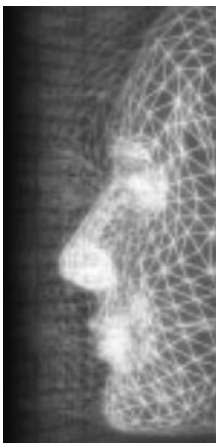
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2009; **20**: 121–131
Published online 22 May 2009 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.284

Automatic rigging for animation characters with 3D silhouette



*By JunJun Pan, Xiaosong Yang, Xin Xie, Philip Willis and Jian J Zhang**

Animating an articulated 3D character requires the specification of its interior skeleton structure which defines how the skin surface is deformed during animation. Currently this task is to a large extent accomplished manually, which consumes a large amount of animators' time. This paper presents an automatic rigging method making use of a new geometry entity called the 3D silhouette. The first step is to extract a coarse 3D curve skeleton and some skeletal joints of a character. This curve skeleton is then refined with a perpendicular silhouette. According to the connectivity of the skeletal joints, the hierarchical animation skeleton is finally constructed. By avoiding complicated computation such as voxelization and pruning, this method is simple and efficient, much faster than existing methods. It proves very useful for quick animation production, with applications including games design and prototype graphical systems. Copyright © 2009 John Wiley & Sons, Ltd.

Received: 23 March 2009; Accepted: 24 March 2009

KEY WORDS: rigging; curve skeleton; animation skeleton; medial axis; 3D silhouette

Introduction

To animate a 3D character, be it a virtual human or a creature, the animator usually embeds a skeleton into the character model and binds it to the character. This process is called rigging. Once a character is properly rigged and the influence of the skeletal joints is associated with the skin surface (known as the skinning), animating the skeleton will animate the character. Such type of animation is called skeleton-driven animation and it has become a *de facto* industrial standard for character animation. With the current industrial animation pipeline, using the state-of-the-art commercial software such as Maya, rigging is done manually. This process includes placing each skeleton link (bone) and joint inside the character, which represents one of the most tedious tasks for character animation. Though several methods have been reported for skeletonization, the computation is complex and expensive, and the result is not good enough to be used directly. Therefore, until now, rigging remains a skilful and time-consuming job.

In this paper, we present a technique that can automatically rig a 3D character by extracting an appropriate skeleton structure including bones and joints inside a character body. This skeleton is called the *animation skeleton* (which is different from the *curve skeleton* as explained later). The objective is to develop a computationally efficient and easy-to-use method to generate a desirable animation skeleton suitable for the subsequent skinning operation. Our method aims to satisfy the following four criteria:

1. Centrality: Quality animation requires that all the joints and bones of the animation skeleton lie on the central path of the body segments, such as the arms and legs of a human model or the tentacles of an octopus. Although it contradicts the real anatomy of a human body, this condition prevents biased deformation from arising during animation.
2. Generality: A good skeletonization method should be applicable to a wide variety of character models, such as human-like characters, hand and foot models, four-legged animals, molluscs and so on.
3. High-efficiency: Skeletonization is often a computation intensive operation, as potentially it may involve all the mesh vertices of a character model. Efficient computation makes it suitable for quick animation production.

*Correspondence to: J. J. Zhang, National Center for Computer Animation, Media School, Bournemouth University, Poole, UK.
E-mail: jzhang@bournemouth.ac.uk

4. Ease-of-use: An easy to use rigging technique is not only important for professional animators, but also can help the novices to learn animation more quickly.

Before diving into the technical details, we need define some terminology first, which will be used throughout this paper.Animation skeleton: the skeleton used in character animation, which is made up of bones (skeleton segments) and joints.Curve skeleton: the medial axis of an object, which is a term often used in CAD (computer aided design), graphic data compression and object topology analysis.Skeletonization: the process of animation/curve skeleton extraction.

Related Work

Curve skeleton extraction is useful for many areas of CAD and topology analysis.[1] Although such a skeleton is different from the animation skeleton used in animating 3D characters, research on this topic lays some important ground work. Existing techniques of curve skeleton extraction can be classified into two main categories: geometric and volumetric, according to whether an algorithm is surface or interior based.

Geometric methods are applicable to polygonal meshes or scattered points. One popular method in this category is to use the Voronoi diagram to generate a skeleton from the 3D polygonal vertices.[2] The Voronoi diagram describes the subdivision of a space into regions that are closest to a generator element, of which the interior edges and faces can both be used. The disadvantage of this method is that the outcome is a medial surface, not a medial axis. Algorithms for further pruning and curve thinning are needed. Another method gained much attention in recent years is the Reeb graph. The Reeb graph is a 1D structure whose nodes are critical points of a defined function on the model surface, which encodes the topology of the model.[3,4] However, most Reeb-graph-based methods require the user to set the boundary conditions explicitly. Au *et al*.[5] presented a mesh contraction method to extract the skeleton using implicit Laplacian smoothing. The contracted mesh is then converted into a 1D structure curve through a connectivity surgery process. This approach is robust and noise insensitive. Nevertheless, it has some obvious flaws besides the limitations mentioned by authors. To ensure the collapsed shape maintaining the original geometry in the contraction process, the weights of the attraction and contraction constraints for different vertices must be carefully set, which in practice is a difficult task for the animator.

The skeletonization algorithm presented in our paper belongs to the Voronoi-based geometric methods since it generates a curve skeleton through constrained Delaunay triangles. However, compared with other Voronoi-based geometric methods, one remarkable advantage of our algorithm is that the skeleton is extracted from two perpendicular 3D silhouettes, not the medial surface. It can generate a curve skeleton directly without post-processing, such as thinning and pruning, which significantly speeds up the computation.

Volumetric methods can be subdivided into three groups: thinning, distance field based methods and general field functions. Thinning is a process of generating the curve skeleton by iteratively removing pixels or voxels from the object boundary until some required conditions are satisfied. Recently, Wang and Lee[6] applied a volumetric models shrinking algorithm before thinning, which produced smoother skeletons but pruning is still required. Most thinning algorithms are data dependant. They are usually designed for models with a specific connectivity.[1]

The field of the shortest distances from the interior points to the boundary is defined as the distance field. Most distance field based methods choose Euclidean distances as their distance functions.[7,8] They attempt to search the ridges (the set of voxels that are locally centred in the 3D objects) of the distance field. After pruning and connecting these candidate voxels, the topology is confirmed and the curve skeleton is extracted. These methods, however, are not always robust and some additional processing steps are sometimes needed to translate surfaces to curves.

General field function based approaches usually choose a generalized potential field function in which the potential of an interior point in 3D object is computed as the sum of the potentials generated by the object's boundary. Grigorishi and Yang[9] presented an electrostatic field function to generate a potential inside a 3D object to extract the curve skeleton. Liu *et al*.[10] introduced the Newtonian repulsive force to solve the searching problems. Compared with the distance field, these methods can generate nicer curves from medial surfaces since they take into account larger boundary areas. However, this merit is achieved at the price of increasing computation complexity.

Skeleton embedding differs from the above methods, which extract a curve skeleton. Its idea is to place an existing template animation skeleton into a given model. To obtain an optimal skeleton fitting for the given models, Baran and Popvic[11] designed a maximum-margin supervised learning method with a set of hand-constructed penalty functions. They also presented a Laplace's diffusion equation based method for skin

DOI: 10.1002/cav

attachment. Their method can rig a character under 1 minute on a midrange PC. However, this approach requires the character to have approximately the same proportion and pose as the template skeleton. Due to the hampered flexibility, it had only been applied to human-like characters. Aujay *et al*.[12] presented a method to construct the Reeb graph of a harmonic function, which gives the overall morphological structure of the model. Then it is refined and embedded to generate the animation skeleton using the anatomical information. This approach is quite fast, but it requires parameter tuning and needs a reference anatomical template during the refining process.

In summary, although there are many algorithms presented by the research community, they are primarily concerned with a general (not specific) problem of curve skeleton generation. As a result, these methods are typically numerically expensive or need manual intervention from the user. A few methods were aiming at computer animation, but they have their own special drawbacks. In this paper, we develop a new technique, whose purpose is to rig a 3D animated character automatically and properly.

Overview

Unlike most automatic skeletonization methods described in the previous section, our aim is not to solve a 'general' case. In animation practice, rigging is always performed at the neutral rest pose. The skin mesh of this pose is roughly expanded (i.e. no occlusion) which lays the basic pre-condition for our technique. It allows us to develop an efficient and effective purpose-built rigging approach for animating various characters.

The operations employed by existing skeletonization methods, such as voxelization, thinning and pruning, are all computationally costly. They are not an ideal candidate for character animation. Our method takes advantage of the fact that common 3D animation characters such as humans and animals are usually composed of a series of segments whose cross-sections are approximately elliptic. Therefore, the projection of their 3D curve skeleton on a 2D plane can be approximated with the 2D medial axis of the projection of the original model in the same projection orientation. This suggests that the 3D problem, which is to find a curve skeleton, can be dealt with in 2D spaces leading to a much cheaper and quicker solution. We can reconstruct a 3D curve skeleton from multiple 2D medial axes as illustrated in Figure 1a. Here the yellow and red curves represent the 3D silhouettes of the thumb in two perpendicular projection orientations and the blue curve represents the 3D medial axis.

Our automatic rigging method consists of three steps. Firstly we introduce a new geometric entity, called the 3D silhouette, extending from the ordinary silhouette. Based on the 3D silhouettes detected from two perpendicular projections of a character model, we extract a curve skeleton, which depicts the rough shape of the final animation skeleton, using constrained Delaunay triangulation. Then we identify the necessary joints to form the animation skeleton by studying the triangulated primary silhouette and down sampling the curve skeleton. Finally, the skin is attached to the skeleton automatically ready for skinning.

Our method has the following advantages:

1. The method works on the original mesh directly and handles the 3D skeletonization problem in two 2D spaces, making it highly efficient.
2. The skeleton extraction process in our method does not change the mesh connectivity. The final curve skeleton is clean, centred and preserves the topology of the original model.
3. Without any parameter setting and complex control during rigging, this method is easy-to-use in practice.

Primary 3D Silhouette Detection

As we know, a silhouette is a 2D geometric entity. Here we extend it to a new entity, called the 3D silhouette. Given a polygonal mesh, a 3D silhouette is the 2D silhouette with its depth values recorded.

Let V be a vertex set with connectivity, which represents the mesh model. Without losing generality, suppose the appropriate projection orientation with the least occlusion of this mesh model is along the Z axis and the 2D projection of V on the XY plane is P. For P, we can detect its 2D silhouette C'. For each element $c'_i(x, y)$ in set C', we record its Z coordinate Z_{ci} from the corresponding vertex $c_i(x, y, z)$ in V. Here $c_i(x, y, z)$forms a vertex of the 3D silhouette of V when the projection direction is along the Z axis. So we can define a 3D silhouette of mesh model V as follows:

$$C\{c_i(x,y,z)|c_i(x,y,z) \in V, c'_i(x,y) \in C', C' \subset P, x_{c_i} = x_{c'_i}, y_{c_i} = y_{c'_i}\}$$

To facilitate the generation of an ideal curve skeleton, it is important the projection direction of the original 3D model should be selected that it minimizes the like-

DOI: 10.1002/cav

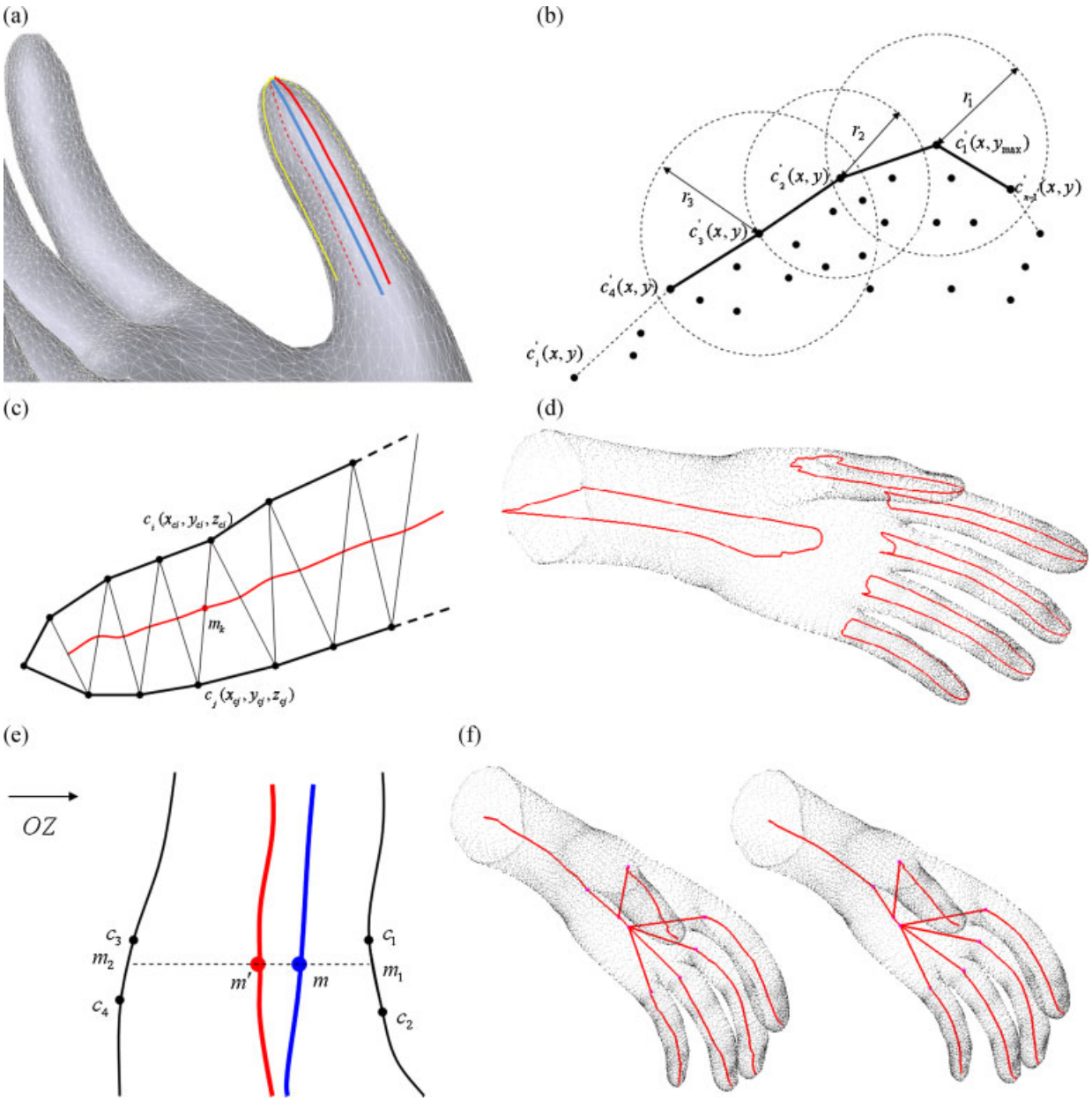


Figure 1. Illustrations in automatic rigging with our method. (a) 3D silhouettes and medial axis in the hand example, (b) illustration of Level 1 search, (c) illustration of extracting the 3D medial axis from a 3D silhouette, (d) the second 3D silhouettes for six branches of the hand model, (e) illustration of the curve skeleton refinement, (f) comparison between curve skeletons of hand model before and after refinement. Left: skeleton before refinement, right: skeleton after refinement.

lihood of occlusion. This is equivalent to making the model have the maximal projection area approximately.

The primary 3D silhouette is defined as the 3D silhouette when the original 3D model is projected from this optimal orientation. We designed a two-level-search algorithm to detect the primary 3D silhouette. Level 1 is for global search, which is to find the 3D silhouette vertices of this mesh model with no regard to the connectivity. Level 2 is for local search, which is to connect all the 3D silhouette vertices considering their

DOI: 10.1002/cav

connectivity in the mesh model. Figure 1b illustrates the level 1 search process.

Level I: Global Search

Step 1: Find the highest vertex $c_1'(x, y_{\max})$in the projection set P, as the start vertex.

Step 2: Search all the vertices whose distance to $c_1'(x, y_{\max})$ is within r in P. r is an experimental parameter which can be evaluated by the maximal distance between current vertex $c_1'(x, y_{\max})$ and its neighbour vertices. All the distances in the global search are 2D Euclidean distances.

Step 3: Suppose $\overrightarrow{OX}$is the positive orientation of the polar coordinates and $c_1'(x, y_{\max})$ is the origin. If the range of polar angle is set as $(-\pi, \pi)$, search the vertices with the least polar angle in the candidates of Step 2. This vertex $c_2'(x, y)$is the second vertex.

Step 4: Search all the vertices whose distance to the current vertex $c_2'(x, y)$ is within r in P. Find the vertex $c_3'(x, y)$with the maximal angle $\angle c_3' c_2' c_1'$in these vertices. $c_3'(x, y)$ is the third vertex of 2D silhouette.

Step 5: Repeat Step 4 until the current detected vertex $c_i'(x, y)$meets the start vertex $c_1'(x, y_{\max})$.

Level 1 search is completed when the 2D silhouette is closed. This process is similar to Jarvis' March for convex hull computation.[13] The difference is that the new vertex to be searched is restricted in the neighbourhood of the current vertex in our algorithm. The accuracy and robustness of searching are influenced by the searching radius r. Here we evaluate r with an experimental self-adaptive formula (1). $q_i(x, y, z)$is the neighbour vertex connected with the current vertex $q(x, y, z)$.

$$r = \max ||q_i(x, y, z) - q(x, y, z)|| \quad (1)$$

When 2D silhouette $C'$is detected, we can acquire the Z coordinates (depth information) for all the vertices according to the definition above. After adding the Z coordinate to each vertex in 2D silhouette $C'(c_i'(x, y) \in C')$, a 3D vertex set $C''\{c_i''(x, y, z) \in C'', x_{c''} = x_{c'}, y_{c''} = y_{c'}\}$ is constructed. The next step is the local search, which connects these 3D vertices C''.

Level 2: Local Search

Connect each vertex $c_i''(x, y, z)$to the next vertex $c_{i+1}''(x, y, z)$ in C''according to the connectivity of original mesh model. This process can be regarded as searching the approximate shortest route for each vertex pair on the mesh, which starts at $c_i''(x, y, z)$ and ends at $c_{i+1}''(x, y, z)$. Suppose there are n vertices in set C'', it can be described as the following loop operation:

For (i from 1 to n) Connect $(c_i''(x, y, z), c_{i+1}''(x, y, z))$

The operation *Connect* $(c_i''(x, y, z), c_{i+1}''(x, y, z))$ can be described as follows:

Step 1: Treat the first vertex $c_i''(x, y, z)$as the start point $c_{i1}(x, y, z)$, find all the neighbour vertices connected with $c_{i1}(x, y, z)$ in V.

Step 2: Find a vertex $c_{i2}(x, y, z)$, which has the maximal angle $\angle c_{i2} c_i'' c_{i-1j}$ in these vertices (c_{i-1j} is the previous vertex of $c_i''(x, y, z)$when $c_{i-1}''(x, y, z)$and $c_i''(x, y, z)$are connected). $c_{i2}(x, y, z)$ is the second vertex of this 3D silhouette section.

Step 3: Search all the vertices connected with current vertex $c_{i2}(x, y, z)$. Find a vertex $c_{i3}(x, y, z)$with the maximal angle $\angle c_{i3} c_{i2} c_{i1}$in these vertices. $c_{i3}(x, y, z)$ becomes the third vertex of this local 3D silhouette.

Step 4: Repeat Step 3 until the currently detected vertex $c_{ij}(x, y, z)$meets the end vertex $c_{i+1}''(x, y, z)$ or the distance between $c_{ij}(x, y, z)$ and $c_{i+1}''(x, y, z)$is less than the experimental threshold value r. (r is defined in formula (1)).

Although the level 2 method is effective for local search, without the global estimate of the silhouette from level 1, level 2 usually falls into an endless loop. An example of the detected 3D silhouette of the hand model is shown in Figure 2b.

Curve Skeleton Extraction

For a discrete 2D silhouette, its 2D medial axis can be detected easily by several Voronoi-based geometric methods mentioned in the Related Work. In this section, we extend the constrained Delaunay triangulation-based method[14] to detect 3D medial axes. This process can be simply illustrated in Figure 1c. We perform 2D constrained Delaunay triangulation for the 3D silhouette regardless of its Z coordinates. Then the 3D medial axis is generated from the 2D medial axis coupled with the depth information which is interpolated from the 3D silhouette.

In Figure 1c, assume the vertices of a triangle edge are $c_i(x_i, y_i, z_i)$ and $c_j(x_j, y_j, z_j)$, the midpoint $m_k\left(\frac{x_i+x_j}{2}, \frac{y_i+y_j}{2}, \frac{z_i+z_j}{2}\right)$ of the edge $c_i c_j$forms a vertex of the 3D medial axis for the 3D silhouette. When the 2D medial axis is detected, the 3D medial axis $M\{m_k(x, y, z) \in M\}$is also constructed. After the constrained Delaunay triangulation, we can divide all the triangles into three categories: triangles with two external edges (*terminal triangles*), triangles with one

DOI: 10.1002/cav

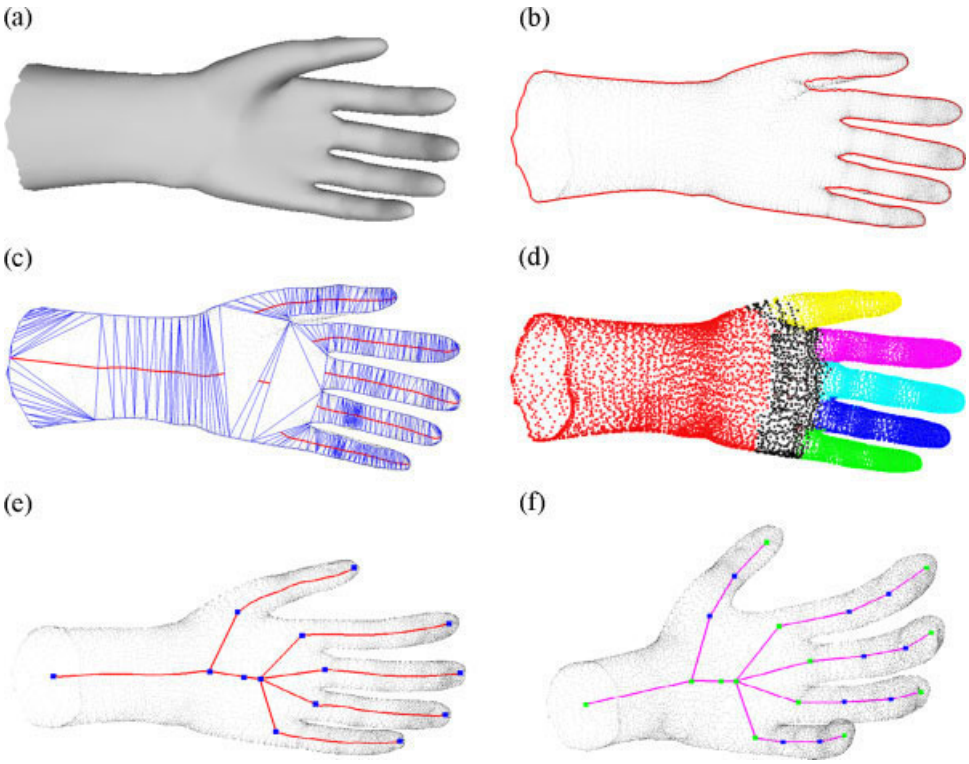


Figure 2. Generating animation skeleton of a hand model. (a) Original mode, (b) primary 3D silhouette, (c) 3D medial axis of hand through constrained Delaunay triangulation, (d) decomposition result, (e) coarse curve skeleton and key skeleton points, (f) animation skeleton and skeletal joints.

external edge (*sleeve triangles*) and triangles without external edges (*junction triangles*). A medial axis is obtained by connecting the midpoint of the internal edges.[14] The experimental result of constrained Delaunay triangulation and 3D medial axis detection of a hand model is shown in Figure 2c.

Decomposition

From our observation stated earlier, an accurate curve skeleton can be determined by two perpendicular 3D silhouettes (Figure 1a). The coarse curve skeleton is centred in the first projection plane, but some part may not be centred in other projection planes, since the z coordinates of the curve skeleton points are not considered during coarse curve skeleton generation. Therefore when a coarse curve skeleton is extracted from the primary 3D silhouette, we need to decompose the object and detect the second 3D silhouette in another perpendicular projection plane to adjust the z coordinates of the skeleton for each branch.

In the decomposition step, we use a simple but effective method which is similar to the nearest-neighbour algorithm in pattern recognition to classify all the vertices of the object. It can be described as:

$$s(v_i,k) = \min_{j=(1,n)}^{k} |\text{distance}(v_i, m_{k,j}(x,y,z)) - l_{k,j}| \quad (2)$$

where distance$(v_i, m_{k,j}(x,y,z))$is the Euclidean distance between vertex v_i and point $m_{k,j}(x,y,z)$ on the kth branch of the coarse curve skeleton. k stands for the sequence number of this branch; j is the sequence number of point $m_{k,j}(x,y,z)$ on this kth branch; $l_{k,j}$is the distance between central point $m_{k,j}(x,y,z)$ and the primary silhouette of this branch. So $s(v_i,k)$ is the minimum value of the difference between distance$(v_i, m_{k,j}(x,y,z))$ and $l_{k,j}$. When $s(v_i,k)$is determined, vertex v_i will be classified into the kth branch of the model. Figure 2d shows the decomposition result with seven parts of the hand.

DOI: 10.1002/cav

Curve Skeleton Refinement

When an object is decomposed, we need to detect the second 3D silhouette for each branch to fine tune the z coordinates of the curve skeleton on the second projection plane, which is perpendicular to the first projection plane. The projection orientation can be calculated with formula (3) as it makes the projection of this curve skeleton branch approximately fully extended on its corresponding second projection plane. In Equation (3), $\overrightarrow{v_k}$is the projection vector calculated for the kth curve skeleton branch. $\overrightarrow{p}$is the projection vector during the generation of primary 3D silhouette. $m_{k\text{end}}$ and $m_{k\text{start}}$ stand for the end point and start point of the kth curve skeleton branch.

$$\overrightarrow{v_k} = \overrightarrow{p} \times \overrightarrow{m_{k\text{end}} - m_{k\text{start}}} \tag{3}$$

Here, we use the global search method described in primary 3D silhouette detection to extract the second 3D silhouette for each decomposed part of the model. Figure 1d gives the result.

When the second 3D silhouette is detected for each curve skeleton branch, the next step is to fine tune the z coordinates of the curve skeleton. Figure 1e illustrates how to refine a coarse curve skeleton through the second 3D silhouette. For a given point m in the coarse curve skeleton (blue line), we can find four nearest neighbour vertices c_1, c_2, c_3, c_4 in the second 3D silhouette of this branch. As we only adjust the z coordinates of m, the centre point m' which forms the new position of point m can be acquired by the four nearest neighbour interpolation method. Figure 1f demonstrates the variance of the curve skeleton of the hand model during refinement. Some parts of the curve skeleton in the fingers are more centred after adjustment.

Animation Skeleton Generation

The curve skeleton produced so far gives us an approximation. To make it useful for character rigging, we must convert it to an animation skeleton. Here we discuss the algorithms to achieve this goal.

An animation skeleton is composed of a series of skeletal joints with connectivity. Our work in this section is primarily to locate the skeletal joints properly on the curve skeleton, based on the derived curve skeleton. The final animation skeleton is one that links all the skeletal joints with straight line segments.

A 3D medial axis (curve skeleton) ends at the midpoint of the internal edges when it meets the large junction triangles (Figure 2c). These end and start points of the 3D medial axes form the key skeletal joint points for the animation skeleton. They are marked in blue in Figure 2e. For the hand model, the wrist, palm and five fingers form seven branches, which contain seven 3D medial axes and 14 key skeleton points. After these key skeleton points are connected, the hierarchical structure of the model is constructed and a coarse discrete animation skeleton is generated.

The next step is to identify the middle joint points, i.e. those between the start and end joint points of each 3D medial axis branch. Two methods are developed to locate these joints automatically. However, each has its pros and cons. The first is the so called down sampling method.[5] This is based on the assumption that the skeleton links exhibit obvious rigid body rotation displacements around a joint. Therefore, for each node with two neighbours on the curve skeleton branches, we calculate the bending angle between its adjacent nodes. If the bending angle is more than a threshold (we use 18°), we treat this node as a skeletal joint. The advantage is that it is not restricted by the types of the models, whether it is a human-like character or an octopus. This however is not without limitations. If at the pose where the animation skeleton is extracted, a joint is not bent, then it is not able to identify the joint. For example, one cannot identify the elbow if the arm is fully extended. The second method which can rectify this deficiency is to produce a database of skeleton templates covering different types of characters. Technically speaking, it is relatively straightforward. For the current implementation, we have not included this function, which will be left as our future work.

Automation is helpful, but should not be perceived as a replacement of the craftsmanship of the animator. The animator is therefore given the flexibility to make the final decision. On this note, the user is allowed to interactively add, remove joints and adjust the positions of the joints produced by the automatic algorithm.

After all skeletal joints are located, a special joint is selected to serve as the root joint for the skeleton.[15] The final result of an animation skeleton generated for a hand model is given in Figure 2f. The green nodes represent the key skeletal joints, and the blue nodes represent the middle joints.

Skin Attachment

To apply deformations of the skeleton to the character mesh, the final step of rigging is skin attachment.

DOI: 10.1002/cav

Though a variety of skinning techniques can be adopted, we choose the linear blend skinning method (LBS) due to its widespread use.[11]

Given a skeletal configuration c, if v_d is the location of a vertex at its initial pose, the deformed result v_c can be computed by

$$v_c = \sum_{i=1}^{n} w_i \mathbf{M}_{i,c} \mathbf{M}_{i,d}^{-1} v_d \tag{4}$$

where w_i are the weights, $\mathbf{M}_{i,c}$ denotes the transformation matrix associated with the ith joint in configuration c, and $\mathbf{M}_{i,d}^{-1}$ denotes the inverse of the transformation matrix associated with the ith influencing joint at the binding pose. To attach the skin to the skeleton, one needs to assign weights w_i for each vertex. There are numerous algorithms available.[11,12,16,17] Here we use the Laplace's diffusion equation-based method[11] to compute the weights.

Experiments and Comparison

Figure 3a illustrates the animation sequence of a hand model rigged by our method. The skeleton preserves primary features and connectivity of the original model and it also conforms to the topology of the hand.

Figure 4 shows the animation skeleton extraction of some other 3D character models. Table 1 shows the execution times for different stages. The timing statistics were obtained using a 1.6 GHz Intel Core Dual PC with 1 GB of memory. For a model with n vertices, assuming the vertex number of the primary 3D silhouette is m and the number of curve skeleton branches is l, the computation complexity for primary 3D silhouette detection, curve skeleton extraction, joint location, skin attachment are $O(m^2)$, $O((m^2/l))$,$O(m)$ and $O(n)$ respectively. Generally, m can be treated as the maximal contour length (perimeter) of the model and n can be regarded as the surface area.$O(m^2)$equals $O(n)$approximately. So the total time complexity of the rigging algorithm in our paper is $O(n)$. According to the experimental analysis mentioned in Reference [2], the computation complexity for the main typical skeletonization methods mentioned in the Related Work is $O(n^2)$. Therefore, our skeletonization method is more efficient. Table 1 suggests that on average we can generate the animation skeleton of a 3D model with around 30 000 triangles in less than 10 seconds, which is much faster than most skeletonization methods.

We also compare our curve skeleton extraction results with five main classes of skeletonization methods discussed in the Related Work, which are medial surface, Reeb Graph, thinning, distance-field and potential-field methods. Figure 3b shows the comparison results. The second subfigure (from left to right) is a standard manually rigging result of a hand model.[18] The skeletons extracted with the thinning and distance-field algorithms are not smooth and contain many small branches. The centrality of the resulting skeleton in the Reeb Graph algorithm is less than ideal. The

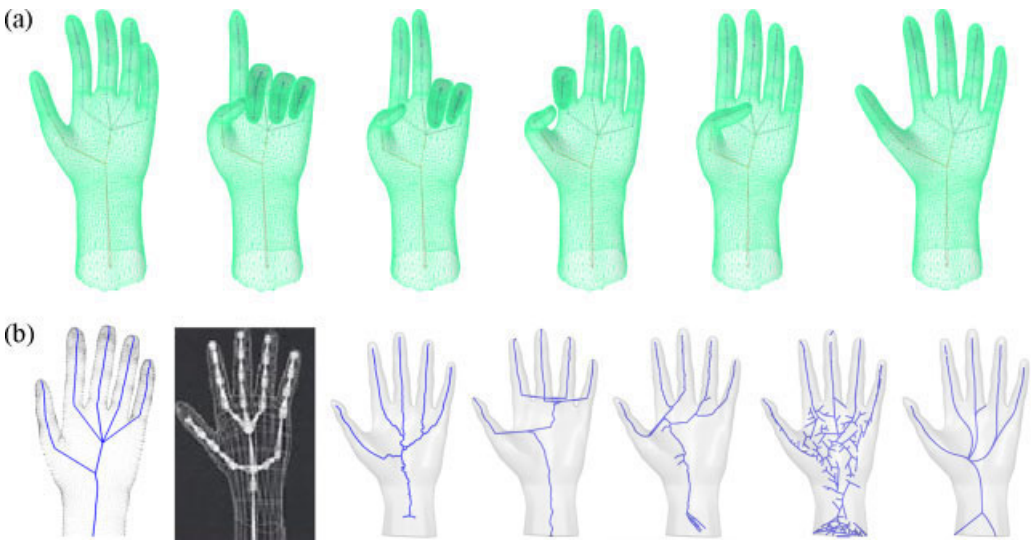


Figure 3. Experimental result and comparison. (a) An animation sequence of the hand model rigged with our method, (b) comparisons with other curve skeleton extraction algorithms. From left to right: our method, a standard rigged hand model[18], medial surface, Reeb Graph, thinning, distance field, potential field.

DOI: 10.1002/cav

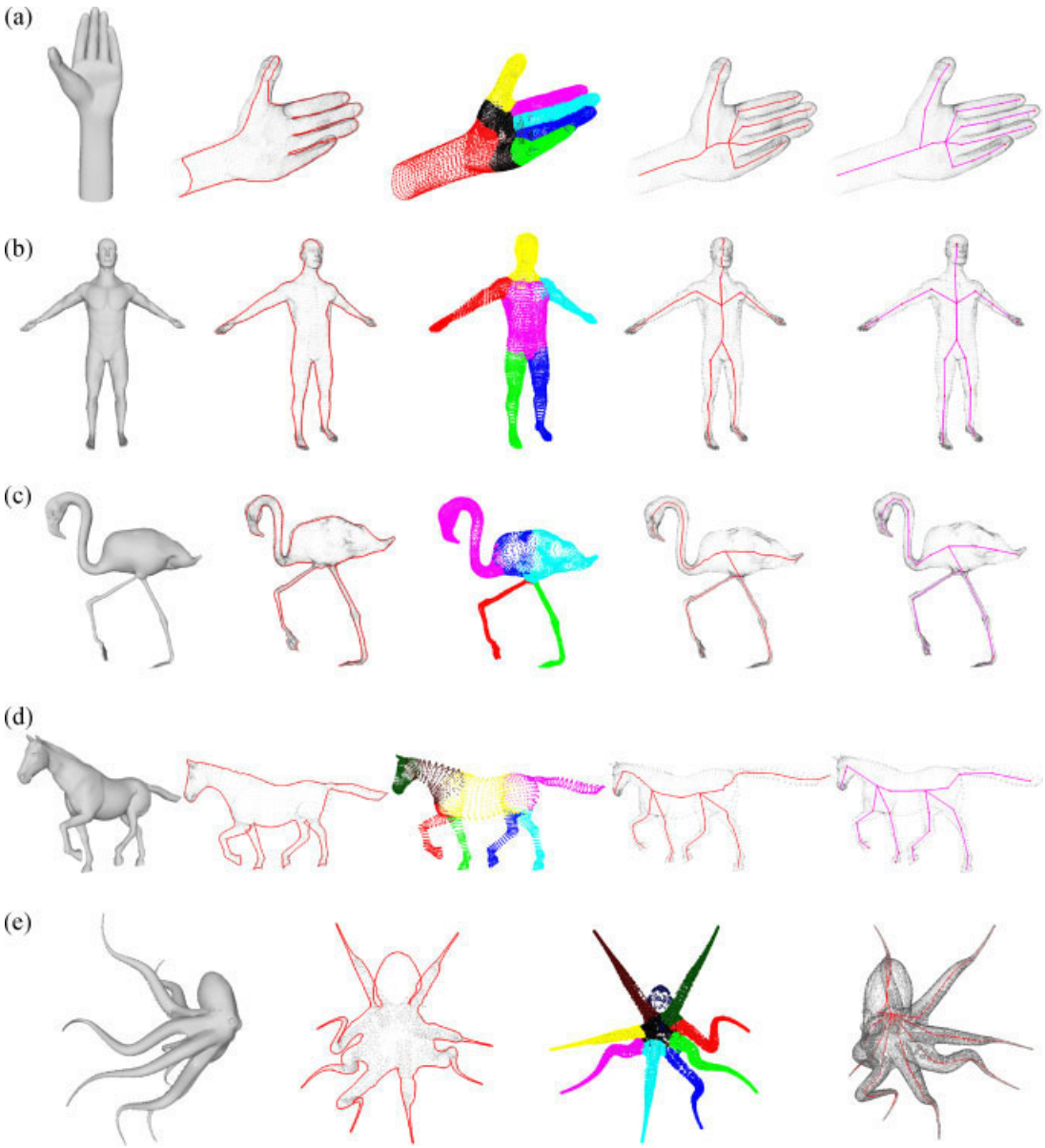


Figure 4. Original model, primary 3D silhouette, decomposition result, curve skeleton and animation skeleton of five 3D models. (a) Hand, (b) human, (c) ostrich, (d) horse, (e) octopus.

Model	Hand	Human	Ostrich	Horse	Octopus
Face number	38 016	29 216	52 895	16 843	48 362
Primary 3D silhouette detection	4.2	3.7	9.2	2.2	8.1
Curve skeleton extraction	2.0	1.9	5.4	1.2	4.8
Joints location	0.4	0.3	0.5	0.2	0.5
Total time	6.6	5.9	15.1	3.6	13.4

Table I. Execution time statistics for five 3D models (seconds)

DOI: 10.1002/cav

potential-field and medial surface algorithms yield cleaner and smoother results. However, they do not always preserve the connectivity and post-processing, such as pruning is required. In contrast, the curve skeleton extracted by our method is clean, connected, centred and topology preserving. It is the result which is the closest to the standard rigged animation skeleton.

Conclusion and Limitations

In this paper, we present an automatic rigging algorithm suitable for characters with various different shape and topology. We introduce a geometric entity, called the 3D silhouette, which are detected from two perpendicular projections of a character model. Using the detected 3D silhouettes, one can extract a curve skeleton automatically with constrained Delaunay triangulation. This gives the rough shape of the character's skeleton. By studying the triangulated primary silhouette, we are able to identify the necessary joints of an animation skeleton. Current skeletonization methods are typically of $O(n^2)$, whiles our method is of $O(n)$ complexity with n vertices. Therefore, ours is much faster and particularly suitable for the production of quick character animation.

Nevertheless, this algorithm also has some limitations due to its special aim. The main disadvantage is that the algorithm is restrictive with the shape of the objects. Although it can handle most character models, if the model cannot be treated as a series of generalized elliptic columns, such as a rock with grotesque shape, the generated skeleton might not be centred. Another shortcoming is that it is sensitive to occlusion where we cannot project all the branches and trunks of the object to the primary projection plane. Although in practice, it is not a restriction to animation production, as the binding pose in animation is always without occlusion, we would still like to develop a more general method.

In future, we plan to decompose the object first to extract the curve skeleton for each branch respectively, and then combine them to form a final skeleton. Several decomposition methods[19] offer good references in this area. In addition, we also intend to develop a template-based method for locating the middle joins more accurately. We consider two types of templates. The first is for the full-body of a character where all joints are pre-defined. The second is simply for a segment where only the joints associated with the body segment are given. For example, for an arm model, we only need to specify the middle joint (i.e. the elbow). Since we have already identified the start and end joints (i.e. the wrist and the shoulder), such a simple model works just as effectively. Thus any middle joints can be easily located by comparing the medial axis branch with the matching template.

ACKNOWLEDGEMENTS

This research is in part supported by the GreatWestern Research Fellowship grant. All the models used for this paper are from 500 3D-Objects, the Ultimate Human (http://www.cgcharacter.com/ultimatehuman.html) and the mesh data provided by Robert Sumner and Jovan Popović in the Computer Graphics Group at MIT.

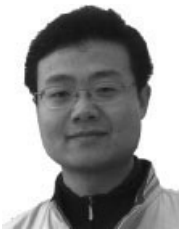
References

1. Cornea D, Silver D, Min P. Curve-skeleton properties, applications and algorithms. *IEEE Transactions on Visualization and Computer Graphics* 2006; **6**(3): 81–91.
2. Wu FC, Ma WC, Liang RH, Chen BY, Ouhyoung M. Domain connected graph: the skeleton of a closed 3D shape for animation. *The Visual Computer* 2006; **22**: 117–128.
3. Dey K, Sun J. Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of the Eurographics Symposium on Geometry* 2006; 143–152.
4. Tierny J, Vandeborre JP, Daoudi M. 3D mesh skeleton extraction using topological and geometrical analyses. *The Visual Computer* 2006; **10**: 120–130.
5. Au K, Tai L, Chu K, Daniel O, Lee Y. Skeleton Extraction by Mesh Contraction. In *Proceedings of ACM SIGGRAPH* 2008; 124–132.
6. Wang Y, Lee Y. Curve skeleton extraction using iterative least squares optimization. *IEEE Transactions on Visualization and Computer Graphics* 2008; **5**: 196–209.
7. Sabry M, Farag A. Robust centerline extraction framework using level sets. In *Proceedings of CVPR* 2005; 458–465.
8. Shilane P, Min P, Kazhdan M, Funkhous T. The Princeton shape benchmark. In *Shape Modelling International* 2004; 191–199.
9. Grigorishi T, Yang Y-H. Skeletonization: an electrostatic field-based approach. *Pattern Analysis and Application* 1998; **1**: 163–177.
10. Liu PC, Wu FC, Ma WC, Liang RH, Ouhyoung M. Automatic animation skeleton construction using repulsive force field. In *11th Pacific Conference on Computer Graphics and Applications* 2003; 323–329.
11. Baran I, Popvic J. Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics* 2007; **26**(3): 223—230. Article 72.
12. Aujay G, Hetroy F, Lazarus F, Depraz C. Harmonic skeleton for realistic character animation. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)* 2007; 144–147.
13. Jarvis RA. On the identification of the Convex Hull of a finite set of points in the plane. *Information Processing Letters* 1973; **2**: 18–21.
14. Igarashi T, Matsuoko S, Tanaka H. Teddy: a sketching interface for 3D freeform design. In *Proceedings of ACM SIGGRAPH 99*, 212–219.

DOI: 10.1002/cav

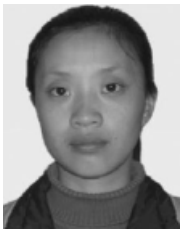
15. Wade L, Richard EP. Automated generation of control skeletons for use in animation. *The Visual Computer* 2002; **18**(2): 97–110.
16. James DL, Twigg CD. Skinning mesh animations. In *Proceedings of ACM SIGGRAPH 05*, 312–319.
17. Schaefer S, Yuksel C. Example-based skeleton extraction. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* 2007; 153–162.
18. Graft L, Dwelly B, Riesberg D, Mogk C. Learning maya: character rigging and animation. *Alias|Wavefront* 2002.
19. Katz S, Leifman G, Tal A. Mesh segmentation using feature point and core extraction. *The Visual Computer* 2005; **10**: 37–48.

Authors' biographies:



Junjun Pan is a Ph.D. student in the National Centre for Computer Animation, Media School, Bournemouth University, United Kingdom. He received his bachelor (2003) and master degree (2006) in Computer Science from Northwestern Polytechnical University (China). His research interests include computer animation, medical visualization and computer vision.

Xiaosong Yang is a research fellow in the National Centre for Computer Animation, Bournemouth Media School, Bournemouth University, United Kingdom. He received his bachelor (1993) and master degree (1996) in Computer Science from Zhejiang University (P. R. China), Ph.D. (2000) in Computing Mechanics from Dalian University of Technology (P. R. China). He worked as research assistant (2001–2002) at the 'Virtual Reality, Visualization and Imaging Research Centre' of Chinese University of Hong Kong. His research interests include 3D modelling, animation, real-time rendering, virtual surgery simulation and computer aided design.



Xin Xie was conferred Bachelor of Engineering degree in Electrical Engineering by University of Electronic Science and Technology of China in July 2001, and worked as a research assistant in Department of Computer Science of Southwest Normal University from July 2001 to June 2003. Currently, she is a Ph.D. student in Computer Animation at Bournemouth University.

Philip Willis is a Professor of Computing and Director of the Media Technology Research Centre at the University of Bath and Head of the Department of Computer Science. He is also a Fellow, past Chair of the Eurographics Association and the first Eurographics-ACM SIGGRAPH joint member. He is a founding member of UKCRC (UK Computing Research Committee). His research interests are within colour raster graphics, computer games, virtual reality and animation and film technologies.

Jian J Zhang is a Professor of Computer Graphics at the National Centre for Computer Animation, Media School, Bournemouth University and Head of Research at Bournemouth Media School, United Kingdom. His research interests include computer graphics, computer animation, physically based simulation, geometric modelling, medical simulation and visualization.

DOI: 10.1002/cav