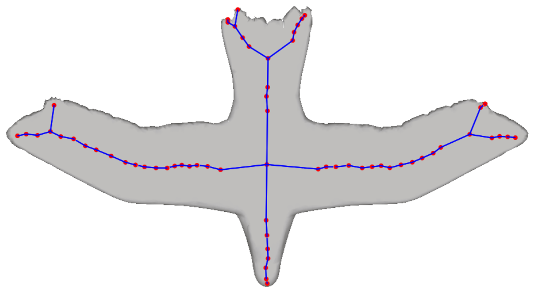
To analyze the results of Au et.al., Cao et.al. and Tagliasacchi et.al., we are using the images captured from authors' original implementations. For our own results and the results of Telea and Jalba, we are using the screen captures from our own implementation. An important point here is that we apply the edge collapse and re-centering steps of our method to Telea and Jalba's results in order to make them more comparable to the others. For the improved version of this method, we have included the results only when our improvements cause significant improvements, e.g. discussion for detail preservation.

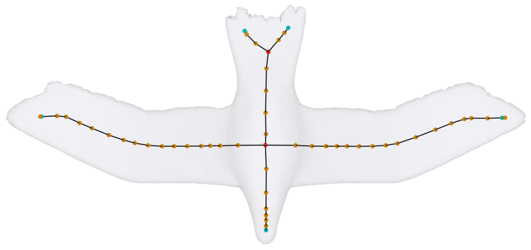
In addition to the methods mentioned above, we have also included results of our method without using the surface skeleton, i.e., by applying the contraction directly to the original mesh, same as [4]. By including this, we hope to see if using an intermediate structure improves the quality of the curve skeletons.

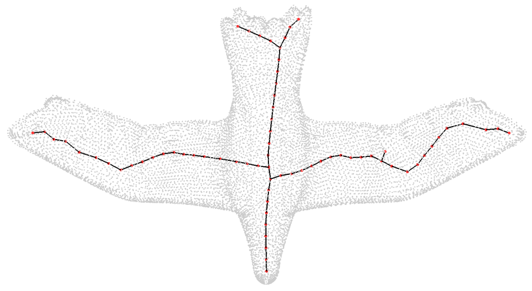
In the remainder of this chapter, we first give outcomes from the applications using several models in Figures 4.1-4.4. Later, in the following sections, we focus on one criterion at a time. In each of these sections, we first examine our results and then compare the results of all methods by zooming in on specific parts of the models, to discuss specific cases.


(a) Our method


(b) Our method, direct


(c) Au et al.


(d) Telea and Jalba


(e) Cao et al.

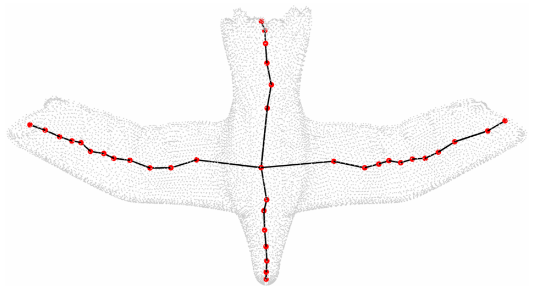
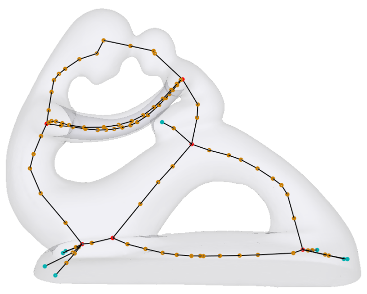
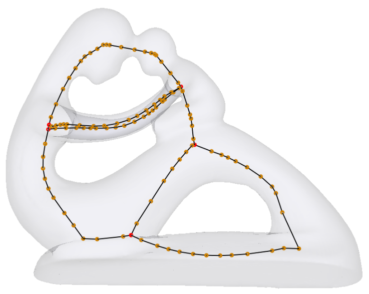
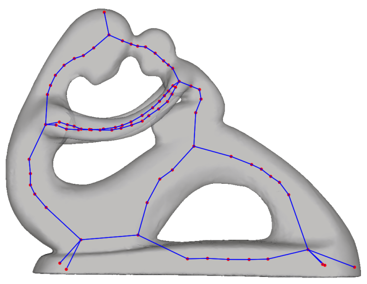
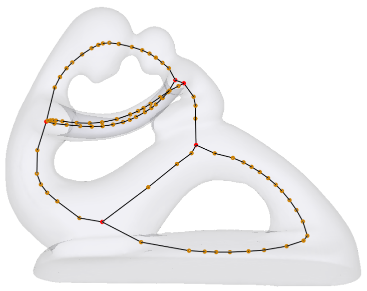

(f) Tagliasacchi et al.

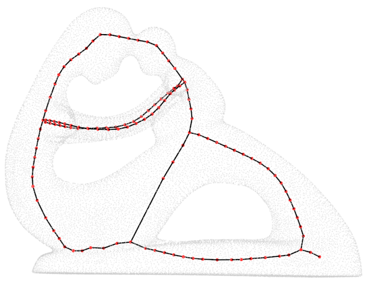
Figure 4.1: Curve skeletons for the bird model (11K vertices, 23K faces)


(a) Our method


(b) Our method, direct


(c) Au et al.


(d) Telea and Jalba


(e) Cao et al.

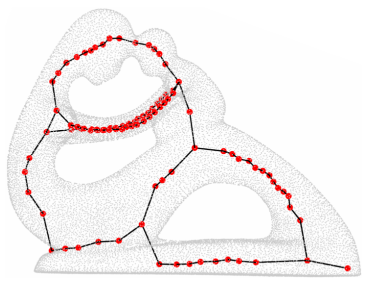
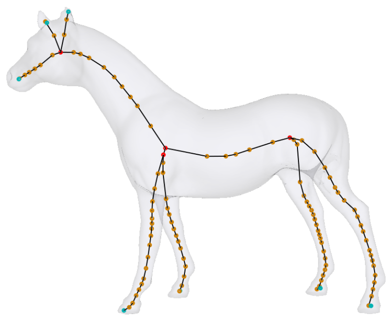
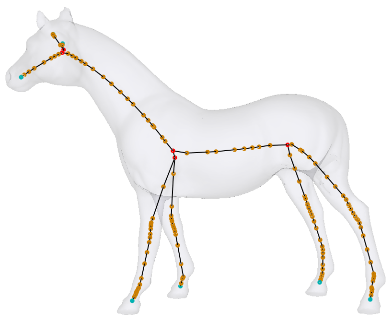
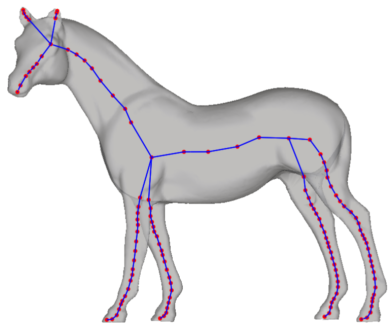
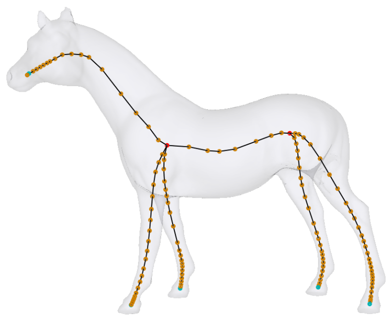

(f) Tagliasacchi et al.

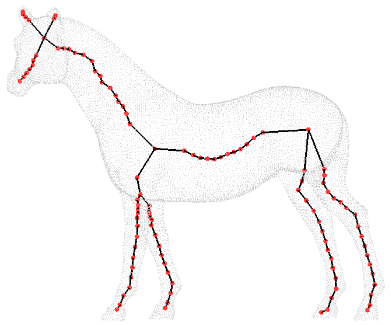
Figure 4.2: Curve skeletons for the fertility model (25K vertices, 50K faces)


(a) Our method


(b) Our method, direct


(c) Au et al.


(d) Telea and Jalba


(e) Cao et al.

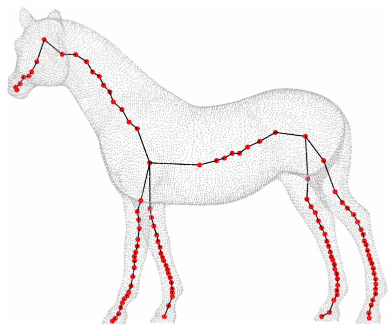
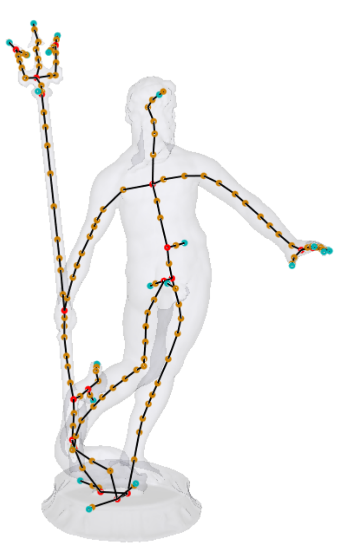
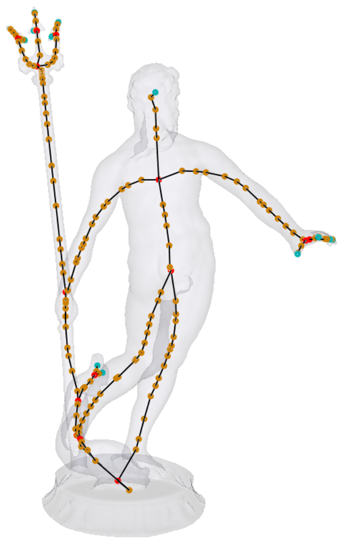
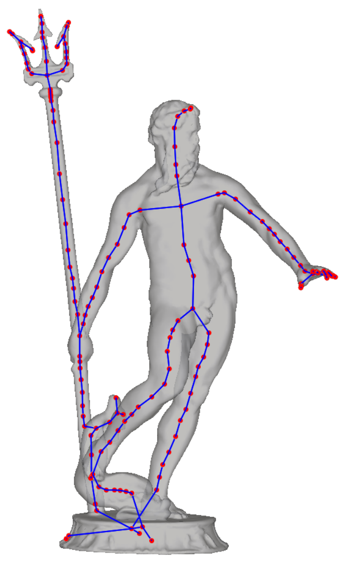
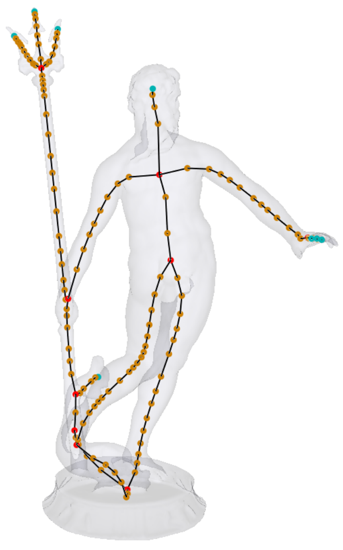

(f) Tagliasacchi et al.

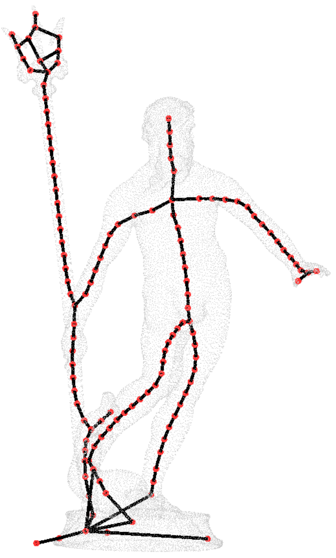
Figure 4.3: Curve skeletons for the horse model (24K vertices, 48K faces)


(a) Our method


(b) Our method, direct


(c) Au et al.


(d) Telea and Jalba


(e) Cao et al.

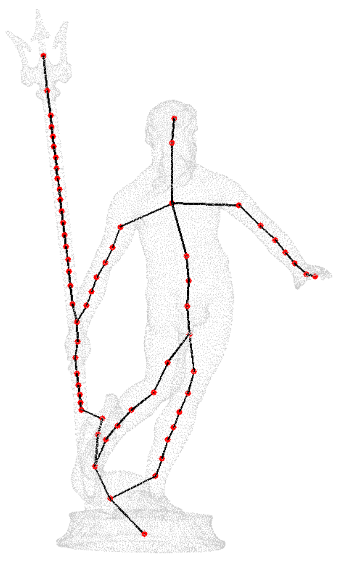

(f) Tagliasacchi et al.

Figure 4.4: Curve skeletons for the Neptune model (28K vertices, 56K faces)

4.1 Centeredness

The importance of centeredness depends on the purpose of an application. For instance, if the skeleton is going to be used for segment identification, as in [3], positions of the nodes will be less important compared to the connection between the node and the surface. However, in applications such as shape matching or modification, as [5] and

[6], the positions of the skeleton nodes carry a bigger importance. Also, if an additional property such as feature points is needed, e.g. the surface parameterization example in Chapter 5, the skeleton nodes again need to be well-centered.

4.1.1 Results

As claimed by Au et.al., the mesh contraction indeed moves the skeleton nodes off-center [4]. In some cases, some vertices may even go outside of the boundary. A perk of starting with the surface-skeleton mesh is that the nodes go less off-center compared to other contraction methods (which contract the original mesh), since surface-skeleton points are already centered by definition.

To re-center the nodes, we apply a geometric, weighted average using a skeleton-node-to-mesh mapping, as described in Section 3.2.3. This approach accurately centers the nodes which are located in cylinder-like branches of the shape; such as arms, legs, horns and wings. However, in bulky parts, junctions and tips, centering is less accurate (Figure 4.5). This inaccuracy is caused by the fact that the skeleton-node-to-mesh mapping has a widely varying distribution of the nodes for the curve-skeleton nodes corresponding to such parts of the surface, rather than a fair or uniform distribution (Figures 4.6g, 4.6h). Although the weighted average tries to solve this issue, the mapping is too unevenly sampled in those parts, therefore a perfect weighting scheme is not available.

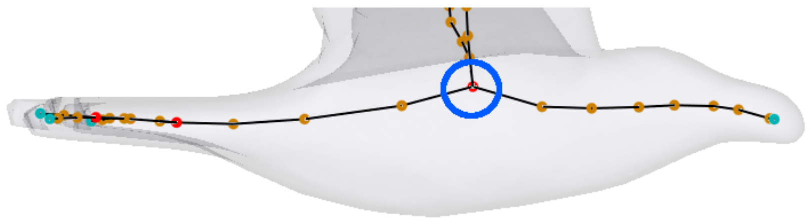


Figure 4.5: Centering is accurate except at junctions of the bulky parts of the shape.

The mapping examples in Figure 4.6 show the three cases (where the subject node is rendered red):

- **Tip node:** Blue frame (Figure 4.6a) together with views from different angles (Figures 4.6b, 4.6c and 4.6d).

- **Regular node:** Red frame (Figure 4.6a) and two views from different angles (Figures 4.6e and 4.6f).

  This is the ideal case for re-centering as the mapping resembles a cylinder with fairly distributed vertices. Even a non-weighted average would accurately re-center such skeleton nodes.

- **Junction node:** This node is a junction node in a bulky part of the object. Green frame (Figure 4.6a) and views from different angles (Figures 4.6g and 4.6h).

  As can be seen from the figures, a lot of mesh vertices are collapsed onto the junction node. This provides a mapping which is distributed to a larger part of the surface. Being a junction also brings an unfair distribution and the node stays off-center even after the re-centering. Due to these reasons, a skeleton node might be pulled away from its position easier.

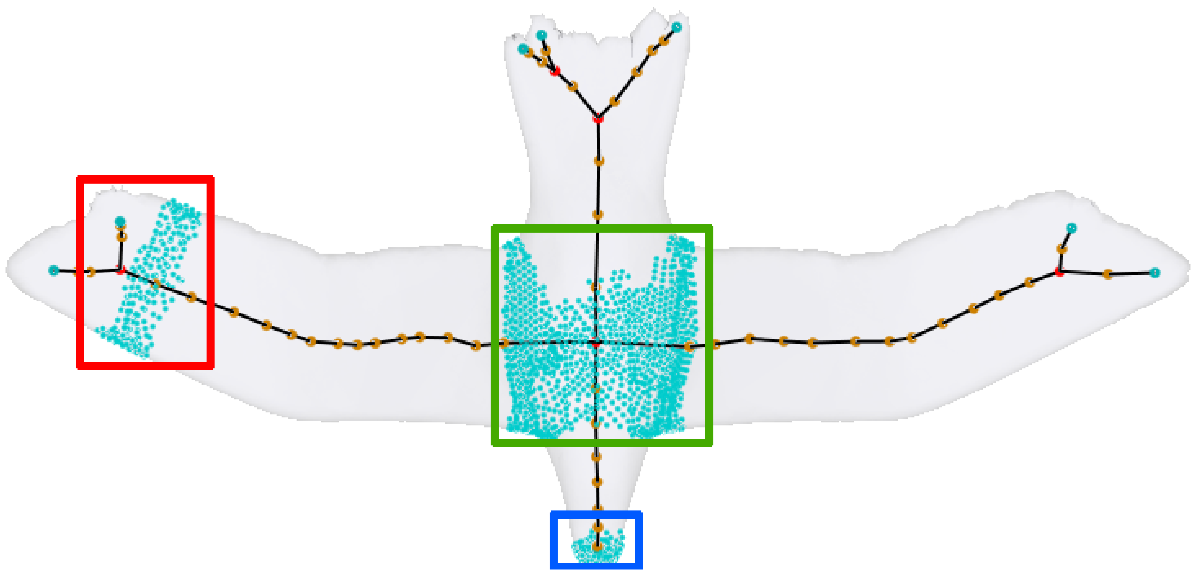
4.1.2 Comparison

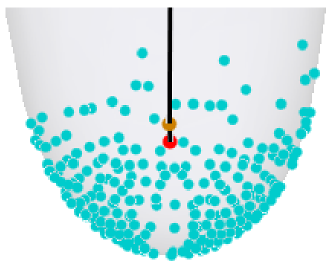
Similar to the result analysis, we compare the methods by looking at a few specific cases: a junction node in a bulky part and a corner node. For this, we zoom in on specific parts of some models as shown in Figure 4.7. The comparison ends with a discussion of overall centeredness.

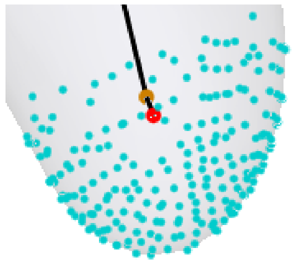
Case 1: Bulky Parts and Junctions

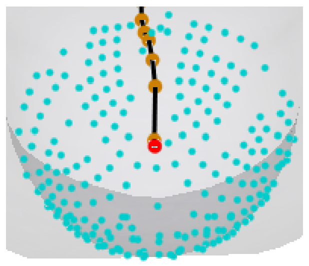
In Figure 4.8, we see the node-of-interest highlighted inside a circle for each method. For every instance, except for Telea and Jalba [7], we see a similar problem: the junction node is off-center. We, Au et al. [4] and Cao et. al [5] use similar re-centering mechanisms, relying on a (weighted) average of the related vertices on the original surface. Figure 4.6 shows that such related vertices are not evenly distributed. Even though the averaging uses an effective weighting scheme, inaccuracy increases as the vertex distribution gets more uneven. When this is combined with the large size of the mapping, the skeleton node is naturally pulled off-center (Figures 4.8a, 4.8b, 4.8c and 4.8e).

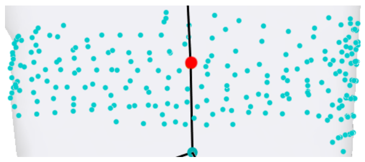
For Tagliasacchi et al. [6], the problem is caused by a slightly different reason. In [6], it is mentioned that orientation information in junction areas is not reliable. Since this method is orientation-sensitive as well as position-sensitive, unreliable information

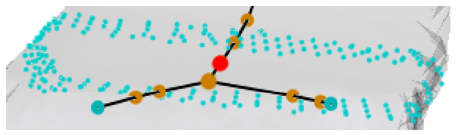

(a)

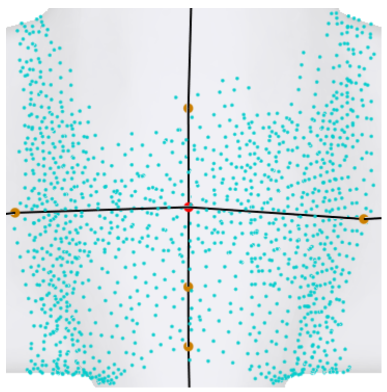

(b)


(c)


(d)


(e)


(f)


(g)

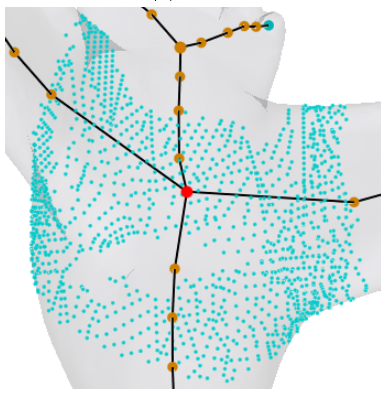
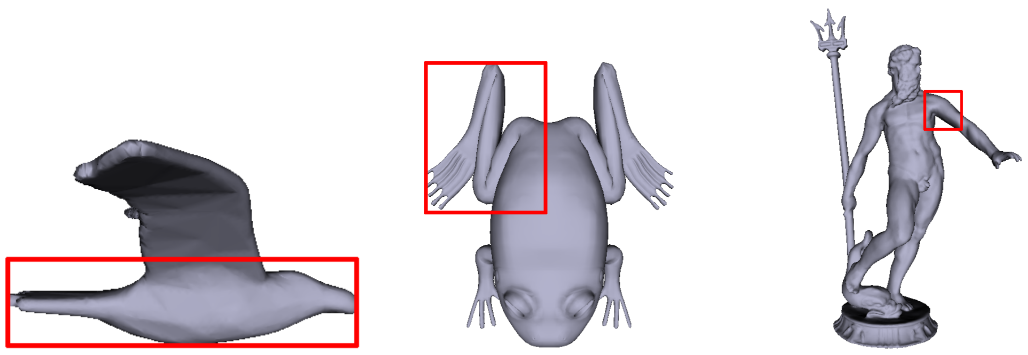

(h)

Figure 4.6: Skeleton-node-to-mesh mapping instances and views from different angles.



(a) Bird (b) Frog (c) Neptune

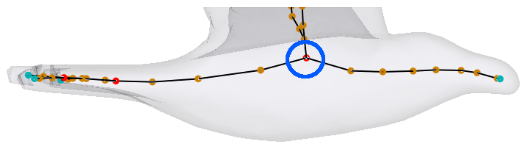
Figure 4.7: Zoomed-in areas of models.

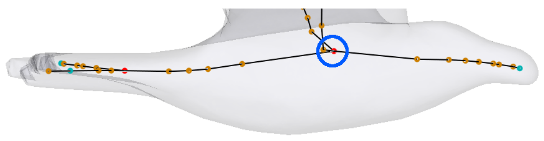
highly reduces the accuracy of centeredness (Figure 4.8f). This unreliability is partly caused by the same reason that yields off-center junction nodes for the previous methods. There are a lot of surface points in this area and their orientation is very diverse. As a result, finding an optimal cut-plane gets very hard, even impossible. To overcome this, the junction nodes are treated specially. This treatment does well in the junction areas that resemble a fork (Figure 2.7), where cylindrical regions meet. In this particular case, the branches meet in the top of the body, where there is a bulky chest below. This makes it very hard to use spatial coherence to make a good "guess" on how the node should be positioned.

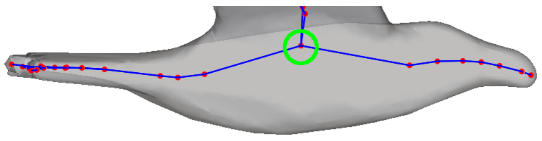
The best centering in this case is obtained by Telea and Jalba [7]. While contracting the surface-skeleton mesh, this method moves the vertices along the gradient field of the importance of surface-skeleton points. As also stated in Section 3.1.2, importance increases towards the center of the object. Therefore, during the contraction, vertices are by default moved towards the center. Although this yields an accurate centering without an extra re-centering step (Figure 4.8d), there is a trade-off concerning detail preservation; this is discussed later in this chapter.

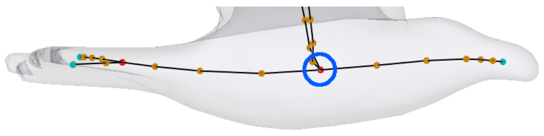
Case 2: Sharp Corners

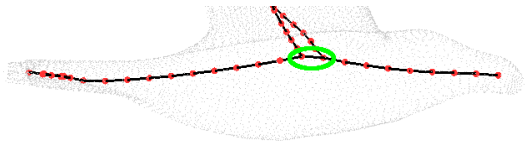
In Figure 4.9, we see a very unique case. One aspect is that the leg has a very sharp beveling. Another aspect is, in the right part of the leg the mesh has a very sparse sampling of vertices. One more aspect is that the mesh is self intersecting, i.e. vertices of one branch is inside of another branch. This is an undesired property, yet this particular


(a) Our method


(b) Our method, direct


(c) Au et.al.


(d) Telea and Jalba


(e) Cao et al.

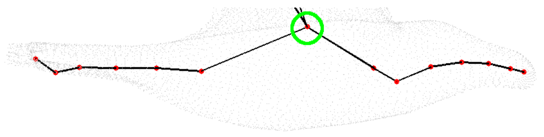

(f) Tagliasacchi et al.

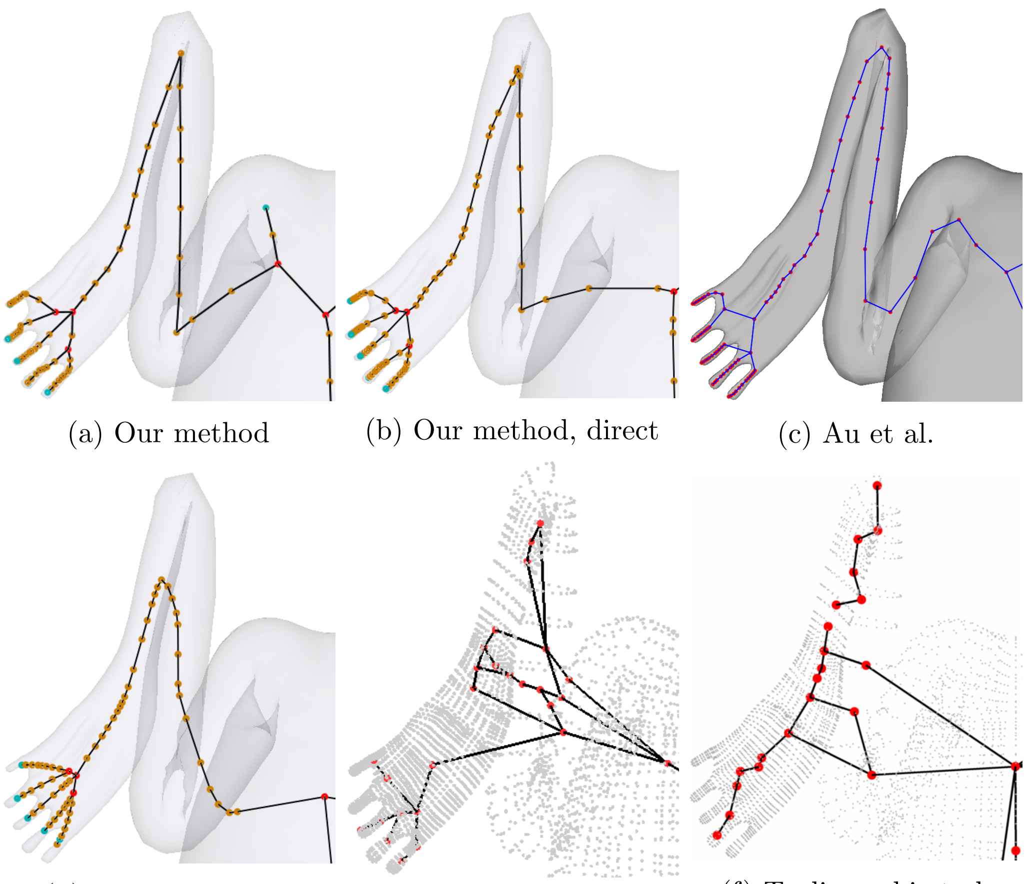
Figure 4.8: Bird model (11K vertices, 23K faces) for comparing centeredness of a junction node in a bulky part.

shape region has a very good example of sharp corners.

Similar to the previous case, both versions of our method and method of Au et al. [4] yield similar results. The reason is again the skeleton-node-to-mesh mapping. This time, the distribution is uneven because of the shape: there are more vertices on the inner side of the corner compared to outer side (vertex distribution is visible in Figure 4.9e). Hence the corner vertices are pulled more towards the inside. In terms of comparison, the skeleton of Au et. al. has better node positioning in the upper corner, while our method has a better result in the lower corner of the leg, as shown in Figures 4.9a-4.9c.

In Figure 4.9d, we see that the skeleton of Telea and Jalba [7] does not have much good accuracy as in the previous case. The reason here is also the unusual vertex distribution. If we look at the gradient field of the importance measure (Figure 4.10), we see that vectors point to various directions around this area. The effect of the sparse sampling is also clearly visible. Lastly, with the lack of a strong positional constraint, corner nodes are pulled in too much and therefore go off-center.

In the results of the point-cloud-based methods, we see the worst outcomes, the curve skeletons even have different topology than the original surface (Figures 4.9e-4.9f). The main reason here is the lack of mesh connectivity and surface information, as well as the self intersection. Since there is no explicit relationship between the vertices, con-


(a) Our method (b) Our method, direct (c) Au et al.

(d) Telea and Jalba (e) Cao et al. (f) Tagliasacchi et al.

Figure 4.9: Frog model (17K vertices, 34K faces) for comparing centeredness in a sharp corner.

nectivity information is artificially calculated using proximity-search-based methods. In this example, the two parts of the leg come very close to each other (and even touch, due to self intersection) and therefore considered as neighbors. The sparse sampling also hinders an accurate nearest-neighbor search, hence the end outcomes have wrong topology. ROSA [6] even shows disconnectivity, as the distance between the neighborhoods causes the algorithm to see these parts as separate objects. There is a special check while finding the cut-planes, to see if the plane cuts the surface on multiple parts, but it falls short when there is self intersection, and sampling circumstances are as unusual as in this case. In order to distinguish different parts in this kind of situations, surface information is thus essential.

If we look at another corner case with less sharpness, we still see similar results. Fig-

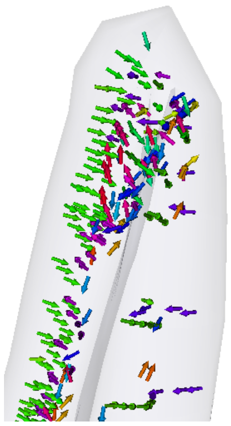
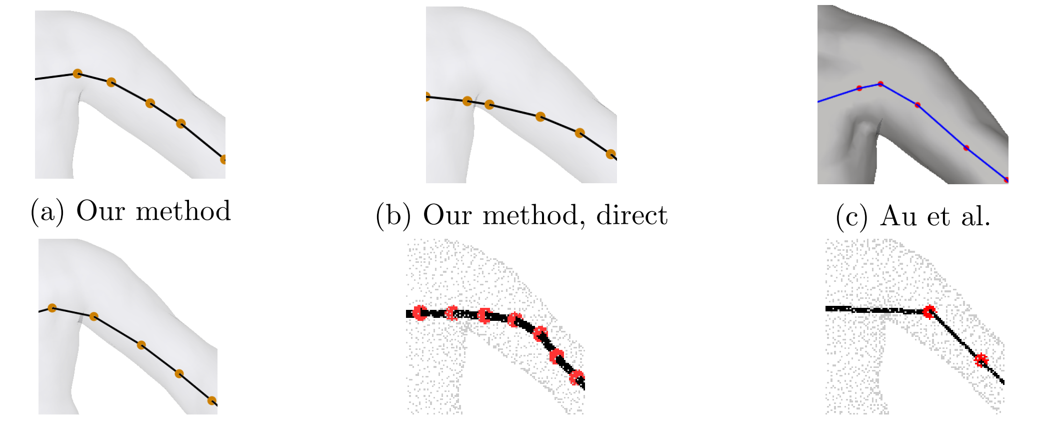


Figure 4.10: The gradient field around the sharp corner. Also notice the right side of the leg area has a sparse sampling of gradient vectors.

ure 4.11 has zoomed in images from the armpit area of the Neptune model (Figure 4.7c, red frame). This area does not have exceptional properties as in the previous example. As a result, having a decent vertex distribution and fair distances between object parts indeed improve the accuracy of all methods. Figures 4.11a and 4.11c show better corner handling compared to others, while the weakest link among mesh-based methods is our method with direct contraction (Figure 4.11b).



(a) Our method (b) Our method, direct (c) Au et al.

(d) Telea and Jalba (e) Cao et al. (f) Tagliasacchi et al.

Figure 4.11: Zoom in around the armpit of Neptune model. The results are comparable concerning the centeredness, if the mesh does not have unusual properties.

Overall Discussion for Centeredness

We can conclude that the methods utilizing mesh contraction yield better-centered skeletons compared to the point-could based methods. In general, the method of Telea and Jalba [7] results in the best centeredness in regular branches, i.e. tube-like cylindrical regions, without an extra re-centering step. Around the corners, however, the lack of a decent positional constraint really shows itself. After our addition of the positional constraints, centeredness significantly improves (Figure 4.12).

Corners are less of a problem for our approach and that of Au et al. [4], yet the results are still not perfect. As the corner gets sharper, curvature increases and centeredness gets less accurate. But for moderate corners, as in the Neptune model, the results are accurate and visually pleasant.

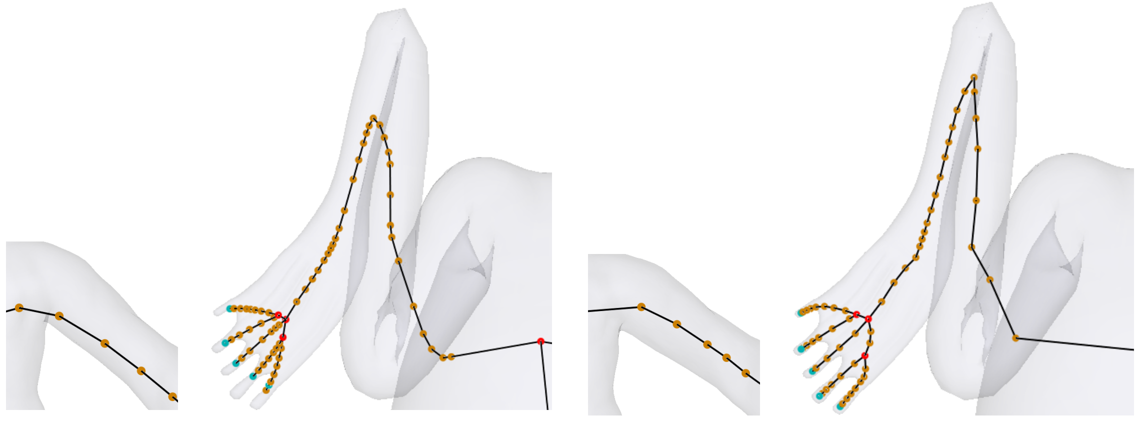

(a) Telea and Jalba (b) Telea and Jalba, improved

Figure 4.12: Comparing centeredness before (a) and after (b) enabling the positional constraints for the method of Telea and Jalba.

One of our main purposes for using the surface skeleton was to get better-centered skeletons. From the case instances and overall pictures, it is clear that the inclusion of this intermediate step yields better skeletons. Since the surface skeleton is formally based on centeredness, applying the contraction to a partly centered structure disrupts less the overall centeredness. All in all, as long as this criterion is concerned, using the intermediate step turned out to be useful. One possible improvement can be improving on the edge-collapse step by calculating an optimal position for the remaining vertex after a collapse, similar to [11].

Among the methods which use point clouds, a common problem is the lack of mesh connectivity information. The relationships between the vertices need to be built artificially. Since this is mostly done by proximity-based methods, the results get affected from close surface parts and sparse vertex sampling.

For ROSA [6], centeredness is even more problematic. Depending on the parameters, few junction nodes are likely to stay off-boundary (Figure 4.13). This also adds parameter sensitivity to the problems. This is actually a big problem for both [5] and [6]. There is a lack of globally-appropriate parameter values, as the present ones can tune the outcome of only a specific area or a specific mesh. The connectivity-building radius of Cao et al. [5], plus the graph-extraction and neighborhood radius parameters of Tagliasacchi et. al. [6] are such examples.

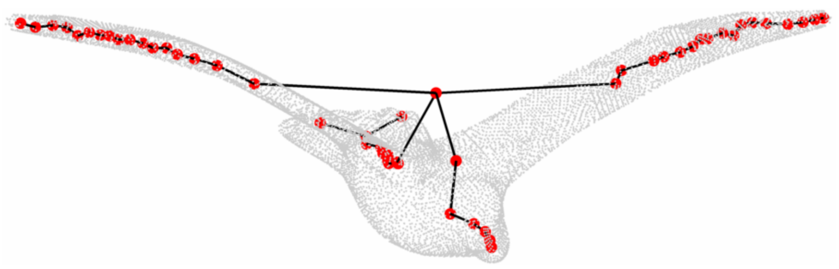


Figure 4.13: Slight change in parameters cause drastic differences in results. In this figure, the radius for the Mahalanobis distance computation (for ROSA [6]) is set to 0.1, as opposed to 0.2 used earlier in Figure 4.8f.

4.2 Detail Preservation

Preserving detail means that it should be possible to observe the details of the shape by looking at its skeleton. For the large parts, this is less of an issue since the methods can easily preserve these (e.g. wings of a bird). However, smaller details (e.g. fingers, horns, body bumps) are harder to preserve, since it gets harder to determine if such a section is noise or an important detail.

For some purposes, such fine details are not essential. The methods of Cao et al. [5] and Tagliasacchi et. al. [6] are more focused on the larger parts of the object. For skeletal animation, shape matching or shape modification, it is useful to have skeleton parts for even the small details of an object, since this gives more freedom for analyzing and controlling the shape.

4.2.1 Results

Smaller details such as bumps are generally not of importance. But sometimes actual parts may be embedded in a bump. In Figure 4.14a, the bumps on the wings represent such a case. A perfect detail-preserving method would create branches in the curve skeleton for those bumps, since the veins on the wing represent an actual detail of the shape.

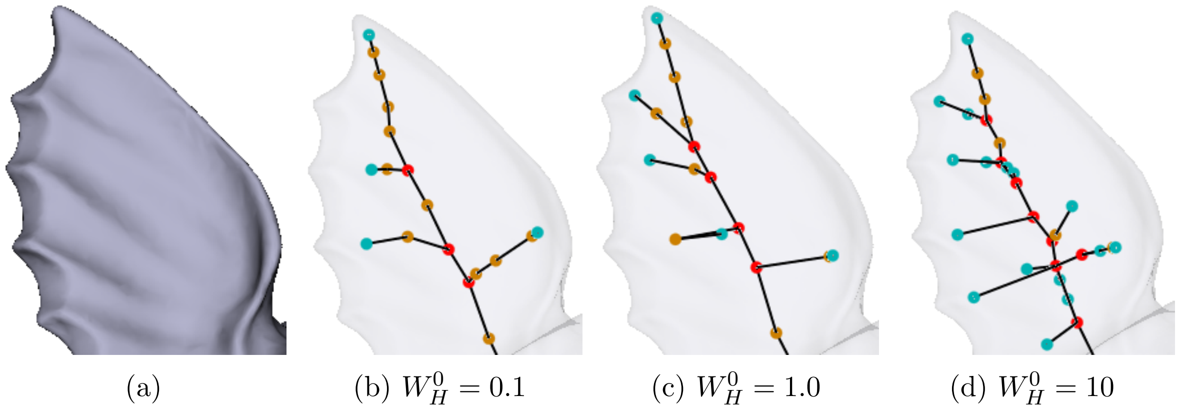

(a) (b) $W_H^0 = 0.1$ (c) $W_H^0 = 1.0$ (d) $W_H^0 = 10$

Figure 4.14: Wing of the gargoyle model with small bumps on it (a). Curve skeletons of the same part with different W_H^0 values (b, c and d).

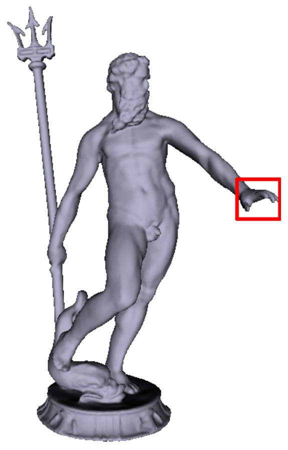
The most significant factor related to preserving details belongs to the contraction step, namely, W_H in Equation 2.1. Recall that W_H is the weight for attraction. That is, it tries to prevent over-contraction of the mesh by trying to keep vertices close to their initial position. Recall Equation 2.1 from Section 2.1.1:

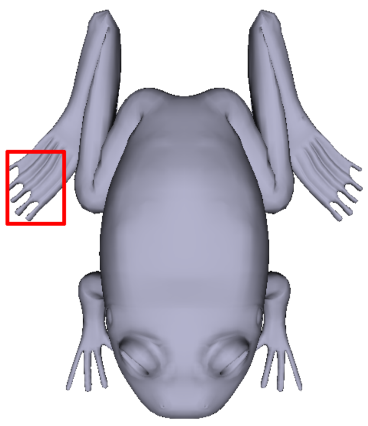
$$W_{H,i}^{t+1} = W_{H,i}^{0}\sqrt{A_i^0/A_i^t}.$$

By updating W_H with different coefficients, i.e. giving different values to W_H^0 in Equation 2.1, we can get different results. In Figure 4.14, we see the wing of the gargoyle model and the curve skeletons extracted with different W_H^0 values. As can be seen, larger weights preserve more details. However, there are two issues to consider. The first one is that this tuning can make the contraction sensitive to noise. Figure 4.14d has the highest level of detail, but also the highest level of noise. For such large W_H^0 values, the cleaning process in the post-processing step needs to be fine-tuned to distinguish noise and real detail. This is because such details may also be made of only one edge, which are likely to be removed by the cleaning process. The second issue is that a larger constraint means that other parts of the body will be contracted less, since the attraction constraint will hold other vertices in position as well, though not as strongly as the vertices corresponding to the small details. As a result, the contracted mesh can be thicker than desired, and the edge-collapse process will yield a lesser-quality curve skeleton. In thick areas, the edge-collapse process leaves less skeleton nodes hence causing long edges and poor skeleton-node sampling. Skeleton-node sampling is examined in Section 4.3.

4.2.2 Comparison

In order to compare detail preservation, we again look at different cases for comparing different methods and finish the section with a general discussion. The cases examined here are small extremities and meaningful bumps, see Figure 4.15.


(a) Neptune


(b) Frog

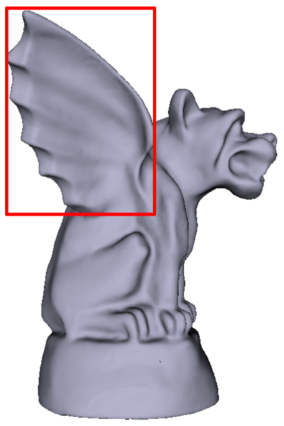

(c) Gargoyle

Figure 4.15: Zoomed-in areas of models.

Case 1: Small Extremities

In this part, we look at the small extremities of the surface, such as fingers and horns. These are mostly cylindrical parts, so all methods are expected to perform well, but the problem is that their size might cause them not to be considered as meaningful details but as noise. Figure 4.16 shows the hand of the Neptune and the foot of the frog model, for each method.

Two versions of our method, and Au et al. once again have the best results. Even though our method preserves all the details, there are centering issues with the hand. For the foot, there are no significant problems. Also, the junction merging step of [4] is clearly visible here, since our results have more junctions at the same areas. But this does not cause a problem, or make one result better than another; see Figures 4.16a-4.16c for the results.

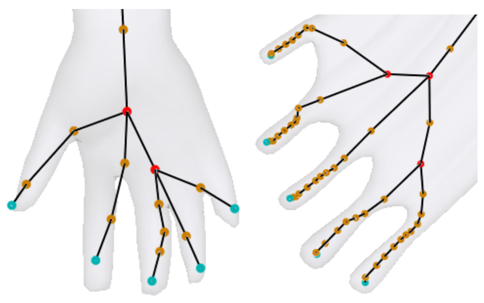
We have mentioned that the centering accuracy of Telea and Jalba [7] comes with a trade-off in detail (Figure 4.16d). The vertices in the small details are pulled towards the global center, rather than towards a local one. Since the importance is based on geodesic length, the importance of the vertices in such small details are smaller, even though they are locally centered. This is still less visible in the foot, but if more iterations were applied, those fingers too, would have vanished. In the improved version (Figure 4.16e), we see that the detail-loss problem is solved. Combined with the near-perfect centeredness of [7], the improved version has the best results in terms of detail preservation and centeredness, compared to the other methods.

For the point-cloud-based methods, the same disadvantages for centeredness apply here as well (Figures 4.16f and 4.16g). The small size of the details and proximity of the vertices again result in a curve skeleton without any small branches. In order to protect such details, these methods will need local parameters, as the global ones are focused on a more high-level skeleton that captures only the significant details. Setting the global parameters in favor of the small extremities disrupts the overall quality of the curve skeleton.

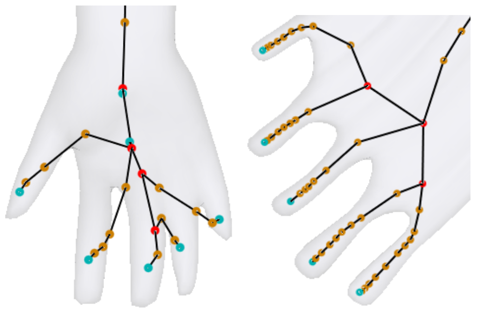
Case 2: Meaningful Bumps

For this case, we consider the bumps which are real shape details, see Section 4.2.1. The same part of the wing from the gargoyle model is examined in Figure 4.17. Here, our method is run with default $W_H^0 = 1$.

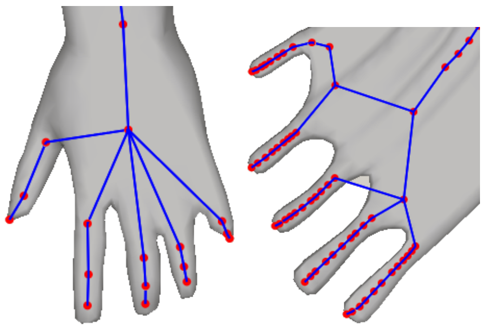
In order to analyze contraction-based methods better, in Figure 4.18 we also look at the contracted meshes, before extracting the final curve skeleton. Here, it is possible to see that only our method preserves the fine details during the contraction step. This is a perk of applying the contraction to the surface skeleton. However, as mentioned before, the global importance measure and the lack of positional constraints causes the skeletons of Telea and Jalba [7] to lose almost all fine detail: all small details are contracted towards



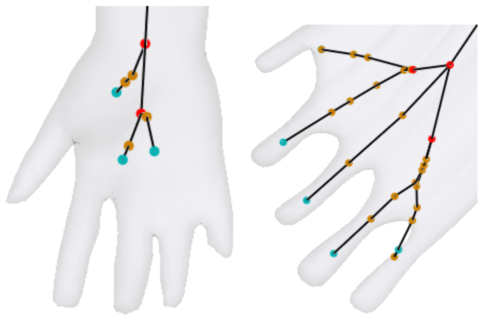
(a) Our method



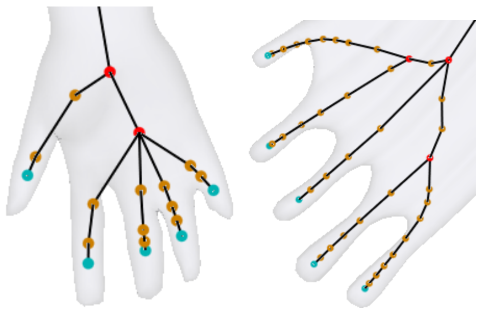
(b) Our method, direct



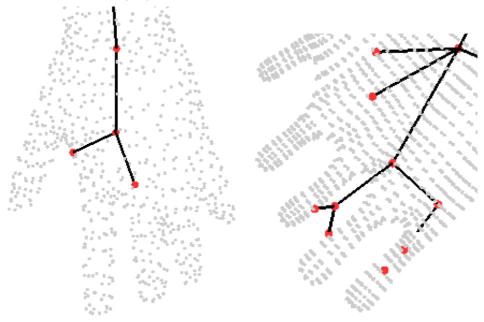
(c) Au et.al.



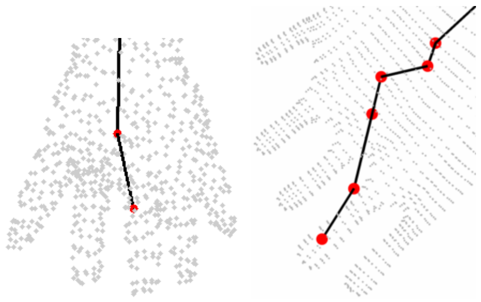
(d) Telea and Jalba



(e) Telea and Jalba, improved

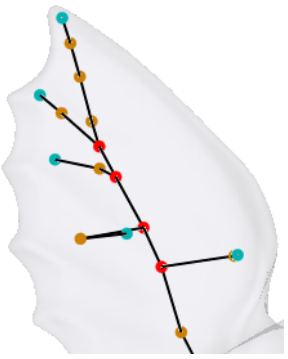


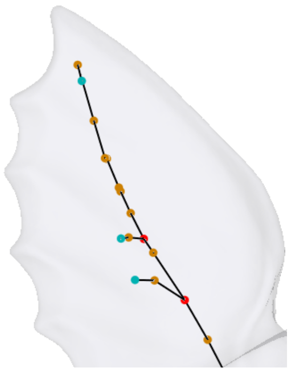
(f) Cao et al.

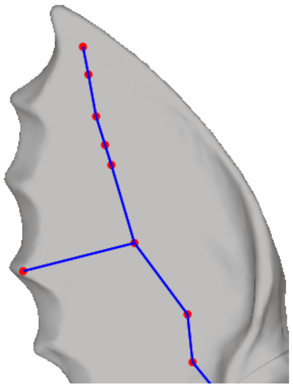


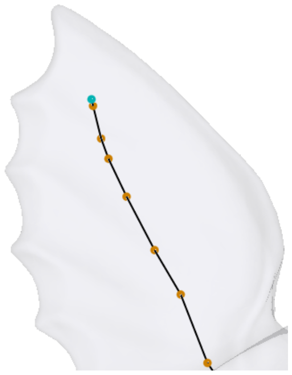
(g) Tagliasacchi et al.

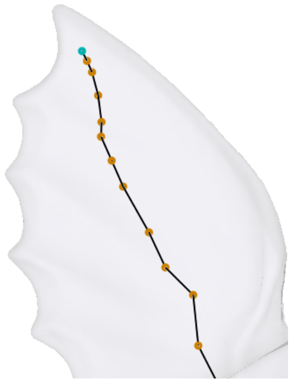
Figure 4.16: The hand of Neptune and the foot of the frog for the comparison of detail preservation in small extremities.

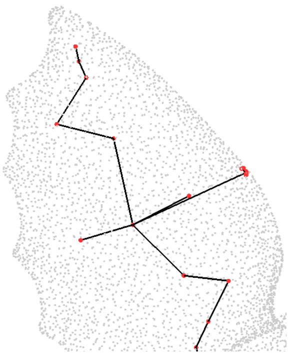

(a) Our method


(b) Our method, direct


(c) Au et.al.


(d) Telea and Jalba


(e) Telea and Jalba, improved


(f) Cao et al.

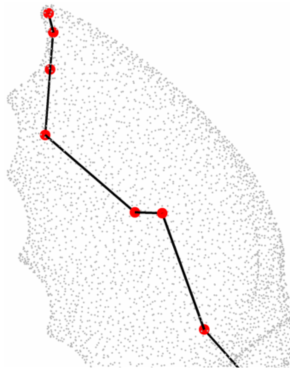
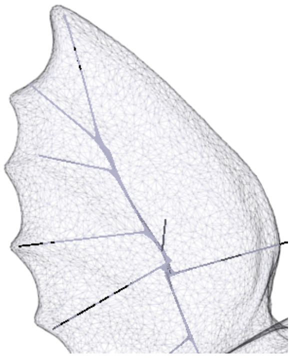
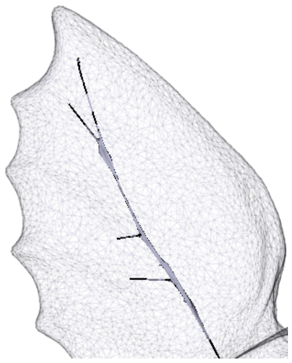
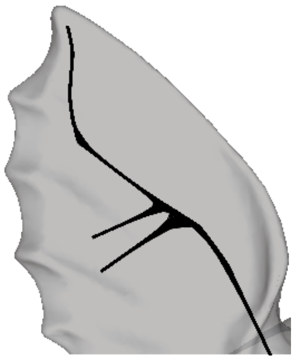

(g) Tagliasacchi et al.

Figure 4.17: The wing of the gargoyle model for comparing details in bumpy parts of the shape.


(a) Our method


(b) Our method, direct


(c) Au et.al.

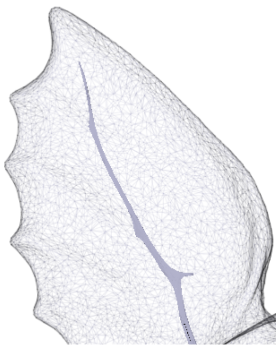

(d) Telea and Jalba

Figure 4.18: Some contracted meshes just before the edge-collapse step.

the main center of the wing (Figure 4.17d). Even the improved version of this method cannot preserve this kind of details (Figure 4.17e), whereas extremity details such as the fingers can adequately be preserved. The only improvement visible between the original method of Telea and Jalba, and its improved version in this case is in the increased number of skeletal nodes and edges.

If we compare the results, before and after the edge collapses, we also see the effect of the collapsing step in the detail preservation. Figures 4.17a, 4.17b and 4.17c have one less branch compared to their mesh equivalents, Figures 4.18a-4.18c. So, while the edge collapse can eliminate noise, it can sometimes remove branches as well.

For the point-cloud-based methods, the results resemble the ones in the previous case. But from the zig-zag effect, we can conclude that small details still have some influence, since they have pulled skeleton nodes towards themselves, when re-centering was applied (Figures 4.17f and 4.17g).

General Discussion

In general, the same reasons causing inaccuracy in centeredness disrupt the quality of detail preservation as well. Especially for Cao et al. [5] and Tagliasacchi et. al. [6], the same issues come up case after case.

In terms of using the surface-skeleton mesh, the results have proved in favor again. With a well-constrained contraction scheme, more details can be preserved in the curve skeleton. Another important point here is the detail preserved after contraction, during the post-processing step. When contraction and edge collapse are tuned for maximum detail, many noisy edges are left behind as well as real details. If the cleaning process is tuned for good cleaning, some detail that consists of a couple edges is likely to be removed. On the other hand, a softer cleaning policy leaves behind a lot of noise especially in spherical areas.

The problem with the method of Telea and Jalba [7] is the lack of a local importance measure, in addition to the lack of positional constraints. Recall Figure 3.3, where the surface-skeleton nodes with the biggest importance are in the center of the bulkiest part of the object. When contraction is applied, most of the small details are pulled towards the areas with bigger importance. A global importance measure is therefore good for overall centeredness but not for preserving small details. This is also the same reason why our experiments for using the global importance for re-centering and cost mechanism for edge collapse have failed; we wanted to preserve as much detail as possible.

4.3 Smoothness and Node Sampling

In this section we look at two closely related properties. Smoothness is the quality concerning the bends in the curve skeleton. A smooth skeleton captures smoothly the bends and corners of the original shape. This means that the curve skeleton should not have sharp bends while the corresponding part of the surface has a low curvature. Node sampling is the quality of having enough skeleton nodes, to accurately represent the shape. In addition, if the nodes have similar distances between them, resembling uniform distribution, this also adds to node sampling quality. Technically, a smooth curve skeleton has good sampling of nodes, since there are more nodes needed to represent a curvy part compared to a straight one.

If the skeleton is going to be used for any shape modification purpose, such as animation, these properties become essential as the shape will be represented more accurately and the user will have more precise control. Especially in large body parts, most methods leave only few nodes, since such a part is generally not curvy. On bends, it is likely that a method may yield a sharp corner, while the original shape has a lower curvature. Examples for both cases can be seen in Figures 4.19 and 4.20.

In the remainder of this section, we first discuss our results and the factors which affect them. Next, in the comparison part, we look at two cases where sampling and smoothness might be an issue. We finish the section with a general discussion.

4.3.1 Results

In our method, the steps that directly affect the node sampling are mesh contraction and edge collapsing. If the contraction leaves a really thin mesh, then the edge-collapse step can yield a good sampling of nodes. Thicker regions require more collapses and therefore the edges in such areas travel a longer distance during the collapse step. Also, when the part of the shape is straight (as opposed to curvy), the collapse is expected to leave a sparse sampling as well, because straight parts can be represented by fewer nodes. Yet, this is prevented by utilizing sampling costs and boundary costs as stated in Section 3.2.2.

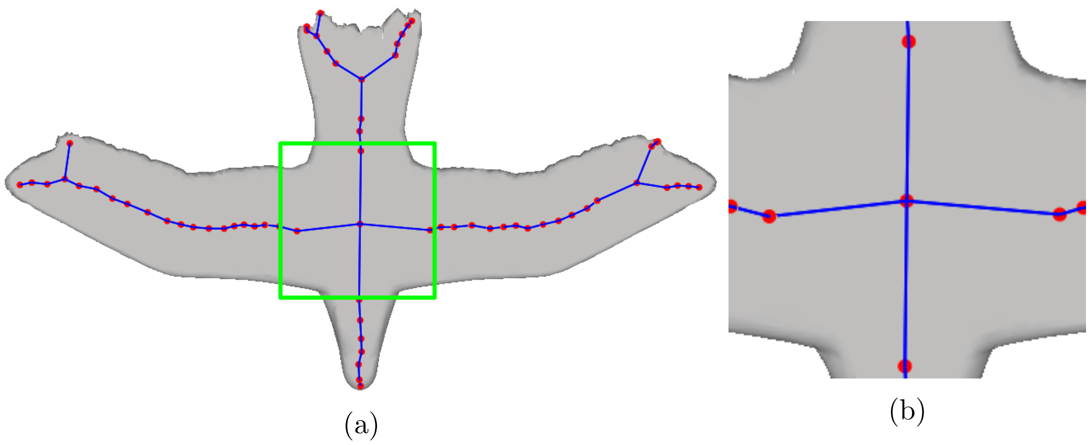


Figure 4.19: The method of Au et al. [4] applied on bird model to show long edges caused by the sparse skeleton node distribution in a bulky body section (a). Zoomed-in section (b).

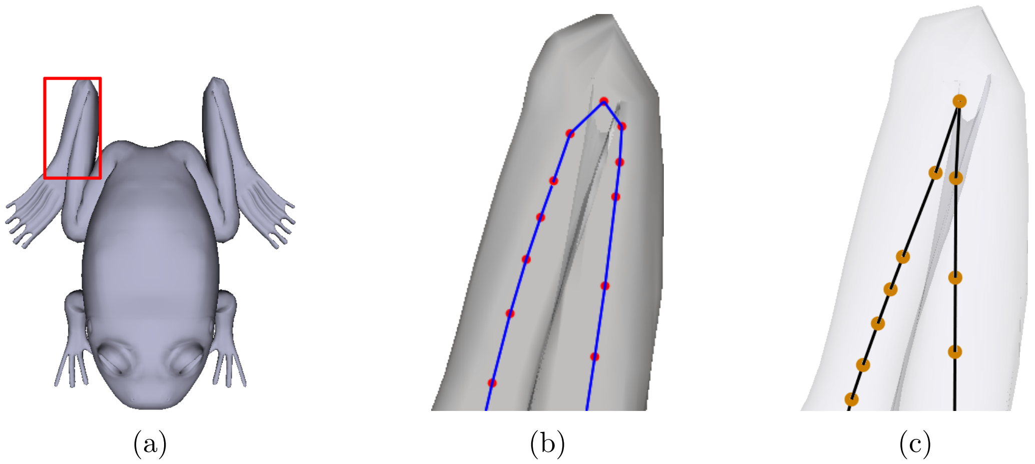


Figure 4.20: Zooming on the leg of the frog model (a) to compare a smooth corner (b) to a sharp one (c).

It is possible to obtain skeletons with different node distributions by changing the related parameters. In the contraction step, a large contraction weight (W_L) moves the vertices farther in less iterations. This means that a larger W_L can yield a thinner mesh, but this will disrupt the shape more, i.e. vertices will move more off-center. The attraction weight (W_H) can be increased to protect the details and the shape resemblance, but a larger (W_H) will result a thicker mesh, at the end. The best choice would be a balance between detail and thickness. In this section, we will use the default values for these weights and focus on the effect of the parameters in the edge-collapse step. The default values of the contraction parameters are already tuned for a good balance between shape preserving and contraction thickness.

For the edge-collapse step, we examine the effect of the elements of our cost mechanism. The first question here is if such a complex scheme is really better compared to a simple one, e.g. using the length of the edge as cost (as in [3, 5]). The next question is, if such a scheme is necessary, how do different elements of the mechanism affect the quality?

Necessity of a Complex Cost Scheme

To see if our current cost calculation for the edge-collapse step is really effective, we temporarily switched to a simpler method to make comparisons. In the simpler method, the cost of an edge is simply its length, i.e., $cost_{ij} = \|\mathbf{v_i} - \mathbf{v_j}\|$. Similar to the original mechanism, the edge with the smallest cost is removed at each iteration.

The top row of Figure 4.21 shows the result using the simpler cost mechanism for three models. The curve skeletons still resemble the original shape, especially if the shape has curvy areas as in the fertility model. However, in general, the results are too simplistic, i.e., the curve skeleton has a poor node sampling. Thus, even in a cylindrical part with a moderate curvature, the curve skeleton has many sharp corners (Figure 4.21b). Compared to the results of the original cost scheme (Figure 4.21, bottom row), the simplistic cost mechanism does not prove to be suitable for our purpose; considering the difference in level of detail preservation, sampling and smoothness between the two schemes.

Effect of the Boundary Cost

It is possible to prevent the removal of an edge that does not have a face by simply ignoring it during the candidate edge selection. However, we do not want to prevent this completely, but we want to make it less likely for this to happen. This choice makes it possible to clean unwanted extremities by regular edge collapses, if necessary. In the case that a faceless edge is chosen as a candidate, despite the boundary cost, it means that it represents an insignificant detail. This fact can be used to further refine a curve skeleton.

To see the effect of the boundary cost, we simply turn it off and on. Figure 4.21

shows significant differences between turning the boundary cost off (middle row) and on (bottom row), for three models. Even though the curve skeletons are far better than the skeletons extracted using the simple cost scheme, node sampling and smoothness are still too poor.

Effect of the Sampling Cost

The effect of the sampling cost seems small, since it is given a low weight in general, and it is dominated by the shape cost, after some iterations. To see the effect of the sampling cost, we look at the curve skeletons of the same model with different sampling cost weights. In one instance, we also turn off the shape cost completely and observe the full effect of it.

Figure 4.22 shows results in four different scenarios. In these examples, the weights in Equation 3.5, α and β, are given different values. In general, there are no drastic differences between the results. Still, in the center of the shape, where the shape gets thicker, the sampling cost does make a visible difference. With the shape cost completely turned off, Figure 4.22d has the best sampling, with small disruptions in the shape, in the central junction.

Without the sampling cost ($\beta = 0.0$) however, the nodes are distributed more poorly, even in thinner parts like the wings. In such "regular" regions as the wings, the sampling difference is also visible between Figures 4.22a-4.22c. These results show that the sampling cost indeed makes a difference. We can say that the presence of the sampling cost is important, while its value is not sensitive to fine-tuning.

4.3.2 Comparison

In general, all methods can effectively extract good-quality curve skeletons of the tube-like branches. This is also reflected by the sampling quality: parts such as legs and arms have almost always well-sampled curve skeletons. Smaller cylindrical parts like horns and fingers also have the same quality, except for the point-cloud-based methods which have trouble handling such parts. Thus, we do not focus on such areas. In the first case for the comparison, we look at the bulky body part, as thicker areas are always harder to handle.

For parts that resemble a torus, all methods yield also good results in terms of

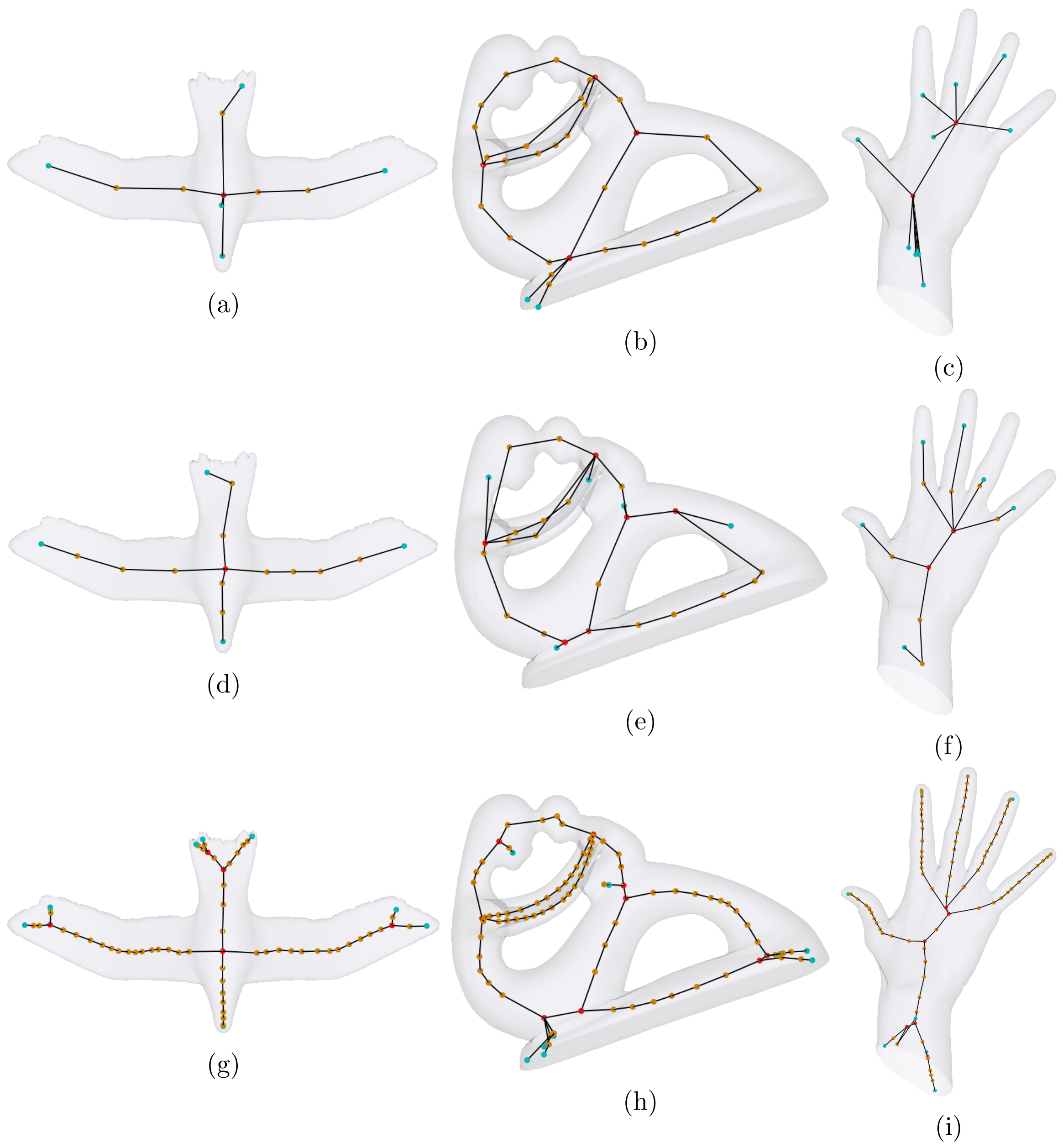
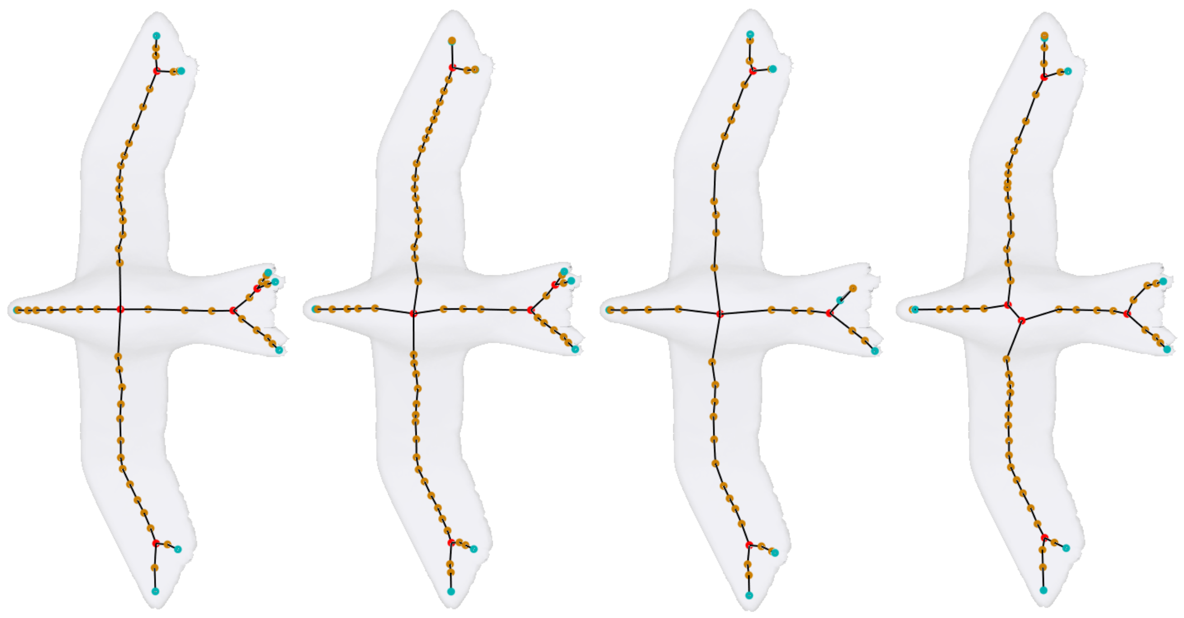


Figure 4.21: The effects of our cost scheme shown for the bird, fertility and hand models. Top row shows the results of using a different, simpler cost mechanism. Middle row shows the edge collapsing without boundary cost. Lastly, bottom row shows the result of the default cost scheme.

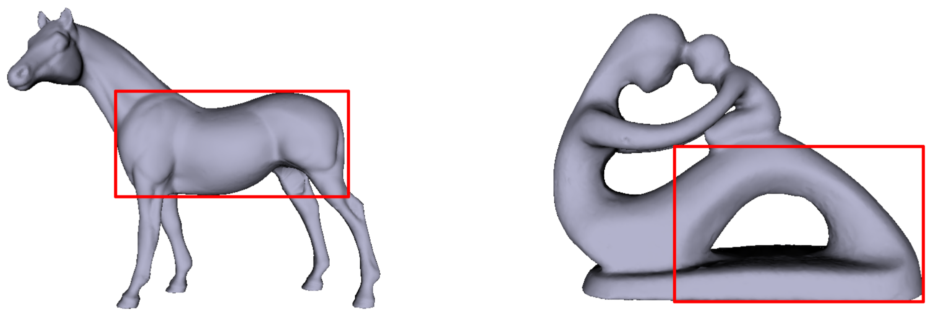
sampling quality. Yet, it is possible that the curve skeletons may have some sharp bends. Therefore, for the second case, we look at a torus-like part. After the second case, this section ends with an overall discussion of the sampling and smoothness criteria.



(a) $\alpha = 1.0, \beta = 0.2$ (b) $\alpha = 1.0, \beta = 0.7$ (c) $\alpha = 1.0, \beta = 0.0$ (d) $\alpha = 0.0, \beta = 1.0$

Figure 4.22: Curve skeletons of the bird model with different cost weights; α and β are the weights of shape and sampling cost, respectively.

The parts of the objects examined in the cases are shown in Figure 4.23.



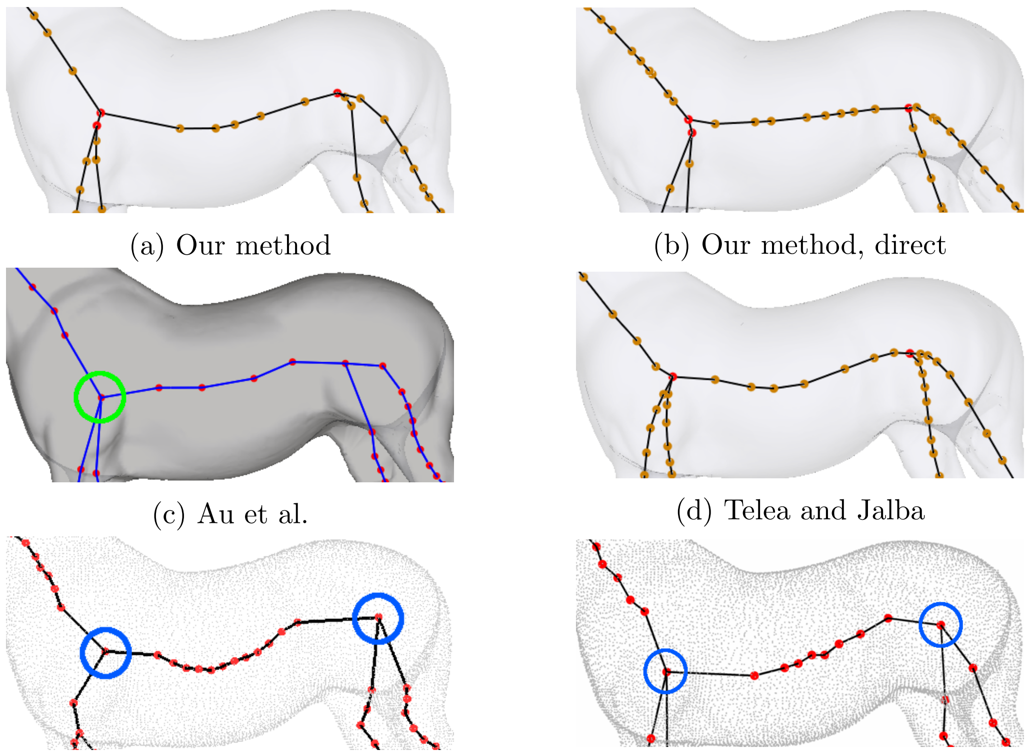
(a) Horse (b) Fertility

Figure 4.23: Zoomed-in areas of the models used.

Case 1: Bulky Body Part

In this case, we examine the body of the horse model. This part of the shape is cylindrical, so centeredness is not problematic: but it is very thick and node sampling can have issues. Another property of this body part is that it is slightly curved. As a result, poor sampling is likely to cause sharp bends, which makes observing the problems easier. Figure 4.24 shows the curve skeletons in the torso area of the horse model for each method.

In Figures 4.24a-4.24d we see a clear difference between methods that use the surface skeleton and those which do not. All four skeletons have comparable sampling of skeleton nodes, but the difference is in their smoothness. Our method and the method of Telea and Jalba [7] reflects the curvature of the body accurately. The positions of the curve-skeleton nodes correspond to the bends of the original surface. There is one very long edge in our curve skeleton, but it is in a section where curvature does not change, and therefore, the smoothness of the skeleton is not affected.


(a) Our method

(b) Our method, direct

(c) Au et al.

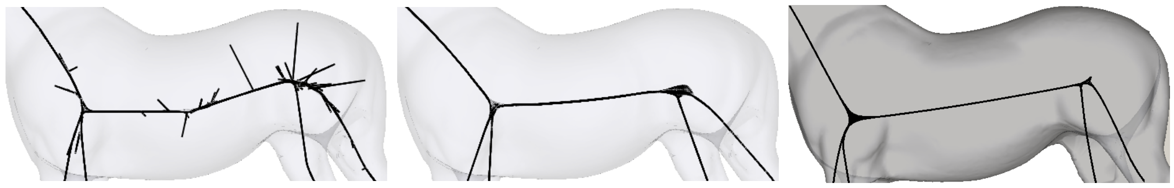
(d) Telea and Jalba

(e) Cao et al.

(f) Tagliasacchi et al.

Figure 4.24: Torso of the horse model to compare skeleton-node sampling and skeleton smoothness.

On the other hand, for the two methods which contract the mesh directly, the curvature of the body is not well-preserved. Our direct contraction yields basically a linear structure, not reflecting the bends of the surface. The method of Au et al. [4] does have bends, but these are rather sharp ones. Also, because of the inaccurate positioning of



(a) Our method (b) Our method, direct (c) Au et al.

Figure 4.25: Curve skeletons just before the edge-collapse. Bends are smoothed out when the mesh is contracted directly. Contracting from the surface skeleton helps protecting geometric surface properties.

the junction node (marked green in Figure 4.24c), the last edge before the junction fails to follow the shape of the original surface.

For both approaches, there is a reason why this difference occurs. When the mesh is contracted directly, the bends on the surface are smoothed out (Figure 4.25). This was also the reason behind not being able to preserve small surface details mentioned in Section 4.2. With a perfect re-centering process, this would not cause problems, since a perfectly-centered curve skeleton would reflect the correct curvature of the original surface. However, the re-centering step is not perfect, and once the bends are smoothed, it is very hard to bring them back precisely. When the surface skeleton is used as the intermediate stage, spiky structure of the surface-skeleton mesh prevents a very strong smoothing, since the contraction method considers spikes as details and tries to preserve them. The spikes on such a mesh generally occur if the surface has bends or bumps, hence geometric properties of the surface are preserved better.

Another method which uses contraction, by Cao et al. [5] shows potential, but the results are not completely satisfying. In Figure 4.24e, we see the curve skeleton after re-centering was applied. The resulting skeleton reflects the curvature of the surface very well in the torso, but node distribution is not decent. The edges that connect the torso skeleton to the junctions (marked blue in the figure) in the front and the back are very long. In fact, all edges connecting to the marked junctions are very long. Compared to the smooth skeletons of the previous methods, these areas look unnatural. If we look at the same curve skeleton before re-centering, we see that it has better node sampling around the junctions, but as a result of the contraction, bends are all smoothed out (Figure 4.26b). This method would have better results with a more sophisticated re-centering method. The current process uses a simple averaging using the positions of the vertices on the surface. This scheme sometimes yields curve skeletons with lots of zigzags and long edges with sharp bends, as it does in this case.

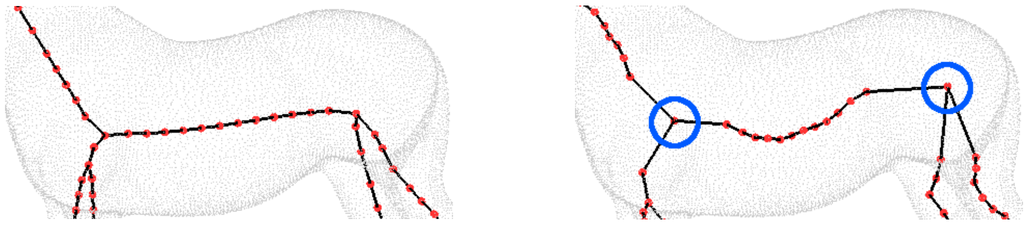

(a) Before re-centering (b) After re-centering

Figure 4.26: Curve skeleton of Cao et al. [5] before and after re-centering.

The results of Tagliasacchi et al. [6] are also similar to the results of Cao et. al. [5], especially around the junction nodes (marked blue in Figure 4.24f). Node sampling is adequate, yet the curve of the shape is not represented accurately. The curve skeleton is not as linear as in Figure 4.24b and 4.24c, and it is possible to understand the curvature of the shape, just by looking at the curve skeleton. But, the skeleton is also not as smooth as in Figure 4.24a and 4.24d: there are sharp bends even around the short edges. The long edges around the junctions are the result of the special junction handling in the algorithm. In the joint identification step, many skeletal point samples collapse to one center [6]. If the joint area is in a large part, as in this case, this influences a larger area, and there we see the large edges around the junction node.

Case 2: Torus-like Body Part

In this case, we look at the fertility model and focus on the area around the hole, as marked in Figure 4.23b. What makes this part of the object interesting is that here the curve skeletons also have at least two junction nodes (three for some methods). In addition, the torus is quite thick including the joint regions, which makes it harder to preserve smoothness.

Among the four mesh-based methods, the skeleton of Au et al. [4] has the sparsest sampling of nodes (Figure 4.27c). This causes several long edges and sharp bends. Long edges mostly occur around the junctions while sharp edges are visible in other parts as well. Our results show better sampling of skeletal nodes while still having long edges around junctions (Figure 4.27a).

Our method with direct contraction and the method of Telea and Jalba [7] have the best distribution of nodes and the smoothest curves (Figures 4.27b and 4.27d). Even though these, too, suffer from the presence of long edges in non-junction parts, the curvature of the surface is better preserved.

The results of Tagliasacchi et al. [6] and Cao et. al. [5] are very similar to the previous case, in terms of quality. The result of Cao et. al., given in Figure 4.27e, is the curve skeleton before re-centering is applied, since re-centering step reduces the quality of the skeleton in terms of node distribution and/or smoothness, same as in Figure 4.26.

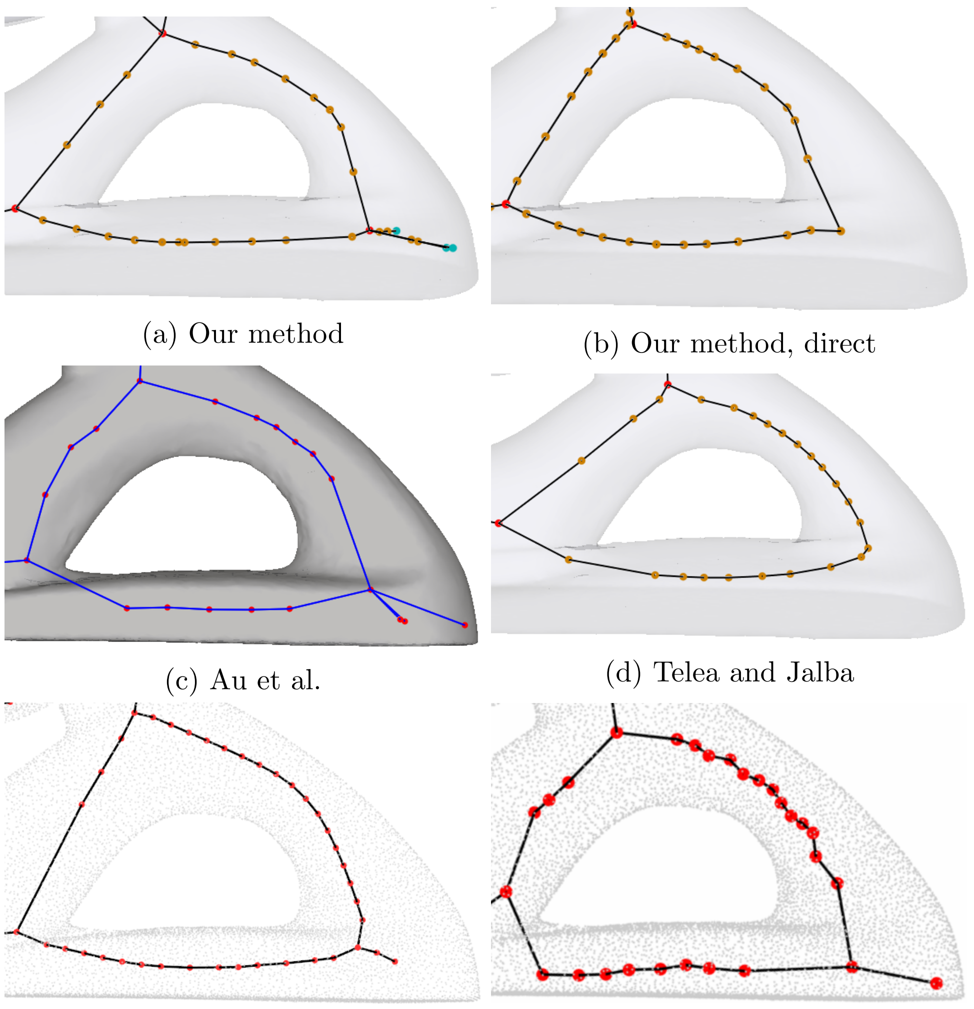
For ROSA, we see that there are again long edges around the junctions. Another similarity to the previous case is that the curvature is captured nicely in non-junction areas, but there are small zigzags visible (Figure 4.27f). These are related to the sampling radius parameter used for extracting the curve skeleton nodes. If this is given a bigger value, curves have long edges and sharp bends. If a smaller value is chosen, then it sometimes causes skeleton nodes to be too close to each other in curvy areas. A smaller value also increases the chance of capturing details, hence a small radius is often preferred over a large one.

General Discussion

For all methods, one common issue is the handling of junctions, especially for those junctions where thicker branches join together. Thickness, in general, tends to cause a sparse distribution of skeletal nodes. If the shape does not have bends, a sparse distribution is acceptable. But when curves are involved, a dense sampling yields a smoother curve that looks more natural.

In terms of using the surface skeleton, we can say it also improves the results in terms of node sampling and smoothness. There are significant differences in the results of our method with and without the surface skeleton, as well as in the two methods which use surface skeletons (ours and that of Telea and Jalba [7]) and those which do not. Sampling-wise, all methods yield similar results, but in terms of curve smoothness, using the surface skeleton proves useful.

This usefulness comes from the formal centeredness of the surface skeleton. As mentioned before, it prevents the vertices to be pulled too far away from the center. The spiky surface of the surface skeleton mesh also prevents the curvature to be smoothed out. Therefore, even without re-centering, the curve skeleton has smooth curves instead



(a) Our method

(b) Our method, direct

(c) Au et al.

(d) Telea and Jalba

(e) Cao et al.

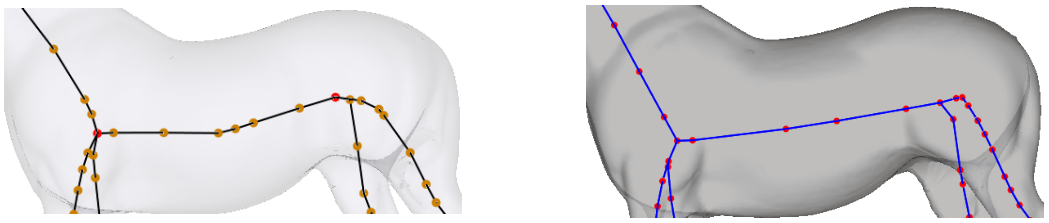
(f) Tagliasacchi et al.

Figure 4.27: Torso of the horse model to compare skeleton node sampling and skeleton smoothness.

of sharp bends. Figure 4.28 compares our curve skeleton to Au et.al's, without re-centering.

4.4 Independence From Mesh Sampling

Being independent from mesh sampling means that the curve skeletons extracted from meshes with different sampling are visually similar to each other. Similarity can also be compared in terms of other criteria that we have discussed so far (Sections 4.1-4.3).



(a) Our method (b) Au et al.

Figure 4.28: Comparing the smoothness of the curve skeletons without re-centering. Our method uses the surface-skeleton mesh as intermediate medium, while the other method does not.

Some qualities such as skeletal node sampling may naturally depend on mesh sampling, if the curve skeletons are the remaining product of the original mesh after a series of procedures, as in [4, 7]. This means that the nodes and edges of the curve skeletons have once belonged to the original mesh. On the other hand, criteria such as centeredness can be compared with respect to mesh sampling.

Similar to the previous sections, here we first discuss our results, and at the end, make comparisons. Results of the methods are shown for two models which have various details as well as bulky sections (Figure 4.29). These models and the sampling values are:

- Dragon:

  Small - S: 14K vertices, 25K faces

  Medium - M: 58K vertices, 115K faces

  Large - L: 231K vertices, 463K faces

- Lucy:

  Small - S: 10K vertices, 20K faces

  Medium - M: 50K vertices, 100K faces

  Large - L: 100K vertices, 200K faces

For the comparison, the point-cloud-based methods ([5] and [6]) are not included for two reasons: First, the applications of those methods are written in MATLAB platform and for large meshes, the execution time is not feasible (ROSA may run for days for a model whereas the others run for minutes). The second reason is that the curve skeletons

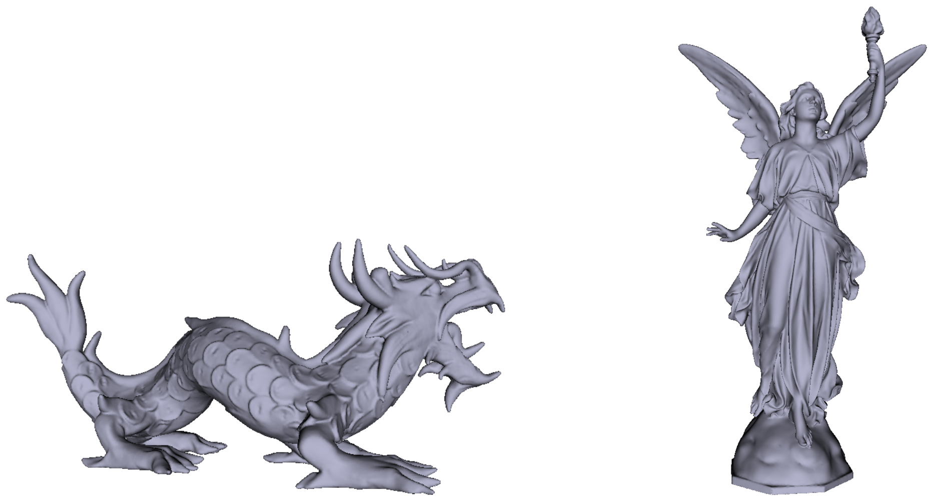

(a) Dragon (b) Lucy

Figure 4.29: The models used for the results.

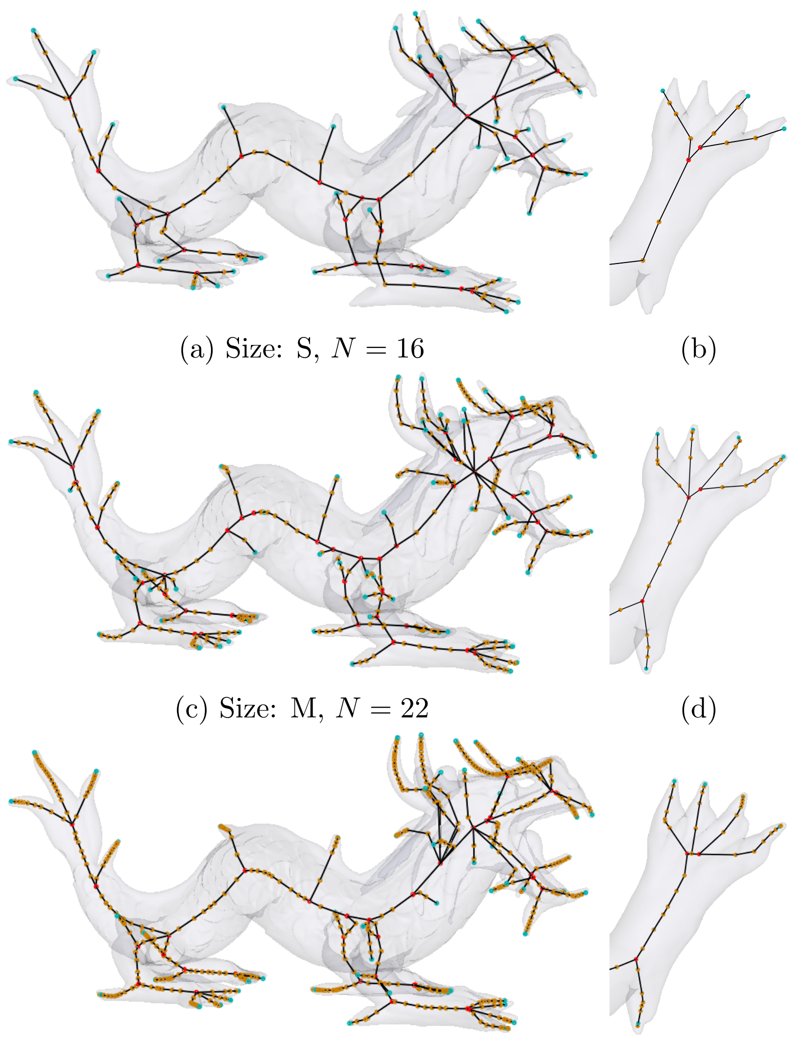
extracted by those methods are independent from mesh sampling by definition and the skeletons are not subsets of the input meshes. Hence, concerning these two methods, the quality of the result is more parameter-dependent than sampling-dependent.

4.4.1 Results

Figure 4.30 shows the curve skeletons of the dragon model, as well as the zoomed in foot, for detail comparison. The first visible fact is that the node samplings are different. As mentioned, this is due to the fact that the curve skeleton represents a processed subset of the original mesh. More vertices of the input mesh yield more points in the surface skeleton, and thus, more nodes remain in the curve skeleton.

Another visible difference is the loss of detail and inaccurate centering concerning the skeleton of the S-size mesh (Figure 4.30b). Such a big difference is not visible between the M and L-size models. This shows that these two criteria are sensitive to the mesh sampling if the sampling is very sparse.

The inaccuracy in centering stems from the over-contraction of the mesh. There are less vertices in the skeleton mesh, even though the volume is the same. Bulky regions still require some number of iterations while smaller parts require less. Also, sparsity of the mesh weakens the attraction constraint as well. Therefore, these reasons make preserving the positions of the vertices harder. The re-centering step is expected to solve


(a) Size: S, $N = 16$ (b)

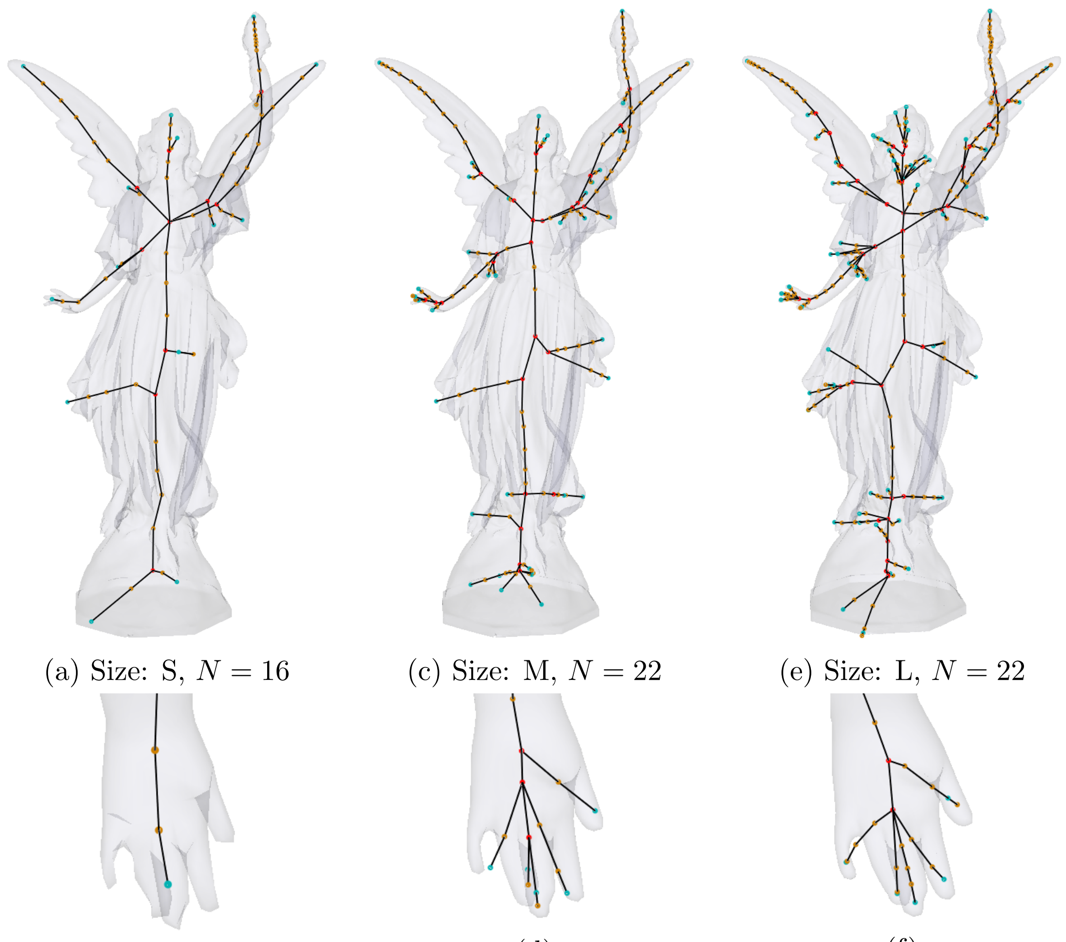
(c) Size: M, $N = 22$ (d)

(e) Size: L, $N = 25$ (f)

Figure 4.30: Curve skeletons of the dragon model with 3 different sampling values, extracted by our main method. Top to bottom: S, M and L. N is the number of iterations used for contracting the mesh.

this problem, but it also gets affected by the sparse sampling, and therefore it is not strong enough to pull the nodes back in the correct positions.

In terms of detail, it all comes down to the edge-collapse and mesh-contraction steps. Just before the edge collapsing starts, contracted meshes with different sampling values will have similar volumes and total areas of the faces, while the number of vertices and faces are different. Hence, there will be less nodes remaining for a smaller mesh after the edge collapsing is finished, as there are less edges which have bigger average length.



(a) Size: S, $N = 16$ (c) Size: M, $N = 22$ (e) Size: L, $N = 22$

(b) (d) (f)

Figure 4.31: Curve skeletons of the Lucy model with 3 different sampling values: S (a-b), M (c-d) and L (e-f), extracted by our main method. N is the number of iterations used for contracting the mesh.

Together with the over-contraction of the smaller parts, loss of detail is then the expected outcome.

The differences mentioned so far are mostly related to the S-size mesh. The larger meshes have similar qualities in all criteria except in the sampling of skeletal nodes. As claimed before, the difference in centeredness and detail preservation is reduced after a certain low value of mesh sampling is reached. Still, we can see that the neck of the L-size dragon has sharper bends compared to the neck of the M-size version. This is caused by the attraction constraints of the mesh-contraction step. For larger meshes, the contraction step naturally requires more iterations to reach the same level of thickness.

But for very large meshes, the bulky parts cannot be contracted as thin as the smaller meshes. As noted in Section 4.3, long edges are a natural result of applying edge collapses to a thick region.

For the Lucy model, we see exactly the same differences between different sizes (Figure 4.31): many fine details are preserved in M and L-size meshes, while the S-size mesh does not even preserve the fingers of the hand (Figure 4.31b).

4.4.2 Comparison

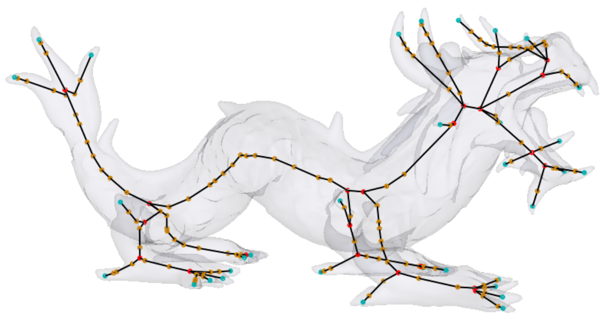
Our Method with Direct Contraction

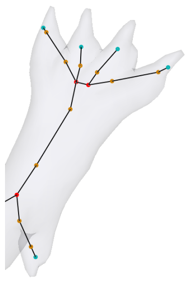
If we look at the results of our method with direct contraction of the surface mesh (Figures 4.32 and 4.33), we see similar differences between the different size meshes, concerning curve-skeleton node sampling. The difference in level of detail between the different sizes of the mesh is still visible, albeit the difference is rather small. If we compare the foot of the dragon in Figures 4.30b and 4.32b, we see that direct contraction preserved some of the fingers, even though the larger-size models do this more accurately. If we look at the hand of Lucy in Figure 4.33b, we see that the fingers are lost. This shows that smaller sizes still have relatively low detail compared to the larger ones, but the amount of detail changes from mesh to mesh.

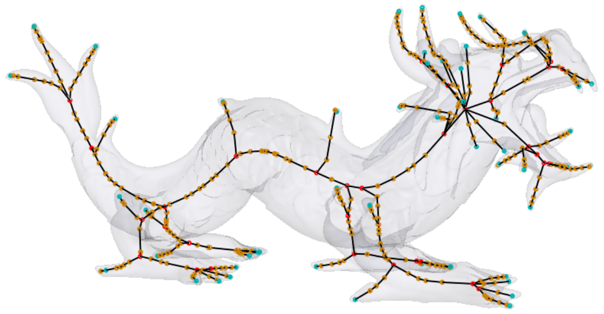
One interesting fact about this method is that it extracts curve skeletons with near-perfect smoothness in thicker areas, as in the neck of the dragon. The skeletons of M and L-size meshes have excellent sampling and accurate representation of the curve of the body. Our original method has a worse result in this part. The fact is that the original mesh can be smoothed easily whereas the surface-skeleton mesh is harder to contract. This is a result of the spiky nature of the surface-skeleton mesh. Although this property helps our method to preserve more details, in this case it also prevents the contraction of regular surface parts due to the strong attraction constraints. The same difference is also visible for the body of the Lucy model, shown in Figures 4.31c and 4.33c.

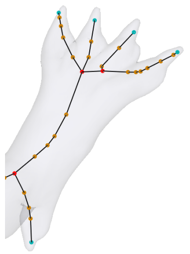
The Method of Au et al.

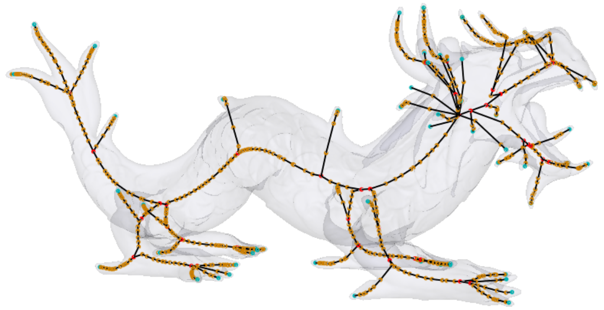
This method produces skeletons with significantly larger differences in smoothness between the M and L-size models (Figures 4.34c and 4.34e), which was not the case for


(a) Size: S, $N = 17$


(b)


(c) Size: M, $N = 22$


(d)


(e) Size: L, $N = 27$

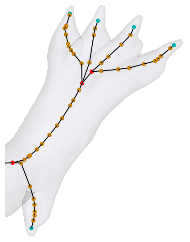
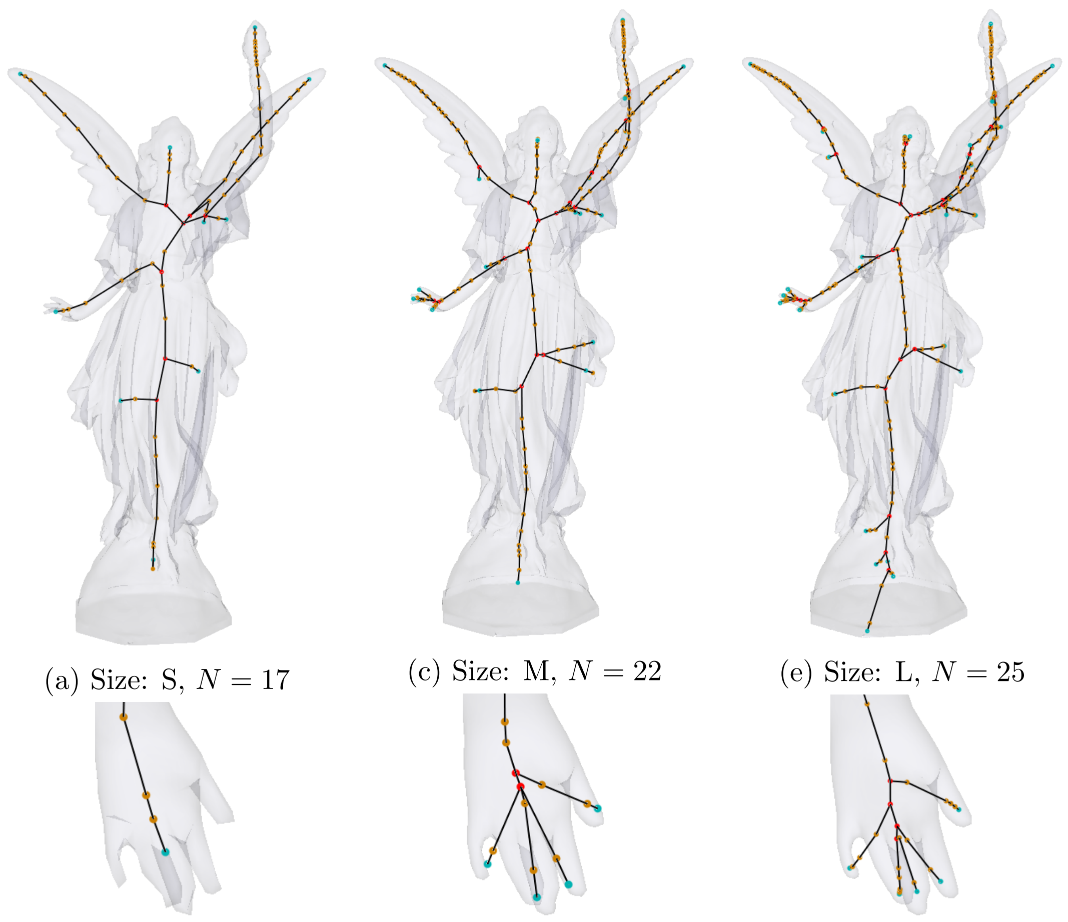

(f)

Figure 4.32: Curve skeletons of the dragon model with 3 different sampling values extracted by our method with direct contraction. Top to bottom: S, M and L. N is the number of iterations used for contracting the mesh.

our method. Despite this, the difference in detail between different sizes is also the same for this method.

One interesting result here is that the curve skeletons of the Lucy model show a slight difference in skeletal structure. The main part of the body is structured differently for each size, with a bigger difference between the S and M-size meshes (Figure 4.35). We consider geometric similarity more important while judging the independence from mesh sampling, since there are natural reasons why other criteria may be dependent on the



(a) Size: S, $N = 17$ (c) Size: M, $N = 22$ (e) Size: L, $N = 25$

(b) (d) (f)

Figure 4.33: Curve skeletons of the Lucy model with 3 different sampling values: S (a-b), M (c-d) and L (e-f). Extracted by our method with direct contraction. N is the number of iterations used for contracting the mesh.

sampling. The joint-merging step in the embedding refinement process described in Section 2.1.3 is the main reason between this structural difference. In general, merging neighbor junctions results in visually pleasant outcomes, but in cases like Lucy's body, this process results in curve skeletons which look different. This can have undesirable results in shape processing applications such as segmentation, where this kind of difference might result in different segmentations for the same object.

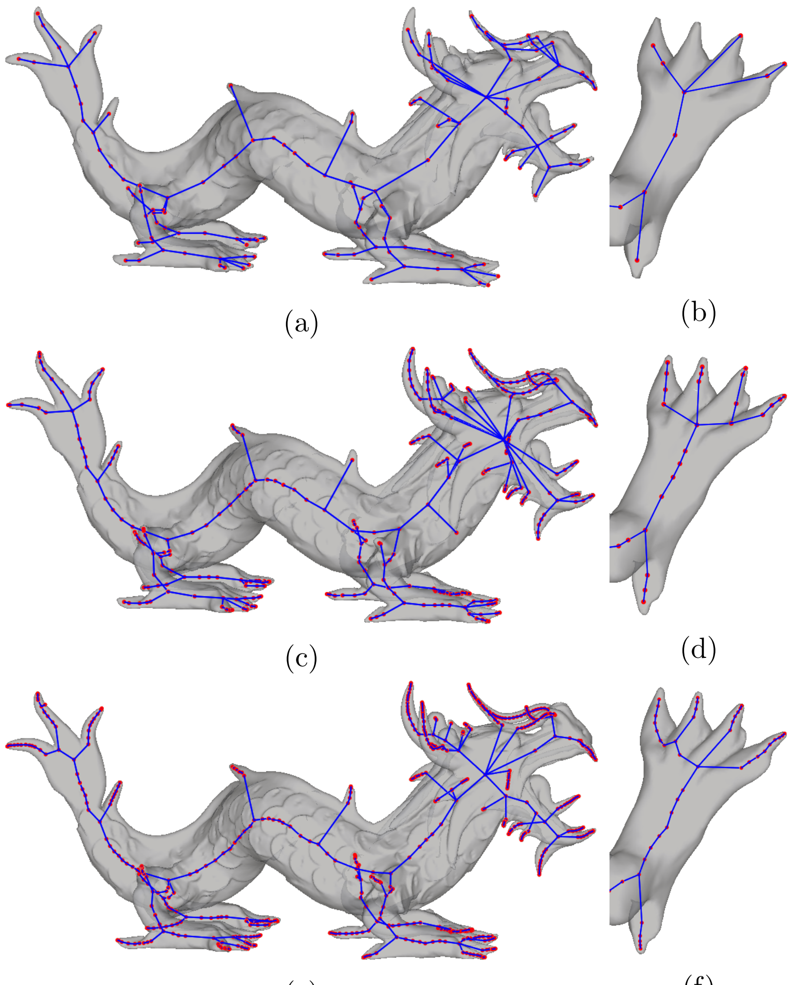


Figure 4.34: Curve skeletons of the dragon model with 3 different sampling values extracted by the method of Au et al. [4]. Top to bottom: S, M and L.

The Method of Telea and Jalba

This method turned out to be the most sensitive to mesh sampling. Although skeletons resemble each other in terms of posture, in terms of other criteria, the results vary a lot. For sparse meshes, this method sometimes cannot even preserve main parts of the shape, or completely misplaces some vertices. In Figures 4.36b and 4.37b, it is shown that the method of Telea and Jalba [7] preserves the least amount of detail. For denser versions of the meshes the results get better, even if they are still not good enough compared to the results of the other methods.

Aside from the small details, sensitivity to sampling can be seen in some main parts

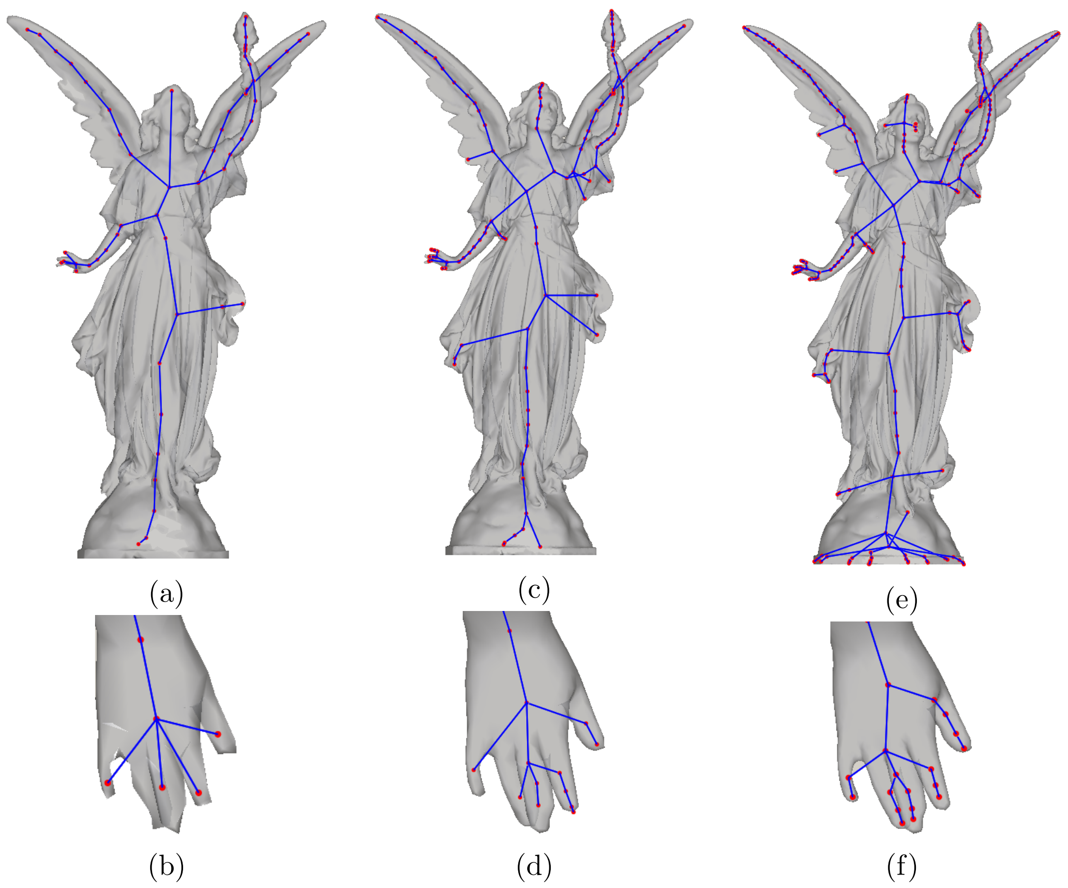
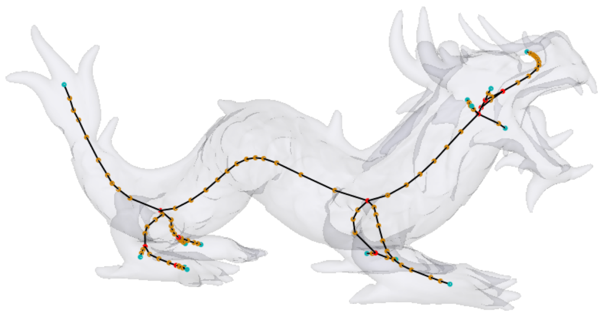


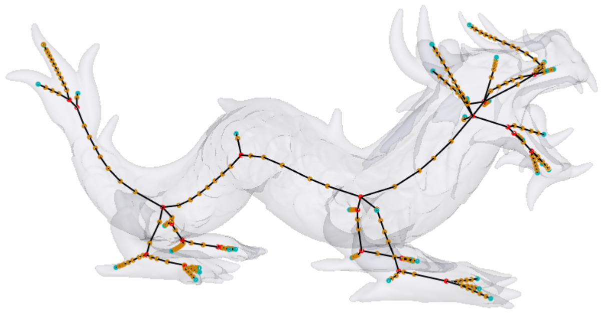
Figure 4.35: Curve skeletons of the Lucy model with 3 different sampling values: S (a-b), M (c-d) and L (e-f). Extracted by the method of Au et al. [4].

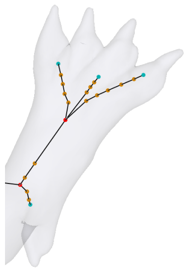
as well. The head part of the dragon and all the extremities of Lucy are pulled inwards and are significantly shorter compared to the skeletons of the larger meshes. This is a more serious issue as these are different skeletons which are only similar on the central parts. This result shows the weakness of the method the clearest, as the shortened parts prove that the vertices indeed go towards the global centers of the shape, where the importance measure is higher.

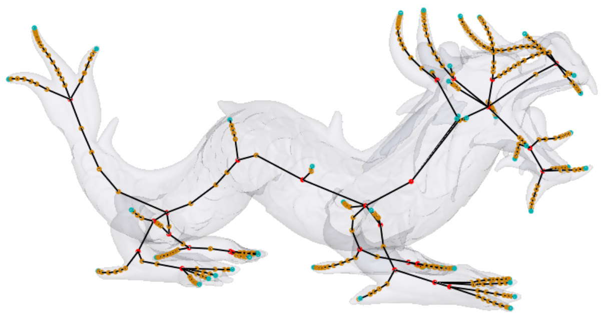
Since poor mesh sampling also causes a sparse surface skeleton point cloud and therefore a sparse gradient field for the advection process, the accuracy of centeredness also decreases. We have seen this problem earlier in the leg of the frog model (Figure 4.10). There, the overall mesh sampling is dense enough, but locally, the sampling in the leg is poor. When the sampling is globally poor, the inaccuracy naturally gets more drastic. If there are no surface-skeleton points with high importance nearby, the mesh vertices


(a) Size: S, $N = 0.5K$

(b)


(c) Size: M, $N = 1.1K$


(d)


(e) Size: L, $N = 2.5K$

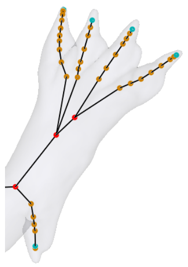
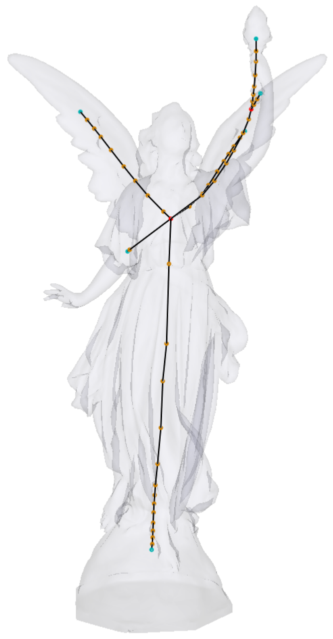
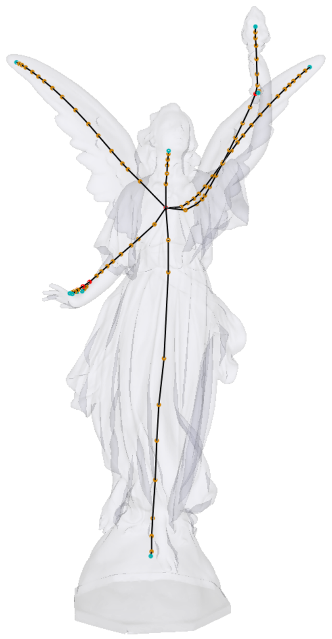

(f)

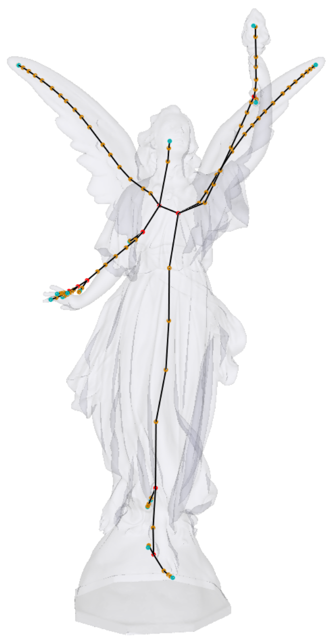
Figure 4.36: Curve skeletons of the dragon model with 3 different sampling values extracted by the method of Telea and Jalba [7]. Top to bottom: S, M and L. N is the number of iterations used for contracting the mesh.

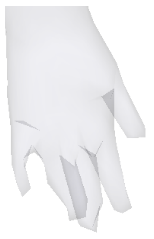
are pulled towards the nearest high-importance surface skeleton point, since the gradient vector will point to there.

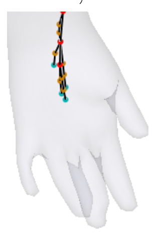
For the improved version of this method, the differences between the different sizes are the same. There is of course less loss of detail concerning the smaller sizes, but these discussions have already been made in the previous sections, i.e., in Section 4.1-4.3.


(a) Size: S, $N = 0.5K$


(c) Size: M, $N = 1.1K$


(e) Size: L, $N = 2.0K$


(b)


(d)

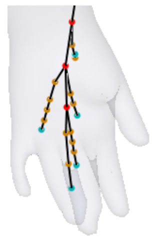

(f)

Figure 4.37: Curve skeletons of the Lucy model with 3 different sampling values: S (a-b), M (c-d) and L (e-f). Extracted by the method of Telea and Jalba [7]. N is the number of iterations used for contracting the mesh.

General Discussion

In conclusion, all methods except the method of Telea and Jalba [7] are quite insensitive to mesh sampling, in terms of skeleton structure and posture. There is a connection between the sampling of the curve skeleton nodes and the mesh sampling, which is natural, considering that all methods reduce the mesh to a curve skeleton. Hence the skeleton is a subset of the original surface.

If we look at the effect of using the surface skeleton, we see that even though there is significant diversity concerning other criteria discussed in the previous sections, in terms of mesh sampling, there is no clear difference.

Chapter 5

Future Work

We have also considered an application area which has not been previously associated with curve skeletons: surface parameterization. The main idea is to use the curve skeleton and the information provided by the surface-skeleton point cloud to build a base for parameterizing 3D surfaces. Although it is still incomplete, the progress accomplished so far seems promising.

In this chapter, we first describe what surface parameterization is. Next, we describe the steps we have taken so far, and discuss how the curve skeleton can be useful.

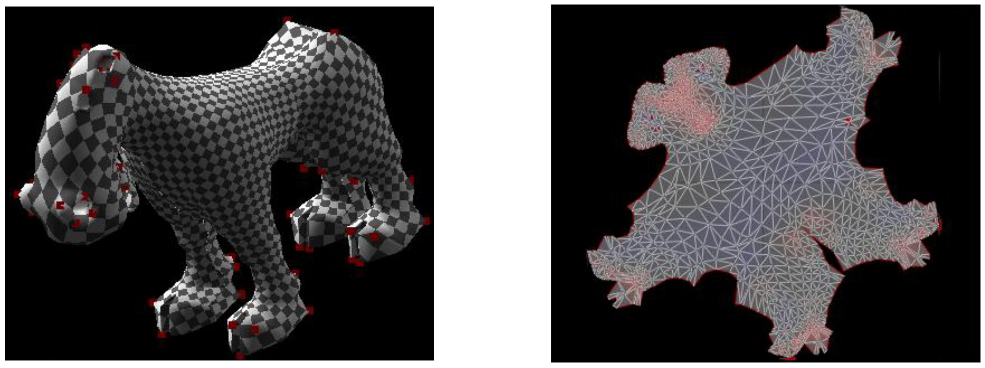
5.1 Surface Parameterization

Surface parameterization aims at establishing a one-to-one mapping ψ between a parameter domain and a surface. In computer graphics, this is mostly done by cutting a 3D mesh into patches and unfolding these patches onto another surface, generally a 2D plane (Figure 5.1). It is mostly seen in the form of texture mapping, where information such as color is stored on a texture map and mapped back on the surface when needed.

The mapping $\psi : M \to S$, where $M \subset \mathbb{R}^3$ and $S \subset \mathbb{R}^2$, is defined as:

$$(x, y, z) \xrightarrow{\psi} (u, v). \tag{5.1}$$

Aside form the use in computer graphics, the most popular application of surface parameterization is map projection. Figure 5.2 shows four different methods for map projection. As clearly visible, every method distorts some properties while preserving



(a) (b)

Figure 5.1: An example of cutting a mesh and unfolding it onto a plane. We want to use the curve skeleton to find good cut-points and cut-paths [13].

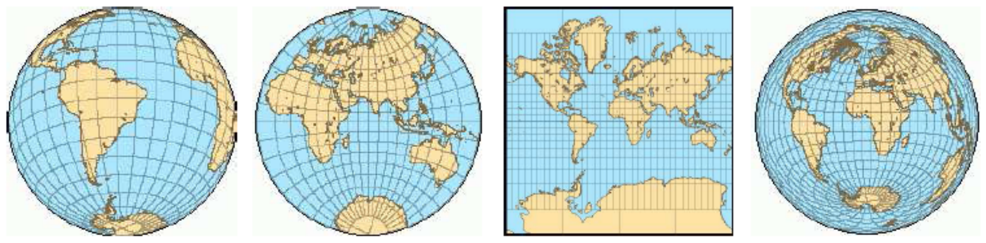
others. In this example, the four methods used have the following properties [14]:

- **Orthographic projection** preserves the directions from the center, but distorts both angles and areas.
- **Stereographic projection** preserves the angles while distorting areas. A mapping that preserves angles is also called *conformal mapping*.
- **Mercator projection** distorts both angles and areas, but it draws a navigational bearing as a straight line, hence it is very useful for naval navigation.
- **Lambert projection** preserves the angles with the expense of angles. A mapping which preserves areas is called *equiareal mapping*.

If a mapping preserves both angles and areas, it is called *isometric mapping*. Ideally, all mappings try to approximate isometry but only developable surfaces (e.g. a cylinder) can be mapped onto a plane isometrically. Therefore, depending on the purpose of the application, mapping methods try to find a balance concerning which property to preserve with what degree.

5.2 Completed Steps

We have considered finding cut-lines on the surface using the curve skeleton and the surface-skeleton point cloud. In a nutshell, the process can be described as follows:



(a) Orthographic (b) Stereographic (c) Mercator (d) Lambert

Figure 5.2: Four different projection methods [14].

- The cut-path follows the curve skeleton.
- Two cut-points on the surface, i.e. feature points, are found for each curve-skeleton node. Surface-skeleton points and their feature points are used for this purpose (see Section 5.2.1).
- A shortest geodesic ring for the feature points of each curve skeleton is calculated. This ring consists of the shortest geodesic between the feature points of a curve skeleton node, and another geodesic in the other direction.
- Later, another set of geodesic rings, which intersect perpendicularly with the feature point geodesics are calculated. These should be equally apart from each other.

So far, we have completed all steps but the last one. We already had the surface skeleton and the curve skeleton, therefore advancing in this direction was smooth. Still, there are problems that need to be solved as well as another step that needs to be completed. The details of how we utilized our methods are described next. The section ends with a view of the current situation and a road map for the future.

5.2.1 Finding the Cut-Points

Initially we have tried to find the nearest surface skeleton point **p** for each curve skeleton node **v** and use the two feature points of **p** as the cut-points. However, this was unsuccessful because of two reasons:

1. Sampling of the curve-skeleton nodes was not dense enough. As examined in Section 4.3, bulky parts of the objects have a sparse distribution of skeletal nodes. For a more accurate cut-path, we needed more cut-points and more geodesics.

2. Two feature points of a surface-skeleton point do not always form a straight (or close to straight) angle α between them (Figure 5.3). As a result, cut-points did not always follow a smooth line, especially around the tip and junction nodes (Figure 5.4). For a better cut, we needed the cut-points to be aligned in a "line" as much as possible.

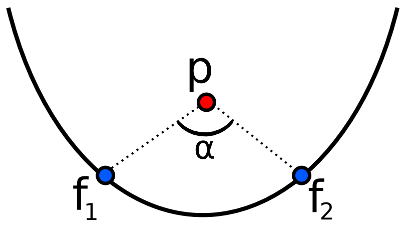


Figure 5.3: A surface skeleton point (**p**) and its feature points (**f₁** and **f₂**) that form an angle $\alpha < 180°$.

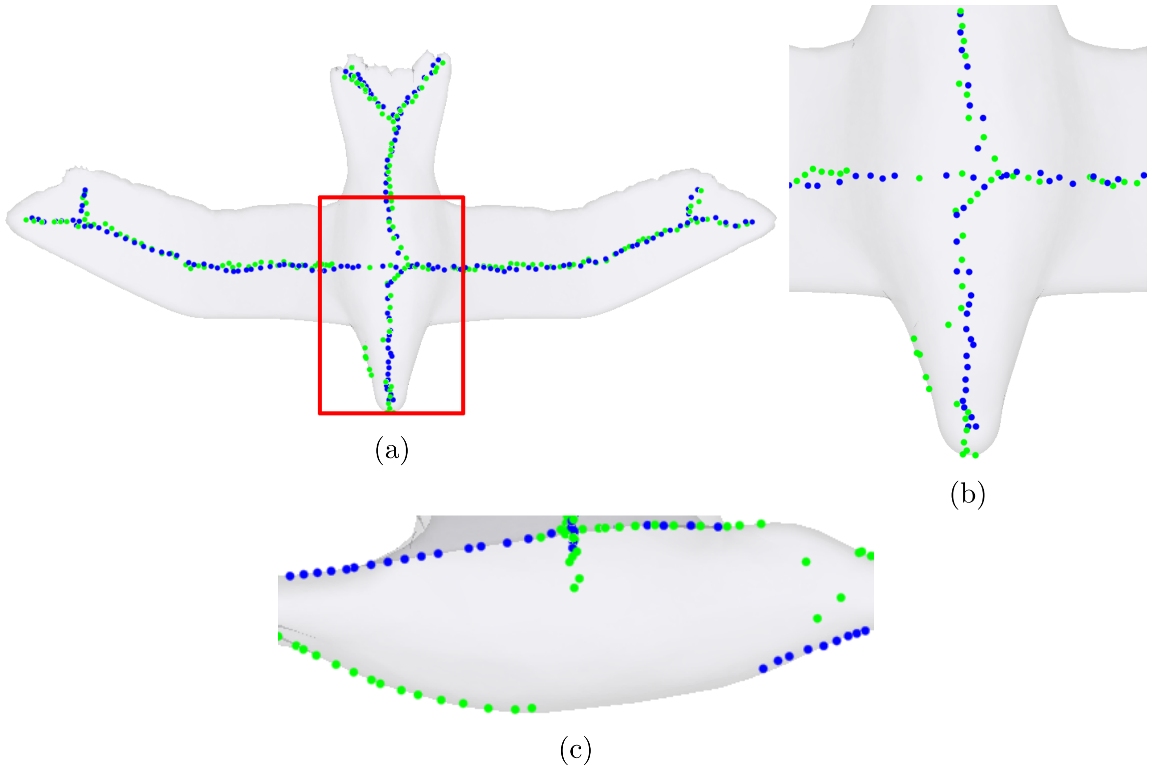


Figure 5.4: Cut points of the bird model, initial method (a). Problematic areas: junctions and tips (b). There are no cut-points on the below side, because all feature points are on the other side (c).

Solution 1: Re-sampling the Curve Skeleton

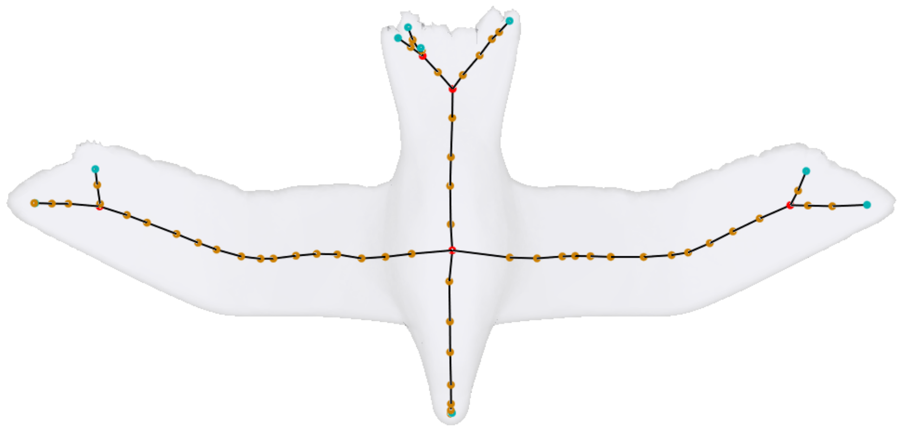
In order to get more curve skeleton nodes, we simply re-sampled the nodes. This yielded a dense distribution of nodes which are distributed almost equidistantly away from each other.

The pseudo-code for re-sampling skeleton nodes is given in Algorithm 11. It takes as parameter a length threshold ℓ, and it traverses the edge set, splitting them when edges longer than ℓ are found. Traversing starts from a junction node and goes towards other junctions or tip nodes. If an edge is longer than ℓ, it is divided into smaller pieces of equal length, where the length of each new piece is smaller or equal to ℓ. The split operation (line 14) keeps the two original vertices of the edge in place and creates new ones in-between them.

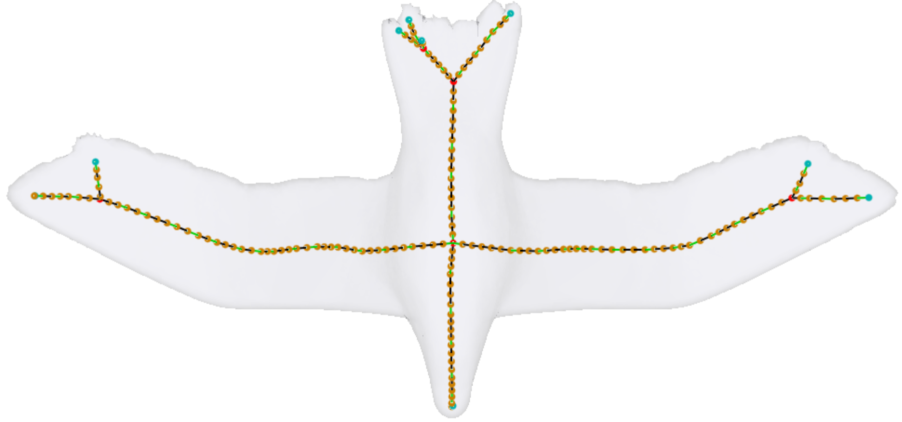
Algorithm 11 Re-sampling the nodes of the curve skeleton

```
 1: procedure ResampleSkeleton($V, E, \ell$)
 2:     for all Vertex $\mathbf{v} \in V$ do
 3:         if $\mathbf{v}.edges.size > 2$ then                    ▷ Start from junction node
 4:             for all Edge $e \in \mathbf{v}.edges$ do
 5:                 if $e.visited = true$ then
 6:                     continue;
 7:                 end if
 8:                 $e.visited \leftarrow true$
 9:                 Vertex $\mathbf{s_1} \leftarrow \mathbf{v}$
10:                 while true do
11:                     Vertex $s_2 \leftarrow s_1 = e.v_1$ ? $e.v_2$ : $e.v_1$
12:                     if $\|e\| > \ell$ then
13:                         $n = \lceil \|e\|/\ell \rceil$                  ▷ How many pieces?
14:                         $SplitEdge(e, n)$
15:                     end if
16:                     if $s_2.edges.size > 2 \vee s_2.edges.size = 1$ then
17:                         break;                             ▷ Reached junction or tip
18:                     end if
19:                     $s_1 \leftarrow s_2$
20:                 end while
21:             end for
22:         end if
23:     end for
24: end procedure
```

In the algorithm, parameters V, E and ℓ represent the set of curve skeleton nodes, edges and the threshold length, respectively. We set ℓ to be a factor of the average length of the edges in E. The curve skeleton before and after re-sampling is shown in Figure 5.5 for the bird model.



(a) Before, 67 vertices



(b) After, 167 vertices

Figure 5.5: Curve skeleton before and after re-sampling. Here, ℓ is set to half of the average edge length.

Solution 2: Finding Better Feature Point Pairs

A perfect pair of feature points form a straight angle with the skeleton point (Figure 5.3 shows the opposite case). This is not always possible, but getting close to a straight angle is also acceptable. However, surface skeleton points with small geodesic importance (Section 3.1.2) may have a pair that makes a small angle (Figure 5.3). Another situation is when the surface has bumps, as this may also cause smaller angle between the feature points. Because of a bump, the ball-shrinking algorithm (Algorithm 1) will pick another feature point nearby, yielding a smaller α angle. These are completely expected facts, but since we want the cut-points to be aligned on a line, it is required to enforce a larger angle between them.

In order to find good feature point pairs for the curve-skeleton nodes, we use the data stored in the surface-skeleton points. After the re-sampling, we find the k nearest neighbors of a curve-skeleton node $\mathbf{v}$ from the surface-skeleton point cloud, instead of the closest one. This gives us $2k$ feature points. Among those, we first select the one closest to $\mathbf{v}$ as the first feature point $\mathbf{f_1}$. Next, for the second feature point $\mathbf{f_2}$, we select the one which forms the largest angle with $\mathbf{v}$ and $\mathbf{f_1}$. This second feature point might not be the closest one, but since we are looking at the feature points of the nearest neighbors, the proximity is still small enough. The results are shown in Figure 5.6. Notice the improvement compared to Figure 5.4, especially around the junction in the chest of the bird.

5.2.2 The End Product

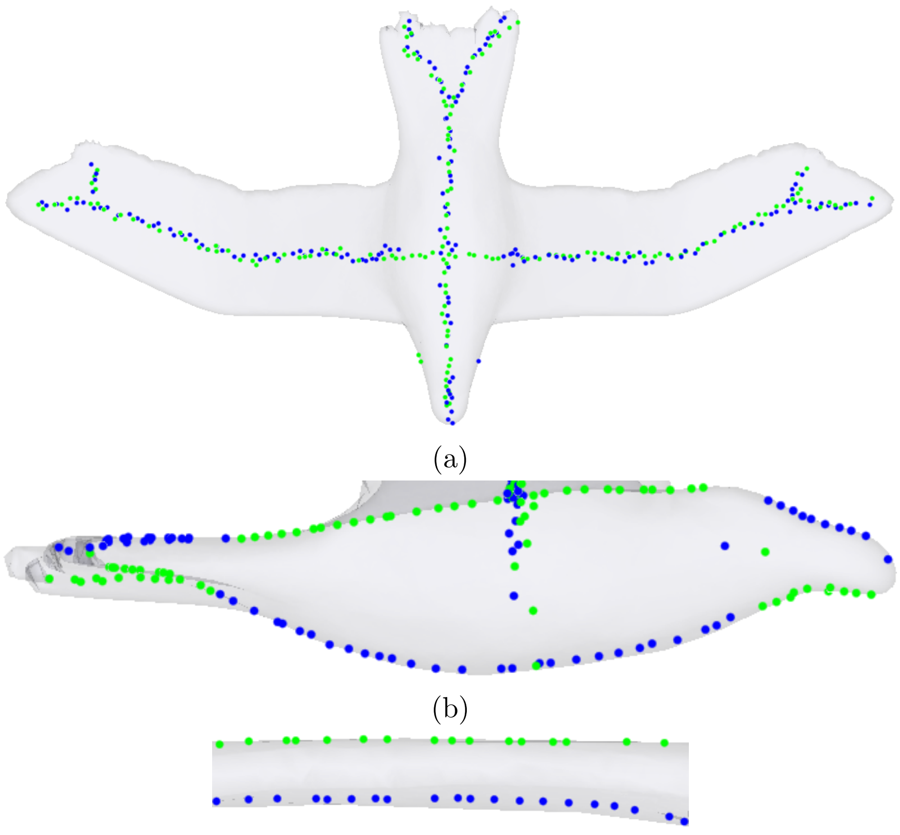
After building a more effective mechanism for finding the cut-points on the surface, next step is to find the geodesics. We calculate the shortest geodesic between the pair of the feature points for each curve skeleton node using a similar mechanism as in Section 3.1.2: sending traces starting from the tangent vector $\mathbf{t_1}$ of $\mathbf{f_1}$. This yields a half-ring, therefore we search for the second one using the same method, but starting the traces around $-\mathbf{t_1}$, i.e. towards to other direction.

After the shortest geodesic calculation, the last point we have reached is shown in Figure 5.7.

5.2.3 Discussion and Roadmap

The framework for parameterization has now a solid base, but there are still some issues to be solved. Cylindrical branches of the objects are handled well, as is the case with the curve-skeleton extraction. The problematic areas are still tips and junctions. Although the solutions have been improved a lot, the results still differ depending on the geometry of the shape.

For the bird model shown in Figure 5.7, cut-points and geodesics look good on the



(c)

Figure 5.6: Cut points of the bird model (a). Looking from a different angle (b). Looking at a part of the right wing (c).

main body and the wings, but at the tail and the tip of the wings, geodesics may intersect with each other. For an effective method, they are expected to be parallel (or close to parallel) to each other. This is actually a result of the method which we use to assign feature points to curve-skeleton nodes, where we enforce big angles as well as short distances. This enforce mechanism yields really good results in bulky junctions, but for triangle-shaped cones, e.g. the tip of the wings, quality is reduced. as there is no mechanism to check geodesic intersection.

If we look at more examples, we see the same issues more clearly. In Figure 5.8, the Neptune and horse models are shown with their geodesics. Cylindrical areas (e.g. arms, legs, spear) have near-perfect geodesics: there is no intersection between the geodesic rings, and they are almost parallel to each other. However, in the main body, there are not even enough geodesics. The reason behind this is the imperfection of the cut-point

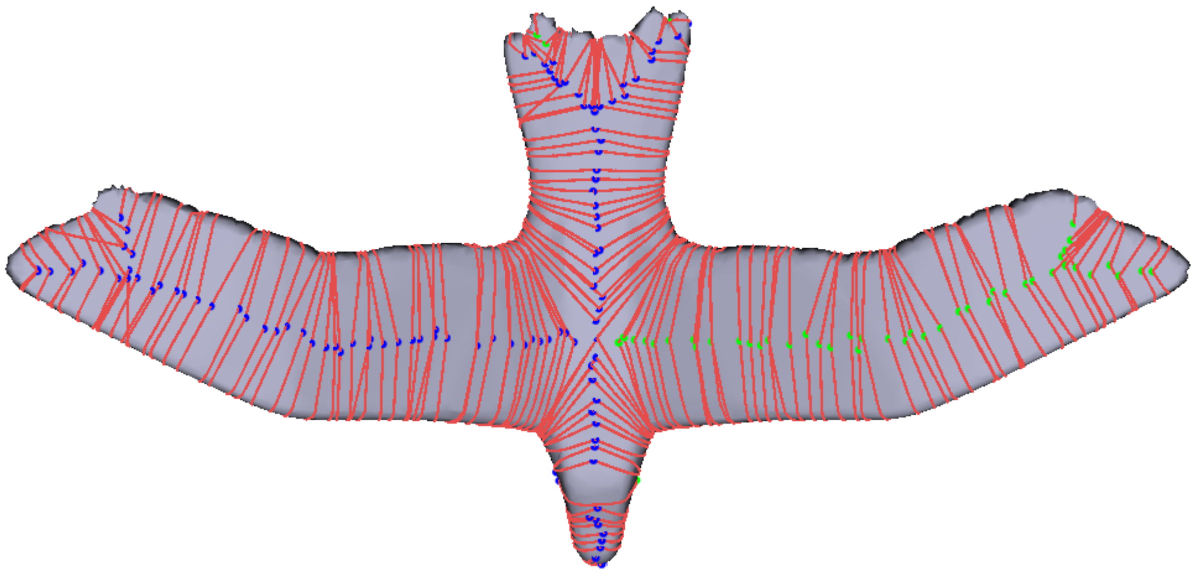
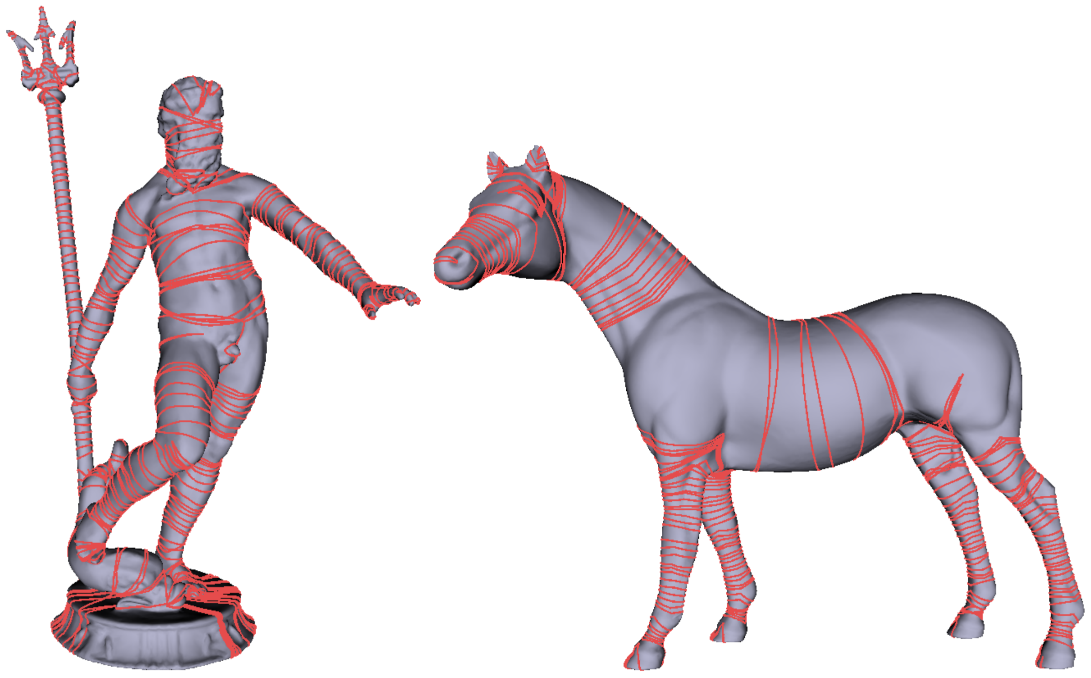


Figure 5.7: Cut-points and geodesics shown on the bird model.



(a) (b)

Figure 5.8: Geodesics of Neptune (a) and horse (b) models.

locations (Figure 5.9). Even if there are strict enforcements for angles and distances, the cut-points are scattered instead of following a line. Therefore, the results still differ from mesh to mesh, depending on the shape properties.

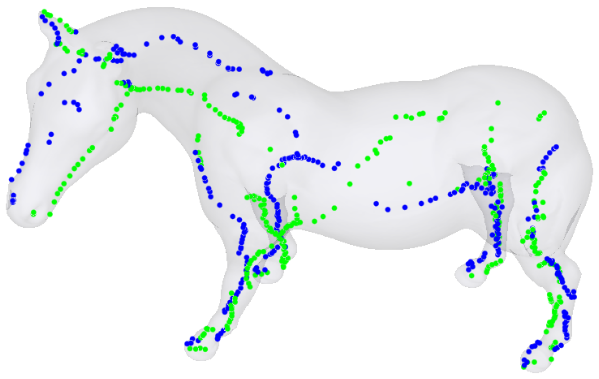


Figure 5.9: Cut-points of the horse model viewed from above. The points do not follow a straight cut-line.

For the future, first the issues remaining with the feature-point selection and shortest geodesics need to be solved. For better results, different treatment for the areas near special (tip and junction) nodes may be a good solution. A better choice of the feature points should also solve geodesic-related problems.

In the last remaining step, geodesics perpendicular to the initial ones have to be computed. The intersection points between these two sets of geodesics will be some distance away from the cut-points. The new geodesics should also be parallel to each other, hence the intersection points with the initial geodesics need to be selected with care. After the remaining issues are solved, and the last step is completed, the system may be used as a general-purpose parameterization framework.

Chapter 6

Conclusion

In this thesis, we have described a novel framework for extracting curve skeletons from 3D meshes. Within this framework, we have included three different ways to extract curve skeletons. Two of these methods are based on the work of Au et.al. [4], which use a Laplacian-based contraction and a series of post-processing steps. One of these employs surface skeletons as an intermediate medium, whereas the other one uses the same ideas from [4]. The third method is based on the work of Telea and Jalba [7], which makes use of a global importance measure of the surface skeleton.

For judging the quality of the curve skeletons, we have defined a set of criteria, rather than creating a formal definition of the curve skeleton. By using this set of criteria, we were able to analyze and compare our results to other state-of-the-art methods. Also, the global set of criteria can be modified or simplified based on the application that the curve skeletons will be used for.

Our framework can extract curve skeletons which are smoother and more detailed compared to the curve skeletons extracted by other state-of-the-art methods. In most cases, our skeletons are also better-centered. However, centeredness highly depends on the geometric properties of the surfaces. When an object has both bulky parts and small extremity details, centeredness may become inaccurate. Another limitation is that the smoothness and node sampling of the curve skeletons depend on geometric properties and also on mesh sampling. But these limitations can be solved by careful parameter tuning for the mesh contraction and the edge-collapse steps. In general, our framework is not sensitive to parameter tuning: a default set of parameters delivers high-quality skeletons for most meshes.

Lastly, we have created a solid foundation for an extension of our framework, which will make use of curve and surface skeletons for surface parameterization. This area has not been associated so far with skeletons, therefore we have proposed an idea, build initial parts of the framework and drawn a road map for future work.

Appendix A

Pseudo-Code Notations for the Algorithms

In some of the algorithms, we have used some expressions, which may not be clear to the readers who are not familiar with programming. In this appendix, we explain those notations. Below, we first give the notation, and afterwards explain its meaning.

1. **Statement:** $a.b[n]$

   Meaning: Here, a and b are variables and b belongs to a. b is also a container, and $b[n]$ refers to the $(n-1)$th element of b, considering the first element is at the index 0. Pseudo-code example of this statement can be seen in Algorithm 6 at line 8.

2. **Statement:** $a \leftarrow b \ ? \ c \ : \ d$

   Meaning: This is a short notation for a conditional statement where a is a variable and b is a boolean condition. The value of a will be c if b holds, and it will be d if b does not hold. The extended pseudo-code is given in Algorithm 12.

Algorithm 12 Conditional statement

```
if b = true then
    $a \leftarrow c$
else
    $a \leftarrow d$
end if
```

Bibliography

[1] Reniers, D., J. van Wijk, and A. Telea (2008) "Computing Multiscale Curve and Surface Skeletons of Genus 0 Shapes Using a Global Importance Measure," *IEEE Transactions on Visualization and Computer Graphics*, **14**(2), pp. 355–368.
URL `http://dx.doi.org/10.1109/TVCG.2008.23`

[2] Reniers, D. and A. Telea (2008) "Hierarchical part-type segmentation using voxel-based curve skeletons," *Vis. Comput.*, **24**(6), pp. 383–395.
URL `http://dx.doi.org/10.1007/s00371-008-0220-5`

[3] Li, X., T. W. Woon, T. S. Tan, and Z. Huang (2001) "Decomposing polygon meshes for interactive applications," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, ACM, New York, NY, USA, pp. 35–42.
URL `http://doi.acm.org/10.1145/364338.364343`

[4] Au, O. K.-C., C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee (2008) "Skeleton extraction by mesh contraction," in *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, ACM, New York, NY, USA, pp. 44:1–44:10.
URL `http://doi.acm.org/10.1145/1399504.1360643`

[5] Cao, J., A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su (2010) "Point Cloud Skeletons via Laplacian Based Contraction," in *Proceedings of the 2010 Shape Modeling International Conference*, SMI '10, IEEE Computer Society, Washington, DC, USA, pp. 187–197.
URL `http://dx.doi.org/10.1109/SMI.2010.25`

[6] Tagliasacchi, A., H. Zhang, and D. Cohen-Or (2009) "Curve skeleton extraction from incomplete point cloud," *ACM Trans. Graph.*, **28**(3), pp. 71:1–71:9.
URL `http://doi.acm.org/10.1145/1531326.1531377`

[7] Telea, A. C. and A. C. Jalba (2012) "Computing Curve Skeletons from Medial Surfaces of 3d Shapes," in *Theory and Practice of Computer Graphics (TPCG), Eurographics*.

[8] Dey, T. K. and J. Sun (2006) "Defining and computing curve-skeletons with medial geodesic function," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 143–152.
URL `http://dl.acm.org/citation.cfm?id=1281957.1281975`

[9] Sharf, A., T. Lewiner, A. Shamir, and L. Kobbelt (2007) "On-the-fly curve-skeleton computation for 3d shapes," in *Eurographics 2007 (Computer Graphics Forum)*, vol. 26, Eurographics, Prague, pp. 323–328.

[10] Ma, J., S. W. Bae, and S. Choi (2012) "3D medial axis point approximation using nearest neighbors and the normal field," *Vis. Comput.*, **28**(1), pp. 7–19.
URL `http://dx.doi.org/10.1007/s00371-011-0594-7`

[11] Garland, M. and P. S. Heckbert (1997) "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 209–216.
URL `http://dx.doi.org/10.1145/258734.258849`

[12] Jalba, A. C., J. Kustra, and A. C. Telea (2012) "Surface and Curve Skeletonization of Large Meshes on the GPU," in *IEEE TPAMI*.

[13] Design, P. (2012), "Unfold 3D Mesh Unfolding Software," [Online; accessed 3-October-2012].
URL `http://www.polygonal-design.fr`

[14] Floater, M. S. and K. Hormann (2005) "Surface Parameterization: a Tutorial and Survey," in *Advances in multiresolution for geometric modelling* (N. A. Dodgson, M. S. Floater, and M. A. Sabin, eds.), Springer Verlag, pp. 157–186.
URL `http://vcg.isti.cnr.it/Publications/2005/FH05`