Technical note

Kinematic skeleton extraction from 3D articulated models

Jaehwan Ma, Sunghee Choi*
Geometric Computing Lab., Department of Computer Science, KAIST, Daejeon, South Korea

HIGHLIGHTS

• A kinematic skeleton extraction method for articulated bodies is proposed.
• It is a hybrid approach combining topology-based and geometry-based methods.
• It does not require manually-chosen feature points or markers.
• It is independent of both postures and the number of branches of the model.
• The results can be directly applied to the character rigging.

GRAPHICAL ABSTRACT

ARTICLE INFO

Keywords:
Skeleton extraction
Character animation
Pose-independence
Morse theory

ABSTRACT

We propose a method to extract a kinematic skeleton from an articulated 3D model. Unlike the curve skeleton which concentrates on the abstract structure of a given model, we aim to obtain the skeleton with accurate joint positions, which can be directly used for character rigging and animation. Restricting our focus on the articulated body, we construct its skeleton in a bottom-up manner starting from a set of automatically obtained feature points. Our method is a hybrid approach combining the advantages of topology-based and geometry-based methods using the Morse theory and the shape descriptor. It does not require manually-chosen feature points or markers, and is independent of both postures and the number of branches of the given model. Experiments show that our method can efficiently extract kinematic skeletons of various kinds of articulated bodies, and also that the results can be directly applied to the character rigging.

1. Introduction

The 1D skeleton of an object provides an intuitive abstraction for both shape recognition and manipulation. A good skeleton should be simple enough to provide a glimpse of the overall structure while preserving a certain level of detail. Because of its importance and usefulness, there have been many previous studies designed to obtain a robust and efficient method for extracting a skeleton from a given object. Strictly speaking, however, the 1D skeleton can be classified into two different categories: the curve skeleton and the kinematic skeleton (or bone skeleton). The former provides an understanding of the abstract structure of the given shape. The latter is used as a tool to control the animation or the deformation of its corresponding model.

Many existing skeleton extraction methods [1–4] focus on the curve skeleton. In this case, most nodes are optional nodes which make the resulting skeleton smoother. For the kinematic skeleton, however, all nodes should be carefully chosen because they become the joints for animating characters. Thus, the direct conversion (downsampling, for example) from the curve skeleton is not a simple task in the sense that it may produce redundant joints [2]. Moreover, the joints can be classified into two different categories. Joints such as the shoulder and the hip introduce a branch on the skeleton. Ideas based on topological features such as the Reeb graph and the Morse complex are suitable for detecting such joints. On the other hand, joints including the elbow and the knee do not invoke a “split” of the skeleton because they are characterized...
solely by geometric features such as angle, curvature, and thickness. Therefore, it is desirable to handle them independently. We call the former topological joints and the latter geometric joints, respectively.

In this paper, we propose a fully automatic, pose-independent method to extract the kinematic skeleton of an articulated model. It is a hybrid approach, and addresses topological and geometric joints from different points of view. It does not require any type of manually-defined markers or feature points and is free from assumptions concerning the poses of the given model or the number of branches (including legs, arms, and tails). Experiments show that our algorithm is also efficient with respect to time. The process takes less than five seconds for the model with 33,827 vertices and 18 terminal branches.

2. Related work

There have been various approaches to extracting the one-dimensional skeleton of a given model. One set of approaches are geometry-based methods that exploit the geometric features of the given model. The 1D simplification of the medial axis has been a popular technique in the computational geometry community [5,13]. Thinning methods [6–8,2] construct the 1D skeleton by repeatedly removing the boundary of the model. Mesh segmentation [9] is also closely related to the problem of skeleton extraction. There have been various studies on mesh segmentation and skeleton extraction based on curvature and geodesic distances [10–12], dihedral angles [13], slippage [14], convex decomposition [15], and shape diameter function [16]. However, geometric attributes of the surface are quite sensitive to the deformation or noise in the model.

Instead of focusing on the local geometry, topology-based methods [17–20,4] define a set of real-valued functions on the input mesh to construct a data structure describing its topology using structures such as Reeb graphs [21–23], contour trees [24], and Morse complexes [25,26]. The quality of a topology-based method largely depends on the selection of a mapping function and the feature points, and the isocontours of the mapping function tend to be highly distorted when the given model has a complex shape.

3. Kinematic skeleton extraction

Our algorithm is composed of three main stages: feature point selection, topology tree construction, and geometric refinement. A set of feature points is first extracted in the feature point selection stage (Section 3.1). Each feature point represents a terminal branch such as a finger or toe. Considering terminal branches as leaf nodes, we construct a tree T which describes the topology of the input model M = (V,E,F) in a bottom-up manner by merging small branches into a larger branch repeatedly during the topology tree construction stage (Section 3.2). Finally, T is fit into M as its kinematic skeleton after geometric refinements including internal joint extraction and joint position adjustment (Section 3.3).

3.1. Feature point selection

Preprocessing. We use the medial axis (MA) point approximation algorithm proposed in [27]. Starting from a sufficiently large ball tangent to M at each v ∈ V, the process iteratively shrinks the ball until it becomes maximal and empty, which results in an approximate medial ball tangent to M at v. We obtain a set M of approximate MA points of a given model M after preprocessing.

Candidate feature point selection. Let Dp(x), where p, x ∈ V, is the shortest distance from the source p to the target x on M. If there are branches such as fingers and toes, the vertices on the tips of the branches are likely highly to be the local maxima of Dp. Based on this observation, we use the local maxima of Dp as conservative initial guesses of feature points. We use the alternative notation Dp(x) instead of Dp(x) when describing the mesh distance between p and x without specifying the source point. When p is close to the tip of the branch, however, Dp may miss the feature point on that branch. To avoid this situation, we first find a feature point s which is the farthest vertex from a randomly chosen source point p, then perform Dijkstra’s algorithm again using s as a new source to compute Dp. The resulting set FP0 of candidate feature points is the union of the local maxima of Dp and s itself. Thus:

\[ FP^0 = \{x|D_p(x) \geq D_p(y), x \in V, \forall y \in N_s \cup \{s\} \} \]  
where Ns denotes the 1-ring neighbor of x. We keep track of δmax = max{Dp} for further processing.

Filtering. After we compute Dp(x) for x ∈ V by Dijkstra’s algorithm, Dp(x) for any point x on the edge or on the face are approximated by the linear interpolation of Dp of neighboring vertices. Let Cp(δ) be the isocontour defined by \(|x|D_p(x) = \delta, x \in M|\). The candidate feature point set FP0 may contain meaningless points that include noise. To distinguish these points from meaningful points, we estimate the depth of each branch and eliminate the points in FP0 whose corresponding branch is shallower than a certain threshold ̄δ (see Fig. 1 for the detailed condition). We repeat this filtering process for a sequence of appropriately chosen source points until there are no more points to be removed. The detailed procedure is described in Fig. 2. We used 3%–5% of δmax as ̄δ throughout our experiments.

When there are multiple feature points on a single branch, we select the one with the smallest corresponding medial ball because a point on a smaller feature tends to be a better representation of a branch than a point on a larger feature. The resulting set FP is shown in Fig. 3.

\[
\text{procedure FP FILTERING}(FP^0) \\
\text{1. } \tau \leftarrow -1, \delta^0 \leftarrow s \\
\text{2. repeat} \\
\text{3. } \tau \leftarrow \tau + 1 \\
\text{4. Compute } D_{p,\tau} \text{ using the Dijkstra’s algorithm} \\
\text{5. for each } p \text{ in } FP^0 \\
\text{6. } V_p \leftarrow \{v_1, v_2|e = (v_1, v_2) \in E, D_p(v_1) \leq \delta, D_p(v_2) \leq \bar{\delta} \} \\
\text{7. if } D_{p,\tau}(p) > \max_{v \in V_p}(D_{p,\tau}(v)) \\
\text{8. } FP^{\tau+1} \leftarrow FP^{\tau+1} \cup \{p\} \\
\text{9. } s^{\tau+1} \leftarrow \arg \max_{p \in FP^\tau}(D_{p,\tau}(p)) \\
\text{10. until } FP^\tau = FP^{\tau+1} \\
\text{11. return } FP^\tau
\]
The computation of our shape descriptor. Note that we present with its center $c$ the local topological splits on isocontours if $t$ compute to the web version of this article. 

For every feature point $f_i \in FP$ obtained in Section 3.1, we compute $D_{ij}$ using Dijkstra’s algorithm and sample $\bar{v}$ contours uniformly with respect to $D_{ij}$. Let $C_{ij}$ be the $j$-th isocontour centered at $f_i$ (using $C_i$ and $D_i$ as abbreviations for $C_i$ and $D_i$). We can detect topological splits on isocontours if $|C_{ij}| < |C_{i,j+1}|$, where $|C_{ij}|$ denotes the number of connected components of $C_{ij}$. Unlike the existing methods [19, 28] that select only one source point, we construct a topology tree $T_i$ for each $f_i \in FP$ locally. If $f_i$ is the lowest index satisfying $|C_{i,j}| \geq 2$, then we search for the boundary of the $i$-th leaf branch by investigating $C_{ij}$ for $j \leq f$. After pruning the branches using these boundaries (called cutting contours), we merge a set of smaller branches into a larger one according to their cutting contours are close to one another. Then, we again try to find the boundaries of newly introduced larger branches. We repeat this cut-and-merge operation until the local topology trees $T_i$ are merged into a global topology tree $T$.

Cutting. In general, the shape of the isocontour drastically changes near the topological joint. Assuming that the ideal shape of the contour as a planar circle, we encode the shape of $C_{ij}$ using a three-dimensional vector $SD = (sd(1), sd(2), sd(3))$, where the components denote thickness, flatness and distortion, respectively (see Fig. 4). The thickness $sd_{ij}(1)$ of $C_{ij}$ can be defined by the radius of the largest medial ball in the band-shaped region $\overline{C_{ij}}$, bounded by $C_{ij-1}$ and $C_{ij}$. For $sd_{ij}(2)$ and $sd_{ij}(3)$, we uniformly sample $\bar{v}$ points $x_{ij,l}$, where $1 \leq l \leq \nu$ on $C_{ij}$, and perform principal component analysis (PCA) on the points to obtain the best-fitting plane $P(C_{ij})$ with its center $c(C_{ij})$. Let $\hat{x}_{ij,l}$ be the projection of $x_{ij,l}$ onto $P(C_{ij})$. Then, $sd_{ij}(2)$ and $sd_{ij}(3)$ are defined by the out-of-plane and the in-plane deviation from the planar circle on $P(C_{ij})$, respectively.

$$sd_{ij}(1) = \max \{ \rho_p | p \in \overline{C_{ij}} \}$$

$$sd_{ij}(2) = \sum_{l} \frac{d(\hat{x}_{ij,l}, x_{ij,l})^2}{\nu \ast \text{rad}_{ij}}$$

$$sd_{ij}(3) = \sum_{l} \frac{(d(\hat{x}_{ij,l}, c(C_{ij})) - \text{rad}_{ij})^2}{\nu \ast \text{rad}_{ij}}$$

where $\text{rad}_{ij} = \sum_{l} \frac{d(\hat{x}_{ij,l}, c(C_{ij}))}{\nu}$ denotes the expected radius of $C_{ij}$. Using these shape descriptors for each $C_{ij}$ we finally choose the $C_{ij}$ that maximizes the following score function (the amount of change) as a cutting contour:

$$\text{score}(C_{ij}) = \sum_{i \in \{1, 2, 3\}} \omega(i)(sd_{ij}(l) - \overline{sd}_{ij-1}(l))$$

$$+ \frac{\omega(l)}{2} (2\overline{sd}_{ij+1}(l) - 2\overline{sd}_{ij}(l) + \overline{sd}_{ij-1}(l))$$

where $\omega(l)$ denotes the weight of $sd(l)$. The two terms in Eq. (5) approximate the first and the second derivative of $SD$, respectively.

Cut-and-merge iteration. For each branch with local topology trees $T_i$, we scan the isocontours while checking $|C_{ij}|$, and find its cutting contour in the cutting stage. Then, some branches are merged into a larger branch by checking the condition in Eq. (6) in the merging stage. We perform this cut-and-merge iteration repeatedly until the global topology tree $T$ is obtained.

For each candidate branch in the current iteration round, there are two possible cases while scanning the isocontours on a branch. One is that the topological split detected successfully in the cutting stage. In this case, the branch should participate as a candidate of the next iteration round because the split implies that this branch is not completely scanned yet. Another possible case is that it penetrates the region covered by different branches before detecting the topological split. When this occurs, we simply stop the scan. The overall process of topology tree construction is described in Fig. 5.

We assign high priorities to thinner branches (i.e., having smaller $sd(1)$) when determining the scanning order among them. As a consequence, the tail branch does not perform the scanning process until all arms and legs are finished (Fig. 5(c) and (d)) because it is attached to the thicker part.
Fig. 5. (a) Feature points of Armadillo (b)-(e) show examples of our topology tree construction process. (b), (c), and (d) represent the results of the first cutting stage, the first merging stage, and second cutting stage, respectively. Note that the node on the torso in (e) is a virtual node whose exact position is not identified yet.

Fig. 6. Pseudocode of the internal joint extraction process.

3.3. Geometric refinement

**Internal joint extraction.** The bone of the \( i \)-th branch \( M_i \) obtained in the previous stage is a set of straight lines connecting two joints \( J_i^1 \) and \( J_i^2 \) of each branch. Our strategy for finding \( J_i^1 \), \( r \geq 3 \), is to detect the MA point with the maximum deviation from the current bone. To overcome noise on the MA, we take a subset of \( M \) whose elements are sufficiently close to the skeleton. For every slice \( \mathbf{E}_{i,j} \subset M_i \), we take the medial balls whose radii are larger than \( \omega \cdot \text{sd}_{i,j}(1) \) as reliable samples (setting \( \omega = 0.8 \) throughout our experiments). \( M_i \) denotes the set of reliable samples on the MA contained in \( M_i \).

Let \( \text{ED}(m, J_i^1) \) be the Euclidean distance from \( m \in M_i \) to the bone segments induced by the joints in the set \( J_i^1 \). Our algorithm incrementally adds the joints \( J_i^1 \), \( r \geq 3 \) into \( J_i^1 \) one by one, where \( J_i^1 \) denotes the set of joints in \( M_i \). If the magnitude of the change of the current bone is negligible after inserting \( J_i^1 \), then we throw out \( J_i^1 \) and halt the insertion process. The detailed procedure of our internal joint extraction method is described in Fig. 6. We set \( \delta = 0.1 \) in our experiments.

Note that our strategy is not an optimization but a handling of local deviation. We choose a point with the maximum deviation (the normalized Euclidean distance with respect to the thickness) as the candidate internal joint (see Line 4 in Fig. 6). Our focus is the extent to which the shape of bones is affected after inserting a joint rather than minimizing \( \text{Er}(J_i^1) \) for every insertion. Thus, \( \text{Er}(J_i^1) \) may not decrease monotonically during the joint insertion process (see Fig. 7).

**Skeleton completion.** At this point, we have obtained a set of internal joints and associated bones for every branch. Because we now have the actual directions of bones inside the branches, we can adjust the cutting contours of each branch to make the boundaries clearer. For each cutting contour with the plane \( P = (c, n) \), we set its new normal \( n \) to be parallel to the direction of the bone adjacent to \( c \) and then move \( c \) along with the direction of \( n \) to adjust its position. This process is similar to that performed in the topology tree extraction stage.

Finally, the root region can be obtained by cutting out every subbranch using its cutting contour. For each \( v_i \in M_{\text{root}} \), we approximate the root joint \( J_{\text{root}} = \sum \rho_i \) by the weighted average of the vertex positions, where \( \rho_i \) is the radius of the medial ball tangent to \( M \) at \( v_i \).

4. Experimental results

Figs. 8 and 9 show the extracted kinematic skeletons and the deformation results after manual rigging for various articulated models, respectively. The feature points and the topological joints are captured quite well. The resulting skeletons convey the abstract structures of given models, indicating that they can be directly used by animators for further processing.

On the other hand, several internal joints are somewhat different from our expectation. Some of them have erroneous positions (e.g., the arms of Armadillo), and others are missing (e.g., the legs of Horse). In fact, our internal joint extraction idea detects the most critical bending within a branch incrementally. The amount of bending depends not only on the angle between two branches around the internal joint but also on the feature sizes of those branches. The bending on the thinner branches is more noticeable than that on the thicker ones if the bending angles are the same. Our algorithm detects internal joints successfully when the bending is quite noticeable, as in the left back leg of Horse in Fig. 8, but the knee joints of the other legs are missed due to the absence of noticeable bending. However, it is extremely difficult to detect the internal joints for such cases in either a geometric or a topological manner without a predefined number of joints or the approximate length of each bone.

The model specifications and the running time of each step are shown in Table 1. Regarding the MA computation time as a preprocessing,
the most time-consuming process is the topology tree construction. In our case, not only the model complexity, indicated by $|V|$ and $|F|$, but also the topological complexity of $\mathcal{M}$ are dominant factors to determine the overall running time. The running time of all substeps, including the computation of distance functions, shape descriptors, and cutting contours, is proportional to the number of branches in $T$.

5. Conclusion

Summary. We proposed a method to extract the kinematic skeleton from an articulated 3D model. Our method is a bottom-up tree construction method which builds an entire skeleton by investigating topological changes on the wavefront of $D_p(x)$. After that, we refine the resulting skeleton with additional joints based on geometric conditions such as (bending) angle and thickness. The medial axis points are used throughout the process for determining joint position and estimating the thicknesses of branches. Our method is a fully automatic process that does not require manually chosen feature points or markers and is independent of both postures and the number of branches of a given model. Experiments show that our method can efficiently extract kinematic skeletons from various types of articulated bodies and that the results can be directly applied to the character rigging.

Limitations and future work. A limitation of our work is that input models should be genus-0 without tunnels. Another limitation is that our method produces star-like skeletons. Though it is suitable for invertebrates such as insects and spiders, the spinal column should be provided for humanoids and four-legged animals to deform the torso part. Finally, a method to symmetrize resulting skeletons to convey the bisymmetric nature of most articulated bodies would be helpful.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0024092), and by the KUSTAR-KAIST Institute, Korea, under the R&D program supervised by the KAIST.

References


